ABSTRACT

Background: Expert judgment is the most frequently practiced approach to estimate software development effort. In Agile Software Development (ASD), effort is typically estimated by teams using expert judgment based approaches. Expert effort estimation, being lightweight and practitioner/team-centric, aligns well with ASD contexts. However, in the absence of explicit guidelines, expert effort estimation can be ad-hoc with inconsistent results. Further, due to the lack of emphasis on documentation in ASD, important estimation experience is also likely to be lost.

Objective: This thesis has two main objectives: First, to identify and organize evidence from both literature and industry on effort estimation in ASD contexts. Second, using the identified and organized knowledge, to develop and evolve checklists to support and improve the expert effort estimation processes of agile teams.

Method: In this thesis we conducted a combination of primary (one industrial survey, one industrial case study involving one company and one multi-case study involving three companies) and secondary studies (one systematic literature review and one systematic mapping study).

Results: The systematic literature review and survey showed that most agile teams estimate software development effort using expert estimation approaches. Further, team-related cost drivers, such as a team’s skill level, are considered most important. The results showed that organizing the knowledge on effort estimation in agile contexts as a taxonomy was helpful in characterizing the effort estimation cases from the literature and industry. Furthermore, the idea of using estimation checklists improved the expert estimation processes in agile contexts by increasing the accuracies of the effort estimates and decreasing the underestimation bias. The study participants from the case companies observed that the use of the checklist during estimation improved task understandability, reduced the likelihood of missing tasks and factors, increased the confidence in effort estimates and lead to more objectivity in the effort estimation process.

Conclusions: This thesis proposes an effort estimation taxonomy that can be used as a guideline for reporting effort estimation studies in ASD to facilitate aggregation and analysis of evidence. This thesis also proposes a process for developing and evolving customized estimation checklists and demonstrated its application in three software companies. The thesis empirically validated the usefulness of the checklists in improving expert effort estimation processes of three companies. In two case companies, the checklists have become part of their regular estimation processes. In the third case company, the project manager continues to use the checklist factors to prioritize the product backlog.
Improving Expert Estimation of Software Development Effort in Agile Contexts

Muhammad Usman
Improving Expert Estimation of Software Development Effort in Agile Contexts

Muhammad Usman

Doctoral Dissertation in Software Engineering

Department of Software Engineering
Blekinge Institute of Technology
SWEDEN
To my family, friends and teachers

“We must accept human error as inevitable, and design around that fact.”

—Donald Berwick
Abstract

Background: Expert judgment is the most frequently practiced approach to estimate software development effort. In Agile Software Development (ASD), effort is typically estimated by teams using expert judgment based approaches. Expert effort estimation, being lightweight and practitioner/team-centric, aligns well with ASD contexts. However, in the absence of explicit guidelines, expert effort estimation can be ad-hoc with inconsistent results. Further, due to the lack of emphasis on documentation in ASD, important estimation experience is also likely to be lost.

Objective: This thesis has two main objectives: First, to identify and organize evidence from both literature and industry on effort estimation in ASD contexts. Second, using the identified and organized knowledge, to develop and evolve checklists to support and improve the expert effort estimation processes of agile teams.

Method: In this thesis we conducted a combination of primary (one industrial survey, one industrial case study involving one company and one multi-case study involving three companies) and secondary studies (one systematic literature review and one systematic mapping study).

Results: The systematic literature review and survey showed that most agile teams estimate software development effort using expert estimation approaches. Further, team-related cost drivers, such as a team’s skill level, are considered most important. The results showed that organizing the knowledge on effort estimation in agile contexts as a taxonomy was helpful in characterizing the effort estimation cases from the literature and industry. Furthermore, the idea of using estimation checklists improved the expert estimation processes in agile contexts by increasing the accuracies of the effort estimates and decreasing the underestimation bias. The study participants from the case companies observed that the use of the checklist during estimation improved task understandability, reduced the likelihood of missing tasks and factors, increased the confidence in effort estimates and lead to more objectivity in the effort estimation process.

Conclusions: This thesis proposes an effort estimation taxonomy that can be used as a guideline for reporting effort estimation studies in ASD to facilitate aggregation and analysis of evidence. This thesis also proposes a process for developing and evolving customized estimation checklists and demonstrated its application in three software companies. The thesis empirically validated the usefulness of the checklists in improving expert effort estimation processes of three companies. In two case companies, the checklists have become part of their regular estimation processes. In the third case company, the project manager continues to use the checklist factors to prioritize the product backlog.
Acknowledgements

First of all, I am grateful to God for blessing me with the ability and opportunity to complete this thesis.

I would like to thank my supervisors Prof. Jürgen Börstler and Prof. Kai Petersen for their support, guidance, and feedback on my work. I have learned a lot from our discussions and their critique of my work. Beyond work, they have been instrumental in providing me the opportunities to grow as an independent researcher. I am also thankful to my previous supervisor, Prof. Emilia Mendes, for her support during the initial phases of my Ph.D. studies.

I would like to thank Prof. Darja Šmite for supporting me to perform a study at Ericsson. I am also thankful to the industry partners (Ericsson, Infoway and Diya Tech) for allowing me to conduct studies with them. In particular, Lars-Ola Damm at Ericsson, Edmilson Frank and Stênio Galvão at Infoway and Faisal Iqbal at DiyaTech.

I am thankful to my colleagues at SERL Sweden for providing a healthy and constructive work environment. In particular, Dr. Ricardo Britto for collaborating on studies and all the great times we had beyond work. I would like to express my gratitude to Dr. Nauman bin Ali for all the productive discussions and feedback on my work. I am also thankful to Deepika Badampudi, Dr. Ahmad Nauman Ghazi and Nasir Minhas for the nice time we had during courses, conferences and coffee breaks.

I would not have been able to complete this thesis without my family’s support. I am grateful to my parents for their encouragement, love, and sacrifice to support me in my work. I am also thankful to my wife Bushra for her continuous support throughout this period. I would not have been able to complete this thesis without the patience and love of Bushra, and our children Hassan, Amna, and Huda. I would also like to express my gratitude to my siblings Riffat Iqbal, Bilal Jadoon, Dr. Arshad Jamal and Farhat Tanveer for their support and prayers.

Last but not the least, I would like to express my sincere gratitude to my teacher and master thesis supervisor, Prof. Naveed Ikram for motivating and inspiring me to continue research in software engineering. Beyond research, he has been an exceptional role model and a great mentor.
Publications

Papers included in this thesis:


Muhammad Usman lead the design, execution, and reporting of the studies 1, 2, 4, 5 and 6. The co-authors provided their feedback and also contributed to the planning, execution and reporting stages. In Study 1 co-authors also performed cross-validation during the study selection phase. Emilia Mendes reviewed the Manuscript and also did editorial work.

In study 2 co-authors contributed to the design of the data collection instrument. They also reviewed the Manuscript and performed editorial work. Study 3 was lead together by Muhammad Usman and Ricardo Britto. Both authors shared the responsibilities
during the planning, execution and reporting phases of the study. The last two authors provided feedback during all these phases, and also performed cross-validation during study selection and data extraction processes. In Study 4 the co-authors reviewed the results and also performed the editorial work on the Manuscript. In Study 5, Ricardo Britto contributed to the study design and the data preparation steps, and also contributed in the Manuscript, mainly in the related work, methodology and validity sections. The last two co-authors contributed to the planning and analysis phases, and also did some editorial work.

In Study 6, Kai Petersen and Jürgen Börstler reviewed the study design. They also provided their feedback during the execution and analysis phases and contributed to reviewing and improving the Manuscript as well. Pedro Neto participated in the analysis phase, and also reviewed and improved the Manuscript.
Papers that are related to but not included in this thesis


Other Papers not included in this thesis

**Paper 1.** Nauman bin Ali, Muhammad Usman. “Reliability of search in systematic reviews: towards a quality assessment framework for the automated-search strategy”, Accepted for publication at *Information and Software Technology*, 2018.


Contents

1 Introduction ............................................................... 1
  1.1 Overview .......................................................... 1
  1.2 Background ...................................................... 4
    1.2.1 Effort estimation .......................................... 4
    1.2.2 Agile Software Development (ASD) ...................... 5
    1.2.3 Estimation in ASD context ............................... 6
    1.2.4 Use of checklists in other areas ....................... 6
    1.2.5 Expert judgment and checklists in software effort estimation . 8
  1.3 Research Gaps and Contributions .............................. 9
  1.4 Research Questions ............................................ 11
  1.5 Research Methods ............................................. 12
    1.5.1 Systematic literature review ............................ 12
    1.5.2 Systematic mapping study ............................... 13
    1.5.3 Industrial survey ....................................... 14
  1.6 Validity threats and limitations ................................ 17
    1.6.1 Construct validity ....................................... 17
    1.6.2 Internal validity ....................................... 18
    1.6.3 External validity ....................................... 19
    1.6.4 Reliability .............................................. 19
  1.7 Summary of Studies Included in the Thesis .................... 20
    1.7.1 Study 1: Effort estimation in agile software development: A
           systematic literature review ............................ 20
    1.7.2 Study 2: Effort estimation in agile software development: A
           survey on the state of the practice ..................... 21
    1.7.3 Study 3: Taxonomies in software engineering: A systematic
           mapping study and a revised taxonomy development method . 22
    1.7.4 Study 4: Agile estimation taxonomy ........................ 23
### Contents

3.3.3 Validity Threats ................................................. 66
3.4 Results ................................................................. 67
  3.4.1 Respondents' Demographics ................................. 67
  3.4.2 RQ1: Estimation Techniques ................................. 70
  3.4.3 RQ2: Effort Predictors ......................................... 72
  3.4.4 RQ3: Estimation and Agile Development Context ....... 74
  3.4.5 RQ4: Estimation Accuracy .................................... 77
  3.4.6 RQ5: Reasons for Inaccurate Estimates in ASD ......... 81
3.5 Discussion ......................................................... 83
3.6 Conclusions ......................................................... 85
3.7 References ............................................................ 86

4 Taxonomies in Software Engineering: A Systematic Mapping Study and a Revised Taxonomy Development Method 89
  4.1 Introduction ....................................................... 89
  4.2 Background ....................................................... 91
    4.2.1 Taxonomy definition and purpose ......................... 91
    4.2.2 Subject matter ................................................ 91
    4.2.3 Descriptive bases / terminology .......................... 91
    4.2.4 Classification procedure .................................... 92
    4.2.5 Classification structure ...................................... 92
    4.2.6 Validation ..................................................... 93
  4.3 Research methodology ........................................... 94
    4.3.1 Research questions ........................................... 94
    4.3.2 Search process ................................................ 95
    4.3.3 Study selection process ..................................... 97
    4.3.4 Extraction process ........................................... 100
    4.3.5 Analysis process .............................................. 103
  4.4 Results ............................................................... 103
    4.4.1 General results ............................................... 103
    4.4.2 Classification scheme results .............................. 106
    4.4.3 RQ1 – Taxonomy understanding ............................. 111
    4.4.4 RQ2 – Subject matters ........................................ 111
    4.4.5 RQ3 – Utility demonstration ................................. 112
    4.4.6 RQ4 – Taxonomy structure .................................... 113
    4.4.7 RQ5 – Taxonomy use ........................................... 115
    4.4.8 RQ6 – Taxonomy development ............................... 117
    4.4.9 Threats to validity related to the mapping study ....... 119
  4.5 A revised taxonomy development method ...................... 120
6.4.3 RQ3: Which factors impact the accuracy of the effort estimates? 187
6.4.4 RQ4: How do the identified factors impact the accuracy of the effort estimates? 191
6.4.5 Results from the interviews 198
6.5 Discussion 198
6.5.1 Effort estimation processes (RQ1) 198
6.5.2 Accuracy of estimates (RQ2) 198
6.5.3 Factors affecting the effort estimates (RQ3 and RQ4) 200
6.6 Threats to validity 202
6.7 Conclusion 204
6.8 References 207
Appendix 6A: BREbias QE regression model details 212
Appendix 6B: BREbias AE regression model details 215

7 Developing and Using Checklists to Improve Expert Effort Estimation: A Multi-Case Study 219
7.1 Introduction 219
7.2 Background and related work 221
7.2.1 Checklists in other disciplines 221
7.2.2 Checklists in software engineering 221
7.2.3 Process support for software estimation checklists 222
7.2.4 Agile estimation taxonomy 223
7.3 Research methodology 224
7.3.1 Research context 224
7.3.2 Research questions 227
7.3.3 Data collection methods 228
7.3.4 Data analysis 229
7.3.5 Characterizing the three cases with the Agile Estimation Taxonomy (AET) [14] 231
7.4 Results and Analysis - RQ1 231
7.4.1 Understand estimation context 231
7.4.2 Develop and present initial checklist 235
7.4.3 Validate checklist statically 240
7.4.4 Validate checklist dynamically 242
7.4.5 Transfer and follow up 244
7.5 Results and Analysis - RQ2 246
7.5.1 Infoway 246
7.5.2 Diyatech 248
7.5.3 TSoft 250
## Contents

7.6 Discussion .......................... 251
   7.6.1 RQ1: Checklist development and evolution 251
   7.6.2 RQ2: Usefulness of the checklist in improving effort estimation 253
   7.6.3 Implications for research ..................... 254
   7.6.4 Implications for practice ...................... 255

7.7 Validity threats .......................... 256
   7.7.1 Construct validity ........................ 256
   7.7.2 Internal validity .......................... 257
   7.7.3 External validity .......................... 257
   7.7.4 Reliability ............................... 258

7.8 Conclusions and future work .................. 258

7.9 References .............................. 260

Appendix 7A: Checklist Versions .......................... 265
Appendix 7B: Data Collection Instruments ................. 271
### List of Figures

1.1 Thesis overview: Chapters, research methods and questions .......................... 15

3.1 Job roles of the respondents ........................................................................ 69
3.2 Experience in ASD ...................................................................................... 70
3.3 Effort Estimation Techniques ...................................................................... 71
3.4 Size Metrics ................................................................................................. 72
3.5 Cost Drivers ................................................................................................. 74
3.6 Estimation at agile planning levels ................................................................. 75
3.7 Estimation of development activities ............................................................ 77
3.8 Accuracy of estimates in ASD ...................................................................... 79

4.1 Employed systematic mapping process. ...................................................... 94
4.2 Search process. ............................................................................................ 95
4.3 Selection process. ......................................................................................... 98
4.4 Extraction process. ...................................................................................... 98
4.5 Analysis process. ........................................................................................ 104
4.6 Year and venue wise distributions. ............................................................... 105
4.7 Yearly distribution of primary studies by KAs. Horizontal axes represent the years (starting 1987), while vertical axes denote the number of taxonomies. ........................................................................................................... 107
4.8 Systematic map – knowledge area vs research type. ................................. 109
4.9 Systematic map – knowledge area vs presentation type. .......................... 110
4.10 Distribution of average number of citations per year. The x-axis shows ranges of average citations per year. The y-axis shows the number of primary studies. ...................................................................................... 116

5.1 First level of the Agile estimation taxonomy ................................................. 147
List of Figures

5.2 Estimation context dimension . . . . . . . . . . . . . . . . . . . . . 149
5.3 Effort predictors dimension . . . . . . . . . . . . . . . . . . . . . . 151
5.4 Estimation technique dimension . . . . . . . . . . . . . . . . . . . 151
5.5 Effort estimate dimension . . . . . . . . . . . . . . . . . . . . . . . 153
5.6 Extending the context dimension . . . . . . . . . . . . . . . . . . . 161

6.1 QE process for PC (Product Customization) tasks . . . . . . . . . . . 180
6.2 AE process for PC (Product Customization) tasks . . . . . . . . . . . 181
6.3 BREbias of QEs of PCs arranged on actual effort . . . . . . . . . . . 192
6.4 BREBias boxplots by size for QE . . . . . . . . . . . . . . . . . . . 192
6.5 BREbias of AEs of PCs arranged on actual effort . . . . . . . . . . . 193
6.6 BREBias boxplots by size for AE . . . . . . . . . . . . . . . . . . . 194
6A.1 Histogram of residuals . . . . . . . . . . . . . . . . . . . . . . . . 213
6A.2 Normal PP Plot of regression standardized residuals . . . . . . . . . 213
6A.3 Scatter plot standardized residuals and predicted values . . . . . . . . 214
6B.1 Histogram of residuals . . . . . . . . . . . . . . . . . . . . . . . . 216
6B.2 Normal PP Plot of regression standardized residuals . . . . . . . . . 216
6B.3 Scatter plot standardized residuals and predicted values . . . . . . . . 217

7.1 Checklist development process . . . . . . . . . . . . . . . . . . . . . . 233
7.2 Checklist versions . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 237
List of Tables

1.1 Research questions and thesis chapters .......................... 12
1.2 Research methods and thesis chapters .......................... 13

2.1 Keywords from known primary studies ............................ 39
2.2 Search results ...................................................... 40
2.3 Quality assessment checklist adopted from [7, 22] .............. 43
2.4 No. of papers in study selection and QA ......................... 44
2.5 Estimation techniques investigated ............................... 45
2.6 Accuracy metrics used ............................................. 46
2.7 Accuracy achieved by frequently used techniques .................. 46
2.8 Size metrics used ................................................... 47
2.9 Cost drivers ......................................................... 47
2.10 Domain of the dataset used ....................................... 49
2.11 Type of datasets/knowledge used ................................ 49
2.12 Agile methods used ................................................. 50
2.13 Development activity investigated ................................ 50
2.14 Planning levels investigated ...................................... 51

3.1 Respondents by continents and countries ......................... 68
3.2 Planning levels versus estimation techniques ........................ 76
3.3 Development activities versus estimation techniques ............. 78
3.4 Overall estimation residuals in ASD .............................. 79
3.5 Estimation error versus techniques and predictors ............... 81
3.6 Reasons for inaccurate estimates in ASD .......................... 83

4.1 Faceted analysis example ........................................... 93
4.2 Summary of search results ......................................... 97
<table>
<thead>
<tr>
<th>Table</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.3</td>
<td>Rationale for excluded studies</td>
<td>99</td>
</tr>
<tr>
<td>4.4</td>
<td>Data extraction form</td>
<td>101</td>
</tr>
<tr>
<td>4.5</td>
<td>Publication venues with more than two taxonomy papers.</td>
<td>106</td>
</tr>
<tr>
<td>4.6</td>
<td>Approaches for utility demonstration</td>
<td>113</td>
</tr>
<tr>
<td>4.7</td>
<td>Classification structure</td>
<td>113</td>
</tr>
<tr>
<td>4.8</td>
<td>Descriptive bases</td>
<td>114</td>
</tr>
<tr>
<td>4.9</td>
<td>Classification procedure types</td>
<td>114</td>
</tr>
<tr>
<td>4.10</td>
<td>Classification procedure descriptions</td>
<td>114</td>
</tr>
<tr>
<td>4.11</td>
<td>Number of primary studies (Frequency) by number of citations.</td>
<td>116</td>
</tr>
<tr>
<td>4.12</td>
<td>Number of citations (mean and median) for primary papers by knowledge area</td>
<td>117</td>
</tr>
<tr>
<td>4.13</td>
<td>Comparison between the original and the revised versions of Bayona-Oré’s method</td>
<td>121</td>
</tr>
<tr>
<td>4.14</td>
<td>An illustration of the revised method</td>
<td>125</td>
</tr>
<tr>
<td>5.1</td>
<td>Planning levels at which agile teams estimate</td>
<td>134</td>
</tr>
<tr>
<td>5.2</td>
<td>Development activities estimated</td>
<td>134</td>
</tr>
<tr>
<td>5.3</td>
<td>Agile methods practiced</td>
<td>135</td>
</tr>
<tr>
<td>5.4</td>
<td>Estimation techniques used in ASD</td>
<td>135</td>
</tr>
<tr>
<td>5.5</td>
<td>Size measures used in ASD</td>
<td>136</td>
</tr>
<tr>
<td>5.6</td>
<td>Cost drivers used in ASD</td>
<td>137</td>
</tr>
<tr>
<td>5.7</td>
<td>Taxonomy design method from [3]</td>
<td>140</td>
</tr>
<tr>
<td>5.8</td>
<td>Details of the interviewees</td>
<td>146</td>
</tr>
<tr>
<td>5.9</td>
<td>Taxonomy application: Literature</td>
<td>155</td>
</tr>
<tr>
<td>5.10</td>
<td>Characterizing effort estimation activity of company A</td>
<td>158</td>
</tr>
<tr>
<td>5.11</td>
<td>Characterizing effort estimation activity of company B</td>
<td>159</td>
</tr>
<tr>
<td>5.12</td>
<td>Characterizing effort estimation activity of company C</td>
<td>160</td>
</tr>
<tr>
<td>5.13</td>
<td>Comparison of facets in the estimation context dimension</td>
<td>163</td>
</tr>
<tr>
<td>5.14</td>
<td>Comparison of facets in the estimation technique dimension</td>
<td>163</td>
</tr>
<tr>
<td>5.15</td>
<td>Comparison of facets in the predictors’ dimension</td>
<td>164</td>
</tr>
<tr>
<td>5.16</td>
<td>Comparison of facets in the effort estimate dimension</td>
<td>164</td>
</tr>
<tr>
<td>5.17</td>
<td>Comparison of facets in the effort estimate dimension</td>
<td></td>
</tr>
<tr>
<td>6.1</td>
<td>Description of semi-structured interviewee sample</td>
<td>177</td>
</tr>
<tr>
<td>6.2</td>
<td>Mapping RQs with the used data collection methods</td>
<td>178</td>
</tr>
<tr>
<td>6.3</td>
<td>Statistical tests used in data analysis</td>
<td>179</td>
</tr>
<tr>
<td>6.4</td>
<td>Accuracy distribution of the two estimates (QE and AE)</td>
<td>185</td>
</tr>
<tr>
<td>6.5</td>
<td>Mean and Median BRE and BREbias for both estimates (QE and AE)</td>
<td>186</td>
</tr>
<tr>
<td>6.6</td>
<td>Testing the two estimates (QE and AE) for significant differences</td>
<td>186</td>
</tr>
</tbody>
</table>
List of Tables

6.1  Mean and median actual effort for all 60 PCs in work hours grouped by PC size ......................................................... 191
6.2  Mean and Median BREBias for both estimates (QE and AE) grouped by PC size .............................................................. 195
6.3  Mean and median BREBias for both estimates (QE and AE) grouped by PC priority ............................................................... 195
6.4  Mean and median BREBias for both estimates (QE and AE) across all maturity levels .......................................................... 196
6.5  Mean and median BREBias for both estimates (QE and AE) for two groups of teams .......................................................... 196
6.6  Mean and median BREBias for both estimates (QE and AE) for co-located and multi-site PCs .................................................. 197
6.7  Summary of results ......................................................................................................................................................... 199
6A.1 BREbias QE regression analysis: ANOVA .......................................................... 212
6A.2 BREbias QE regression coefficients’ details .......................................................... 212
6A.3 BREbias QE regression models’ summary .......................................................... 212
6B.1 BREbias AE regression: ANOVA .......................................................... 215
6B.2 BREbias AE regression coefficients’ details .......................................................... 215
6B.3 BREbias AE regression models’ summary .......................................................... 215
7.1  Potential checklist factors from previous works .......................................................... 224
7.2  Description of data collection mechanism in the case organizations .......................................................... 229
7.3  Demographic information of the study participants .......................................................... 230
7.4  Using AET to characterize estimation process of the three cases .......................................................... 232
7.5  checklist factors identified through interviews .......................................................... 235
7.6  Infoway checklist V1 ......................................................................................................................................................... 238
7.7  Diyatech checklist V1 ......................................................................................................................................................... 239
7.8  Tsoft checklist V1 ......................................................................................................................................................... 241
7.9  Benefits of using estimation checklist ......................................................................................................................................................... 248
7.10 Teams’ feedback on the relevance of the checklist factors to the effort estimation ......................................................................................................................................................... 248
7.11 Accuracy of estimates before and after checklist introduction at Infoway .......................................................... 249
7.12 Accuracy of estimates before and after checklist introduction at Diyatech .......................................................... 250
7A.1 Infoway checklist V2 (after initial feedback and static validation) .......................................................... 265
7A.2 Infoway checklist V3 (prioritized and classified after dynamic validation) .......................................................... 266
7A.3 Diyatech checklist V2 (after initial feedback and static validation) .......................................................... 267
7A.4 Diyatech checklist V3 (after dynamic validation) .......................................................... 268
7A.5 Tsoft checklist V2 (after initial feedback and static validation) .......................................................... 269
List of Tables

7A.6 Tsoft checklist V3 (after follow up discussion with project manager)  . 270
Chapter 1

Introduction

1.1 Overview

Software development effort is the main contributor to the cost of a software project. Effort estimation, therefore, plays an important role in supporting the budgeting, scheduling and planning activities and decisions during software development. There are many formal techniques (e.g., algorithmic models such as COCOMO [8]) to estimate development effort, but the most frequently used in the software industry is the expert judgment based software effort estimation [41, 42]. The expert judgment in these techniques can vary from pure gut feeling of the involved experts to a relatively more informed judgment based on past data, guidelines, and checklists [40].

Expert estimation of software development effort often involves minimal process and structure. The involved experts rely on their experience to arrive at effort estimates by considering many relevant factors that they think may contribute to the development effort. In the absence of any support structure (e.g., checklist), this process has the potential to be inconsistent, as experts may forget to consider relevant factors and activities in a given situation [40]. Overlooking tasks is one of the major reasons for the underestimation of the development effort by experts [24, 37]. Furthermore, in the absence of any documentation, the factors and aspects considered in estimating the effort for the specific set of requirements would also be lost.

Checklists can be used to support the expert estimation in overcoming the challenges mentioned above [40]. The involved experts use the checklist in recalling the factors, activities, and tasks that should be accounted for during effort estimation. Jørgensen [40] shared that in their experience the organizations find checklists very
useful in effort estimation. Checklists have also been found to improve the accuracy of the software effort estimates [23, 44]. Checklists are also useful in supporting the new team members in the estimation process [40]. Furthermore, the use of checklist would also document the important estimation data, which can be used in future in further improving the accuracy [43].

Despite the recognized benefits of checklists for expert estimation discussed above, we have not seen any evidence on how software effort estimation checklists can be developed and evolved. Further, there is some evidence (cf. [23, 44]) that checklists improve estimation accuracy. However, this limited evidence is not enough. There is a need to empirically validate the estimation checklists, their impact on effort estimates’ accuracy and also other potential benefits beyond just accuracy improvement.

This thesis has been performed in the context of Agile Software Development (ASD), which has been a widely researched and practiced paradigm since the emergence of famous Agile Alliance in 2001. In ASD software is developed incrementally in short iterations, and flexibly to incorporate changing customer requirements and market demands. The focus in ASD shifts to the planning activity as opposed to documented plans in plan-driven development. Agile planning is performed continuously at various levels to support flexibility.

Effort estimation is performed mainly during release and iteration planning stages in ASD. During release planning the issues of scope, schedule and resources are dealt with for the set of user stories that are to be developed in the next release of a product. A release plan is made initially at project inception; however, a good plan in ASD is continuously updated to reflect the current situation [3]. In iteration planning, based on the feedback from the previous iteration, high priority features are selected that the team should develop and test in the next iteration. At this stage, stories are divided into tasks that are assigned to team members to transform a feature request to a working software [3]. In ASD context too the expert judgment based approaches, such as planning poker and estimation by analogy, are most frequently practiced techniques to estimate development effort [36, 37, 45].

Effort estimation in ASD has been investigated in number of empirical studies [25–28]. This evidence has not yet been aggregated into one place. Agile methods and practices are being used in industry all over the world. We are not aware of any studies that attempt to identify the state of the practice on effort estimation in ASD from agile practitioners across the globe covering aspects related to techniques, predictors and estimation context. This thesis first identifies and aggregates knowledge, from literature and industry, on effort estimation in ASD context. The thesis aims to support research and practice on effort estimation in ASD by organizing the identified knowledge.
knowledge. It also proposes a method to design taxonomies in software engineering discipline. The method is then applied to design a taxonomy on agile effort estimation. The proposed taxonomy is used to characterize effort estimation activity of agile projects selected from the literature and the industry. Lastly, the thesis aims to use this organized knowledge to propose estimation checklist, and associated process support, to improve expert estimation process in ASD context.

Using a combination of primary (case studies, survey) and secondary studies (systematic literature review and mapping study), this thesis has made the following contributions:

1. Aggregated the reported evidence on effort estimation in Agile Software Development (ASD) context.
2. Elicited the practice on effort estimation in ASD context
3. Proposed a method to design taxonomies in software engineering in a systematic way
4. Proposed a taxonomy for effort estimation in ASD context, and used it to characterize the effort estimation of selected projects from the literature and industry.
5. Investigated effort estimation in large-scale distributed agile project, and analyzed how different factors in this context impact the accuracy of the effort estimates.
6. Proposed a process to develop and evolve estimation checklists to improve expert estimation process, and also demonstrated the usefulness of the proposed checklists in improving the effort estimation in three case companies.

The remainder of this chapter is organized as follows: Section 1.2 presents the background of the thesis by providing a brief description of the main software engineering areas investigated in this thesis. Research gaps and contributions are described in Section 1.3. Section 1.4 presents the research questions that are investigated in this thesis. Research methods employed in the studies included in this thesis are briefly described in Section 1.5. The discussion on the validity threats is presented in Section 1.6. Section 1.7 presents the summaries of the included studies. Section 1.9 concludes the chapter, and also presents the research directions that we are interested to investigate in the future.
1.2 Background

This thesis deals with the following software engineering areas: effort estimation, expert judgment based effort estimation process, use of checklists in effort estimation, and the agile software development as the context. These areas are elaborated in this section to provide a background in which the contributions made in this thesis can be understood.

1.2.1 Effort estimation

Effort estimation \(^2\) is one of the central activities in software project management. Inaccurate estimates of the development effort have lead to serious problems [4]. In today’s competitive business world, it is critical for software companies to develop quality products within given budget and time constraints. In this regard, underestimation of development effort would hurt the goodwill and competitiveness of the company [5], while overestimation would lead to suboptimal utilization of the precious human resource, which in turn would negatively impact cost-effectiveness [6].

A plethora of research is available on software estimation. This extensive literature on estimation models and techniques has repeatedly been reviewed, see for example [1, 6] for a detailed overview of the literature in software estimation. Many effort estimation techniques have been proposed during last three to four decades. They are broadly divided into three categories [7] as follows:

- **Algorithmic techniques** build models that correlate effort with one or many project characteristics. Product size is considered as the main contributor, while other characteristics (often referred as cost drivers) also contribute to total effort. COCOMO (Constructive COst MOdel) and COCOMO II [8, 9] are famous examples of algorithmic estimation models.

- **Artificial Intelligence (AI) based techniques** use various AI approaches to estimate effort. Several AI based techniques have been proposed in literature such as neural networks [11], case base reasoning [12], fuzzy logic [10] etc.

- **Expert estimation** based techniques rely on subjective assessment of experts to arrive at effort estimates based on past experience with similar projects. A review of studies [2] on expert estimation found that expert based effort estimation is

\(^2\)The terms effort and cost are often used interchangeably in the literature as the effort is the main contributor to software development project cost. However since project cost could also include other factors, we will use the word effort throughout this thesis.
the most frequently applied approach. This review [2] also found that the model estimates are not better than expert estimates in terms of accuracy.

1.2.2 Agile Software Development (ASD)

In plan driven software development processes emphasis is on thorough upfront planning and detailed documentation. These processes also encourage big upfront analysis and design, which results in delayed start of implementation and testing phases. Due to these reasons these approaches have longer development cycles, and are not suitable in situations wherein customer requirements and market needs change frequently.

Agile methods, on the other hand, promote a lightweight and flexible approach to software development. The differences between agile and plan driven are manifested in the four value statements of the agile manifesto [13]:

1. “People and interactions over processes and tools”: This is different from plan driven wherein high emphasis is placed on documented processes, and communication between phases is mainly supported through documentation.

2. “Working software over comprehensive documentation”: It also contradicts with plan driven in which detailed documentation of requirements and design is prepared and signed off before the implementation phase can start. ASD, on the other hand, encourages early start of the implementation phase to have a working software after a short development cycle.

3. “Customer collaboration over contract negotiation”: In plan driven development the requirements specification forms the basis of contractual arrangement with the customers, and are therefore not easily changeable without renegotiating the contract. While in ASD it is recognized that specification would change and evolve over time and development team need to collaborate with customer.

4. “Responding to change over following a plan”: In plan driven methods, as the name suggests, focus is on following a plan. It is difficult and costly to accommodate changes in plan driven approaches due to the lengthy development cycle, and generally a documented process to review changes. In ASD, on the other hand, teams work in short development cycles that allow them to incorporate changes in the coming iterations.

The movement towards these values started years ago, the agile manifesto [13] was officially created in 2001. The manifesto provided a name to what the signing practitioners were already doing.
1.2.3 Estimation in ASD context

The four value statements of the agile manifesto, described in Section 1.2.2 above, lead to highly iterative and incremental agile processes that deliver tested software in short iterations. In ASD, team members rely on tacit knowledge sharing to plan and manage their work in a flexible way, while plan-driven approaches use explicit documentation to share knowledge [31]. In plan-driven approaches, the emphasis is on detailed upfront planning, while in ASD the focus shifts to planning wherein the team is willing to adapt the plan to changing project goals and requirements [32]. In this highly iterative mode, new knowledge is continuously discovered, and customers frequently provide their feedback. These factors lead to changes in the agile plans.

There are multiple levels at which planning is carried out in ASD. Most agile teams plan at three different levels, i.e., release, iteration and current day level [3]. This allows them to focus and plan features that are visible. Estimation is a central activity in the planning process and is also performed at these levels in ASD to support planning. There are other types of planning as well, such as portfolio and product planning [3], but they are performed at a strategic level in the organization beyond the scope of individual agile teams.

During release planning, product features are expressed as user stories. At this level, the agile team estimates user stories and allocate them to specific iterations [3, 33]. Release planning involves broader estimation and planning of user stories to prepare a release plan, which includes details about which stories are included in the next product release. [38]. Iteration planning is performed in more detail and for much shorter time frame whereby stories are broken into tasks, which are then estimated by the developers or the entire development team [3, 33, 38]. Estimating during release planning involves more uncertainty due to many unknown factors related to product, team, and customer [33].

Most of the agile teams use expert judgment based techniques, such as planning poker, to estimate software development effort [36, 37, 45]. The estimation is mostly performed in a group setting; wherein all team members participate in the estimation discussions [38]. Members of the development team, which will later develop and test the requirements under estimation, act as the experts in the expert judgment based estimation in ASD context.

1.2.4 Use of checklists in other areas

In this section, we briefly describe the use of checklists in other disciplines such as aviation and healthcare. The discussion, in no way, is an exhaustive review of the use of checklists in these disciplines.
Human error is inevitable and should be considered as a fact while redesigning systems of work to improve performance and safety [49]. Experts are likely to miss routine tasks due to memory lapse or loss of attention, especially in pressure situations [50]. To support experts in avoiding such errors, checklists have been used as an error management tool in many disciplines (e.g., health care, aviation) [51]. The checklist includes a list of criteria that should be considered during a particular process [52]. They are used as a cognitive aid to experts in accurately completing tasks [52].

Checklists have long been part of the aviation industry to avoid pilot errors during flight operations (e.g., take-off, landing). After the crash of Boeing 3 Model 299 during the trial demonstration in 1935, an internal Boeing investigation found that the pilot forgot to release one of the newly added locks after take-off [53, 55]. To avoid these types of pilot errors, the only change the Boeing made was the introduction of a checklist, which included a list of critical tasks that pilots must complete. The use of checklists during flight operations has been mandated by the Aviation regulators (e.g., Federal Aviation Administration (FAA) in case of the USA) worldwide [53]. Electronic checklists started to replace paper checklists in aircrafts during the 90s to avoid crew errors associated with the paper checklists [54, 56].

In healthcare, checklists have been used to avoid errors and improve performance [51]. One of the most influential checklist related works in health care was initiated by Pronovost at John Hopkins Hospital 4 [58]. Pronovost, through a review of published studies, identified infection-control practices to avoid human mistakes, which were responsible for life threatening infections to patients in the surgical intensive care units [58]. He selected five practices, which had strong evidence and were easy to implement, and formulated them as a simple checklist. The nurses were told to intervene if they found doctors skipping any of the five practices. The intervention resulted in a large reduction (up to 66%) in infection rates [59], and was later adopted in the other states of the USA as well [60]. The checklists have been used in other aspects of healthcare as well. For example, Wolff et al. [57] observed that the checklists and reminders significantly improved the quality of inpatient care.

Checklists are mainly used for two purposes: 1) as an evaluative instrument providing reviewers a common set of criteria or guidelines to review an artifact or entity consistently, 2) as a mnemonic device to recall tasks or practices that should be followed in accurately executing a process [52, 66]. Evaluative checklists have long been used in software inspections area to support reviewers in consistently reviewing artifacts (e.g., code, requirements) [61]. However, checklists were not found to be more effective than ad-hoc reviewing and other reviewing techniques (e.g., scenario, per-

---

3http://www.boeing.com
4https://www.hopkinsmedicine.org/
Chapter 1. Introduction

spective, and usage-based reviewing) in detecting faults in requirements and design documents in many studies (cf. [62–65]). Evaluative checklists are also used in empirical software engineering research guidelines to improve the design, execution, and reporting of the empirical studies in software engineering (cf. [15, 35]. In software effort estimation area, checklists have successfully been used as a reminder or mnemonic instrument by experts to avoid forgetting important activities and tasks [40] (see details in next section).

1.2.5 Expert judgment and checklists in software effort estimation

Many studies found that the software industry prefers expert judgment based techniques to estimate development effort. Expert judgment based estimation is practiced in many ways, ranging from individual estimation involving a single expert to group estimation involving multiple experts. Group estimates have been found to be more accurate than the estimates by individual experts [24]. The involved expert(s) rely on their knowledge of the domain, product and context to estimate development effort [38]. The expert judgment can vary from a simple “rule-of-thumb” to a relatively more structured reasoning such as work breakdown structure technique, and/or relying on the previous data and using guidelines and checklists [38, 40] during effort estimation.

In group estimation, the involved experts can either be a dedicated group of estimation experts or team members involved in product development [24]. A dedicated group of estimation experts can help reduce the personal biases and political considerations, as it would not exist inside the project organization. This group is more likely to develop expertise in estimation due to their dedicated focus on estimation [47]. This setup, however, requires additional resources and an effective communication strategy at the organization level [24, 38].

The alternative, wherein development team itself estimates the requirements, is a resource efficient solution. The development team has the advantage of having better knowledge of the project, product and team capabilities [24]. However, there is a risk of biasness in estimates as some people/roles involved in the project (e.g., project managers) may have interest in under or overestimating the effort [24]. Previous studies [5, 46] found that the accuracy of the estimates is better when members of the development team are involved in the estimation. In all cases investigated in this thesis (Chapter 5, 6 and 7), the members of the development team are involved in the estimation process.

In the absence of any explicit support structure, expert judgment can become an ad-hoc activity. Experts may forget relevant tasks and factors that should be considered while estimating a specific set of requirements [40]. Overlooked tasks [24, 48] and forgetting activities (e.g., non-functional requirements, testing) [37] are cited as reasons
for underestimating the development effort. Due to these reasons, expert estimates are inconsistent. Checklists have been used to overcome these issues, and as a mechanism to support expert estimation process [40]. Checklists support experts in consistently using the same set of factors, and considering the same set of activities and tasks during effort estimation. It also provides a low-cost mechanism for documenting the important estimation experience, which can be used in future to improve estimation accuracy [40].

Furulund and Molokken-Ostvold [23] found that the use of checklists improves the accuracy of effort estimates. They noted that the estimates of the projects that used checklists were more accurate than those projects where checklists were not used to estimate development effort. The checklists supported the experience sharing, also helped in addressing the problem of overlooked tasks.

Passing and Shepperd [44] found that the use of checklist improved the consistency and transparency of the size and effort estimates. They also noted that the checklist increased the subjects’ confidence in their estimates. It helped student estimators in not forgetting important factors during estimation. Trendowich and Jeffery [38] also suggested that the use of checklists improves the accuracy, transparency, and confidence in the effort estimates.

Jorgensen [22] proposed a preliminary estimation checklist based on multiple sources, both from literature and software industry. This checklist has quite a wider scope, and is structured according to a cost management process framework consisting of four phases (preparation, estimation, application and learning). Our idea is to propose a process for developing a more specific checklist that focuses only on the estimation activity, and which is lightweight as well, so that it is suitable for agile teams.

The benefits of using checklists during effort estimation are observed in multiple studies, as discussed above. However, we have not seen guidance on how such checklists can be developed and evolved to support expert estimation process. There is also need to empirically validate: 1) the checklist development process in real contexts, 2) potential benefits of using such checklists during effort estimation, such as improvement in accuracy and confidence, etc.

1.3 Research Gaps and Contributions

The research gaps investigated in this thesis, and the corresponding contributions are described below.

- Gap 1: Effort estimation in ASD has been investigated in multiple studies but this evidence has not been aggregated so far. Gap 1 was identified during initial
Chapter 1. Introduction

literature review of the thesis topic.

Contribution 1: This thesis contributes by aggregating the evidence on effort estimation in ASD by performing a Systematic Literature Review (SLR).

• Gap 2: There are not many research efforts, to the best of our knowledge, to identify the state of the practice on effort estimation in ASD covering aspects such as estimation context, techniques and effort predictors used in practice. This gap was identified during execution and analysis phases of the aforementioned SLR. Only a handful of primary studies of the SLR were performed in real industrial settings. It is important to note that the SLR only considered peer reviewed empirical works as primary studies.

Contribution 2: This thesis contributes by identifying the state of the practice on effort estimation in ASD by directly eliciting agile practitioners’ experiences in a survey study.

• Gap 3: Taxonomies are frequently used in software engineering to organize knowledge in different areas. However, in most cases they are not designed in a systematic way.

Contribution 3: We identified in our initial search that taxonomies have been frequently used in software engineering discipline to effectively organize and classify knowledge. However, we noted that there is a need to improve the methodological support to organize the knowledge in software engineering. This thesis contributes by identifying the state of the art on the use of taxonomies in software engineering through a mapping study. A method to design taxonomies is also proposed in this thesis.

• Gap 4: The knowledge on effort estimation in ASD has not yet been organized.

Contribution 4: This thesis organizes the knowledge on effort estimation in ASD as a taxonomy by applying the taxonomy design method mentioned above in Contribution 3. The usefulness of the proposed taxonomy was demonstrated to characterize the effort estimation activities of agile projects reported in the literature. The taxonomy was also applied to characterize estimation activities of the agile teams of three different software companies.

• Gap 5: There is a lack of empirical studies investigating factors impacting the accuracy of the effort estimates. In particular, investigating how factors that are relevant in the large scale distributed agile context impact the accuracy of the effort estimates.
Contribution 5: This thesis contributes by eliciting how effort estimation is managed in a large-scale distributed software project, and how different factors such as requirements size, team maturity and involvement of distributed teams impact the accuracy of the effort estimates.

• Gap 6: Most of the software teams and companies use expert judgment based techniques to estimate development effort. Expert estimation process can be inconsistent and ad-hoc in the absence of any structure, and over-reliance on the memory of the involved experts.

Contribution 6: This thesis contributes by proposing a method to design and evolve checklists to support expert estimation process. We demonstrated the usefulness of the proposed checklists in three case companies. The checklists improved the estimation process of the case companies, and also improved the accuracy of the effort estimates.

1.4 Research Questions

The main objective of this thesis is to improve the expert judgment based effort estimation process in an agile context. This thesis has attempted to answer the following research questions:

• RQ1: How is effort estimated in Agile Software Development (ASD) context?
  – RQ1a: What evidence is reported in the literature on effort estimation in ASD context?
  – RQ1b: How is effort estimation practiced by teams working in ASD context?
  – RQ1c: Which factors impact the accuracy of the effort estimates in large-scale agile context?

• RQ2: How to organize knowledge on the effort estimation in ASD context to support research and practice?
  – RQ2a: How are taxonomies used to organize the knowledge in software engineering?
  – RQ2b: How can agile estimation taxonomy support research and practice on effort estimation?
Chapter 1. Introduction

• **RQ3**: How to improve the expert judgment based effort estimation process in ASD context?

  – **RQ3a**: How to develop and evolve checklists to improve expert judgment based effort estimation?
  – **RQ3b**: How useful are estimation checklists in improving expert judgment based effort estimation?

Table 1.1 explains which research questions are addressed in which chapters.

<table>
<thead>
<tr>
<th></th>
<th>Chapters</th>
</tr>
</thead>
<tbody>
<tr>
<td>RQ2</td>
<td>RQ1a</td>
</tr>
<tr>
<td>2</td>
<td>RQ1b</td>
</tr>
<tr>
<td>3</td>
<td>RQ2a</td>
</tr>
<tr>
<td>4</td>
<td>RQ2b</td>
</tr>
<tr>
<td>5</td>
<td>RQ1c</td>
</tr>
<tr>
<td>6</td>
<td>RQ3</td>
</tr>
</tbody>
</table>

1.5 **Research Methods**

This thesis has employed multiple research methods to answer the research questions. It can therefore be regarded as mix methods research [14]. Research methods were selected based on their appropriateness to answer each research question. Table 1.2 depicts research methods used in the remaining chapters of this thesis. These methods are described in detail in the corresponding studies (Chapter 2 – Chapter 7). A brief overview of the used research methods is provided in the following subsections.

1.5.1 **Systematic literature review**

A Systematic Literature Review (SLR) attempts to identify, evaluate and interpret all available evidence on a research topic in a systematic way. The aim of an SLR is to provide an evaluation of the available literature on a research topic in a rigorous, unbiased and auditable way [15]. The SLR consists of the following three phases [15]:

1. **Planning**: The first important task is to establish the need for performing a systematic review. Next, in the planning phase, a review protocol is prepared detailing the research questions and procedures for search, selection, quality assessment and synthesis of primary studies.
Table 1.2: Research methods and thesis chapters

<table>
<thead>
<tr>
<th>Research method</th>
<th>Chapters</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2</td>
</tr>
<tr>
<td>Systematic literature review</td>
<td>✓</td>
</tr>
<tr>
<td>Systematic mapping study</td>
<td></td>
</tr>
<tr>
<td>Industrial survey (Questionnaire)</td>
<td>✓</td>
</tr>
<tr>
<td>Industrial survey (Interviews)</td>
<td></td>
</tr>
<tr>
<td>Case study</td>
<td></td>
</tr>
<tr>
<td>Industry involvement</td>
<td></td>
</tr>
</tbody>
</table>

2. **Execution**: This phase follows the procedures laid down in the review protocol. After the search process is complete, study selection procedures are applied to select primary studies. The quality assessment of these selected studies is performed, and data is extracted. The extracted data is then synthesized to answer the research questions.

3. **Reporting**: The findings of the review are documented and published in this phase.

Chapter 2 (Study 1) reports an SLR conducted to identify and analyze empirical evidence on effort estimation in ASD. The SLR was selected as research method since our aim was to identify, aggregate and analyze existing empirical works published in peer reviewed venues. The SLR followed the guidelines by Kitchenham and Charters [15].

1.5.2 **Systematic mapping study**

Systematic Mapping Study provides a broad overview of a research topic, and identifies the available evidence on a research topic [15]. The findings of a mapping study can be used to initiate other studies (e.g., SLR) to investigate the topic further at a deeper level. A mapping study, like a systematic literature review, also includes planning, execution and reporting phases. However, these phases are performed at a broader level in a mapping study. The addressed research questions are also much broader as compared to the questions of an SLR. Mapping studies do not include detailed syntheses and quality assessments of the included primary studies to critically appraise the identified evidence [30].

Chapter 4 (Study 3) reports a systematic mapping study conducted to identify the state of the art on taxonomies in software engineering. The mapping study method was
Chapter 1. Introduction

selected because the topic was very broad, and our goal was to provide an overview on the status of taxonomies in the whole software engineering discipline. The mapping study followed the guidelines proposed by Petersen et al. [16].

1.5.3 Industrial survey

A survey is described as “a system for collecting information from or about people to describe, compare or explain their knowledge, attitudes and behavior” [17]. Surveys are performed in retrospect as an investigation about an intervention that has already been in practice for some time [18], and researchers are interested in collecting the information from a relatively large number of individuals related to the intervention under investigation [19, 20].

Chapter 3 (Study 2) reports an industrial survey that was conducted to identify the state of the practice on effort estimation in ASD. This method was selected since our goal was to identify information from a large and heterogeneous sample of agile practitioners.

Surveys consist of mainly seven steps [17]:

1. Defining the survey objectives
2. Study design
3. Survey instrument design
4. Data collection
5. Data analysis
6. Results reporting

There are different approaches for data collection in a survey, such as questionnaires, interviews, structured record interviews and structured observations [17]. In this thesis, we have used questionnaire (Chapter 3) and interview (Chapter 5) for data collection.

Data collection with questionnaire

The survey study reported in Chapter 3 employed an online questionnaire, hosted at surveymonkey.com, to collect data from survey respondents. Online web-based questionnaires offer a low cost and efficient mechanism to gather data from respondents [20, 21]. The software practitioners who have been involved in effort estimation in ASD represented the survey population. We applied a mix of convenience and
snowball sampling techniques to collect data from survey respondents. The survey participants were recruited initially through personal contacts. The survey link was also posted to the agile and measurement-related forums at LinkedIn. A snowball sampling approach [20] was also used via our initial industry contacts. Finally, the survey was also advertised by one of the authors at three conferences (PROMISE 2014, RE 2014, ICGSE 2014).

![Figure 1.1: Thesis overview: Chapters, research methods and questions](image)

**Data collection with interview**

The studies reported in Chapter 5 employed interviews to collect data from practitioners about effort estimation activities of their teams. Interview is an effective technique to collect rich qualitative data [20]. Four agile practitioners from three different companies were interviewed to collect data about estimation activity conducted in their teams. The coded data, in the form of excel sheets, was shared with the respective intervie-
Chapter 1. Introduction

Wees for validation. The data collected from the interviews was used to characterize the effort estimation activities of the practitioners’ teams using the proposed taxonomy.

Case study

Chapter 6 and 7 report case studies conducted in different companies to investigate effort estimation. Case study is defined by Yin [34] as an “empirical inquiry that investigates a contemporary phenomenon within its real-life context, especially when the boundaries between the phenomenon and its context are not clearly evident”. Our aim was to investigate effort estimation (phenomenon) in real context. We, therefore, selected case study method to perform this investigation, as it supports the investigation of a phenomenon in its natural context [34, 35].

We used the case study guidelines by Runeson et al. [35] to perform the case studies reported in Chapter 6 and 7. Case study process consists of the following five broad steps [35]:

1. Design case study: in this first step case study plan/protocol is prepared, which details elements such as purpose, case, unit of analysis, data collection and analysis methods.

2. Prepare for data collection: data collection plan is defined, which details the types of data sources and methods to be used.

3. Collect evidence: data collection methods are executed, as detailed in the data collection plan.

4. Analyze collected data: collected data is analyzed.

5. Report: The study findings and conclusions are reported in formats suitable for the intended audience.

Case study should include multiple sources of evidence for triangulation purposes [20, 35]. The case studies reported in Chapter 6 and 7 used multiple sources of evidence to collect and triangulate the data. We used a combination of qualitative (e.g., through interviews) and quantitative (e.g., archival data about estimates) data in these case studies.

Case studies can be of different types (e.g., exploratory, explanatory, improving) depending on their purpose, and may involve single or multiple cases [35]. Chapter 6 reports a single case study of exploratory type to investigate effort estimation in large-scale software development. The case is a large telecommunication software product being developed by Ericsson involving agile teams located in different countries.

5www.ericsson.com
Chapter 7 reports a multi-case study of the improving type [35] to develop, evolve and use checklists to improve effort estimation in the following three cases:

1. Infoway\textsuperscript{6} case: Infoway is a medium sized (over 40 employees) Brazilian software vendor in eHealth domain. The study investigated effort estimation process of two products of Infoway, involving two different agile teams.

2. Diyatech\textsuperscript{7} case: Diyatech is a medium sized (over 50 employees) software company based in Pakistan. They develop performance enhancement products. The studied product is designed to enhance performance and support extreme transaction processing in distributed environments.

3. TSoft case: TSoft\textsuperscript{8} is a small and recently established offshore development site of a parent company in Norway. The parent company belongs to the transportation industry. TSoft is developing a product for its parent company. Due to the confidentiality requirements of the company, we are not using the actual company name.

Further details of the above cases are described in Chapter 6 and 7.

1.6 Validity threats and limitations

We used multiple research methods in this thesis. Each research method involves different validity threats. The validity threats and the corresponding mitigating steps taken in each study are detailed in each chapter. Using the validity threats classification from Runeson et al. [35], here we provide a combined discussion on the validity threats to the thesis results, and the steps that were taken to mitigate them.

1.6.1 Construct validity

Construct validity relates to the extent to which the studied measures actually represent the constructs that the researcher intends to study [35]. In case of a survey reported in Chapter 3, the questionnaire instrument was developed iteratively using the feedback from the two researchers having experience in the relevant field. We conducted semi-structured interviews in the studies reported in Chapter 5, 6 and 7. In order to ensure that we and the interviewees understand the concepts discussed in these interviews in a

\textsuperscript{6}https://infoway-br.com
\textsuperscript{7}www.diyatech.com
\textsuperscript{8}Due to the confidentiality requirements of the company, we are not using their real name.
similar way, we performed multiple steps such as: involvement of multiple researchers during the preparation of the interview questions, concluding the interview with a summary of what is discussed in the interview, sharing of the results with the interviewees for further feedback.

The estimation checklists proposed in Chapter 7 included many factors that are to be used by study participants during estimation. To ensure that these are understood correctly by the participants, we presented these factors and their corresponding scale types during workshops. We encouraged participants to ask questions if they think something is not clear. We also provided written descriptions of what these checklist factors and how can they be used. Finally, the participants were requested to apply the checklist on a sample of previously completed tasks, and provide their feedback and ask questions if something is not clear.

1.6.2 Internal validity

Internal validity is related to the presence of confounding factors that the researcher is not aware of, and their potential impact on the investigated variables. Internal validity becomes relevant when casual relationships are investigated [35].

In case of SLR reported in Chapter 2, we took number of steps to limit the chances of missing relevant literature, such as following SLR guidelines [15] to construct search string, searching in multiple databases, using a quasi-gold standard [39] to validate the search string, involvement of multiple researchers in the study selection process. In case of mapping study reported in Chapter 4, we followed the mapping study guidelines [16], constructed the search string from the research questions and involved multiple researchers in the study selection and data extraction phases.

In survey-based investigations, it is essential to ensure that only the relevant persons respond the survey. In the survey-based study in Chapter 3, we took the following steps to ensure that only the relevant persons participate in the survey: clear statement in the survey introduction about who is the relevant person for this survey, their experience in agile software development, ensuring anonymity to the respondents, and matching the respondents’ provided country against the IP address collected by the survey tool.

We followed a process to systematically design the agile estimation taxonomy presented in Chapter 5. However, it is based on the data collected in two empirical studies: an SLR [36] and a survey [37]. It is possible that some factors were missed out due to the limited number of studies/respondents in the SLR/survey. We also compared the proposed taxonomy with a reference work [38] on effort estimation in general.

In all case studies, we interviewed practitioners in different roles, rather than relying on the opinion of a single person or role. The results of the analysis were shared with the study participants to ensure if we have not missed out some important as-
pect. We also involved multiple researchers during the planning, execution and analysis stages of the case studies. These steps reduced the chances of misinterpretation of the data, results, and feedback. In case of the multi-case study reported in Chapter 7, we had a relatively prolonged involvement [35] with the case companies. During this time, we had multiple feedback cycles with the study participants.

1.6.3 External validity

External validity relates to the extent to which the findings are generalizable [35].

For the case studies reported in Chapter 6 and 7, the results are not generalizable outside the context of the studied cases. We have attempted to describe the context of each studied case to facilitate the researchers and practitioners in deciding if the findings are relevant to their unique contexts. The results of these studies would be of interest to companies using expert judgment to estimate development effort and have teams that use agile practices such as sprint and release planning, time-boxed sprints, stand-ups, unit testing, backlog, etc.

The SLR and survey, reported in Chapter 2 and 3 respectively, aggregated the evidence on effort estimation in agile software development context only. Their findings are only applicable to the teams that estimate development effort in the context of agile software development. In Chapter 5 the proposed taxonomy was applied to characterize example instances of the effort estimation of five teams from three different companies. The results of the taxonomy application are not applicable outside the context of the included teams.

1.6.4 Reliability

Reliability is related to the degree to which the data and the analysis are dependent on the specific researchers [35]. The dependence of the data and analysis on particular researchers compromises the repeatability of the research.

We attempted to minimize its impact by involving multiple researchers during the planning, execution, analysis, and reporting of all studies included in this thesis. In the SLR, the first author developed and revised the protocol based on the feedback from the other authors. Likewise, multiple researchers were involved in the study selection process. One researcher mainly did the data extraction, which implies that there is some risk of bias and threat to the validity of the findings. However, this threat was reduced by using well defined data extraction forms, which were reviewed by multiple researchers as part of the SLR protocol. Further, for validation purpose each of the remaining authors performed the data extraction of a sample (10%) of primary studies.
Chapter 1. Introduction

The questionnaire instrument in the survey study in Chapter 3 was developed and revised by the first author based on the feedback of the multiple researchers having experience in the relevant knowledge area. Likewise, the study protocols for case studies reported in Chapter 6 and 7 were also developed and reviewed by multiple researchers. The data analysis was mainly lead by the first author, wherein the remaining authors reviewed the results and provided their feedback. Furthermore, both intermediate and final results were shared with the case participants for their input and further discussion and understanding.

1.7 Summary of Studies Included in the Thesis

This section presents the summaries and the contributions of the studies included in this thesis.

1.7.1 Study 1: Effort estimation in agile software development: A systematic literature review

This study aimed to provide a detailed overview of state of the art in the area of effort estimation in the agile context. Since the emergence of agile methodologies, many software companies have shifted to Agile Software Development (ASD), and since then many studies have been conducted to investigate effort estimation within such context. However, to date, there is no single study that presents a detailed overview of state of the art in effort estimation for ASD.

To report the state of the art, we conducted a systematic literature review following the guidelines proposed in the evidence-based software engineering literature [15].

A total of 25 primary studies were selected. The main findings of this study are: i) Subjective estimation techniques (e.g., expert judgment, planning poker, use case points estimation method) are the most frequently applied in an agile context; ii) Use case points and story points are the most commonly used size metrics respectively; iii) MMRE (Mean Magnitude of Relative Error) and MRE (Magnitude of Relative Error) are the most frequently used accuracy metrics; iv) Team skills, prior experience, and task size are cited as the three important cost drivers for effort estimation in ASD; and v) Extreme Programming (XP) and SCRUM are the only specifically named agile methods for which the effort estimation has reportedly been investigated during release and iteration planning stages.

Subjective estimation techniques, e.g., expert judgment-based techniques, planning poker or the use case points method, are the ones used most frequently in agile effort
estimation studies. In ASD size and team related cost drivers are the most commonly used effort predictors.

<table>
<thead>
<tr>
<th>Contribution:</th>
</tr>
</thead>
<tbody>
<tr>
<td>This study aggregated the reported evidence on effort estimation in ASD context. It highlighted frequently used techniques and their accuracy, cost drivers and size measures used to estimate development effort in an agile context. The evidence identified in this study is used as input in Study 5 to develop a taxonomy of agile effort estimation.</td>
</tr>
</tbody>
</table>

1.7.2 Study 2: Effort estimation in agile software development: A survey on the state of the practice

Study 2 was conducted to complement the findings of study 1. Study 1 identified several empirical studies on effort estimation in Agile Software Development (ASD) to report state of the art. However, to date there are no studies on the state of the practice in this area, focusing on similar issues to those investigated in study 1.

A survey was carried out using as instrument an online questionnaire answered by agile practitioners who have experience in effort estimation. Data were collected from 60 agile practitioners from 16 different countries. The main findings are: 1) Planning poker (63%), analogy (47%) and expert judgment (38%) are frequently practiced estimation techniques in ASD; 2) Story points is the most frequently (62%) employed size metric, used solo or in combination with other metrics (e.g., function points); 3) Team’s expertise level and prior experience are the most commonly used cost drivers; 4) 42%/10% of the respondents believe that their effort estimates on average are under/over estimated by an error of 25% or more; 5) Most agile teams take into account implementation and testing activities during effort estimation; 6) Estimation is mostly performed at sprint and release planning levels in ASD.

Effort estimation techniques that rely on experts’ subjective assessment are most frequently used in ASD, with effort underestimation being the dominant trend. Further, the use of multiple techniques in combination and story points seem to present a positive association with estimation accuracy, and team-related cost drivers, such as team members’ expertise, are the ones used by most agile teams. Finally, requirements and management related issues are perceived as the main reasons for inaccurate estimates.
Chapter 1. Introduction

**Contributions:**
This study elicited the state of the practice on effort estimation in ASD context. It captured the frequently used techniques, their accuracy level and the effort predictors that practitioners use to estimate development effort in the agile context. This study also elicited and organized the main reasons behind the inaccurate effort estimates. The evidence identified in this study is used as input in Study 5 to develop a taxonomy of agile effort estimation.

1.7.3 **Study 3: Taxonomies in software engineering: A systematic mapping study and a revised taxonomy development method**

The knowledge on effort estimation in ASD has not been organized. Our initial search on how knowledge is organized in other software engineering areas identified that taxonomies have frequently been used in software engineering to organize and classify knowledge. Study 3 was initiated to characterize the state of the art research on SE taxonomies, and to identify good practices related to the design and evolution of SE taxonomies. A systematic mapping study was conducted, based on 270 primary studies.

It was found that an increasing number of SE taxonomies were published since 2000 in a broad range of venues, including leading software engineering journals and conferences. The majority of taxonomies are designed in software construction, design, requirements, and maintenance SWEBOK knowledge areas [29]. The majority of the taxonomies in software engineering are not validated using a thorough approach (e.g., case study or experiment).

The results of Study 3 showed that taxonomies in SE have mostly been designed in an ad-hoc manner. The choices regarding the employed classification structures, procedures, and descriptive bases are usually not well described and motivated. Based on our results, we have revised an existing method for designing taxonomies more systematically.

**Contribution:**
The results of the system mapping study showed that the most taxonomies in software engineering are designed in an ad-hoc manner. This study proposed a revision in an existing method to design software engineering taxonomies more systematically. Later in Study 4, the method was used to develop a taxonomy of effort estimation for agile software development context.
**1.7.4 Study 4: Agile estimation taxonomy**

The aim of this study is twofold: 1) To organize the knowledge on effort estimation in the agile context, identified in Study 1 and 2. and 2) To use this organized knowledge to support practice and future research on effort estimation in the agile context. We applied the taxonomy design method proposed in Study 3 to organize the identified knowledge as a taxonomy of effort estimation in the agile context. The proposed taxonomy offers a faceted classification scheme to characterize estimation activities of agile projects. Our agile estimation taxonomy consists of four dimensions: Estimation context, estimation technique, effort predictors and effort estimate. Each dimension, in turn, has several facets.

We applied the taxonomy to characterize estimation activities of 10 agile projects identified from the literature to assess whether all vital estimation related aspects are reported. The results showed that studies do not report complete information related to estimation. The taxonomy was also used to characterize the estimation activities of four agile teams from three different software companies. The practitioners involved in the investigation found the taxonomy useful in characterizing and documenting estimation sessions.

**Contributions:**

This study organized the knowledge on effort estimation in the agile context as a taxonomy. The proposed taxonomy was used to characterize the effort estimation of selected cases from the literature and industry.

The taxonomy is used as one of the bases for the checklist work presented in Study 6.

**1.7.5 Study 5: Effort estimation in large-scale software development: an industrial case study**

Software projects frequently incur schedule and budget overruns. Planning and estimation are particularly challenging in large and globally distributed projects. While software engineering researchers have been investigating effort estimation for many years to help practitioners to improve their estimation processes, there is little research about effort estimation in large-scale distributed agile projects. In this study, we investigated factors that impact the accuracy of the effort estimation in a large scale distributed project.

The main objective of this study is three-fold: i) to identify how effort estimation is carried out in large-scale distributed agile projects; ii) to analyze the accuracy of the effort estimation processes in large-scale distributed agile projects; and iii) to identify the factors that impact the accuracy of effort estimates in large-scale distributed agile
projects. We performed an exploratory longitudinal case study. The data collection was operationalized through archival research and semi-structured interviews.

The main findings of this study are: 1) underestimation is the dominant trend in the studied case, 2) re-estimation at the analysis stage improves the accuracy of the effort estimates, 3) requirements with large size/scope incur larger effort overruns, 4) immature teams incur larger effort overruns, 5) requirements developed in multi-site settings incur larger effort overruns as compared to requirements developed in a collocated setting, and 6) requirements priorities impact the accuracy of the effort estimates.

Effort estimation is carried out at quotation and analysis stages in the studied case. It is a challenging task involving coordination amongst many different stakeholders. Furthermore, lack of details and changes in requirements, the immaturity of the newly on-boarded teams and the challenges associated with the large-scale add complexities in the effort estimation process.

**Contribution:**
This study investigated how project size, team maturity, customer priority and involvement of multi-site development impact the accuracy of the effort estimates in a large-scale software project involving agile teams at different levels of maturity.

### 1.7.6 Study 6: Developing and Using checklists to improve software effort estimation: a multi-case study

Expert judgment based techniques are widely used in the software industry to estimated software development effort. Expert estimates can be inconsistent, as in the absence of any process or instrument support, experts forget essential activities and tasks that should be considered during estimation. This scenario leads to underestimation (i.e., effort overruns), which is a common occurrence in software projects.

This study aims to improve the expert estimation of software development effort. To do so, our goal is two-fold: 1) to propose a process to develop and evolve estimation checklists for software teams and companies, 2) to determine the usefulness of the checklist in improving expert estimation process and accuracy of the effort estimates.

We applied the case study methodology to investigate the stated objectives in four agile teams from three different software companies. The proposed process was successfully implemented in the three case companies to design and evolve estimation checklists. Based on multiple feedback cycles during the process, we customized the checklists for each involved team. The use of checklist significantly improved the accuracy of the effort estimates in two case companies. In particular, the underestimation bias was reduced considerably. For the third case, due to the unavailability of the previ-
ous data, we could not perform any comparison. However, the new effort data when the checklist was used, show that the estimates are quite accurate. The study participants from all cases observed several benefits of using the checklist during estimation, such as increased confidence in estimates, improved consistency due to help in recalling the relevant activities and factors, some objectivity in the process, improved understanding of tasks being estimated due to good impact analysis, and reduced chances of missing tasks.

The use of a checklist in the expert estimation process adds the much-needed structure and some objectivity in the otherwise mostly ad-hoc and inconsistent process. The consideration of the checklist factors was found to be useful in reducing the underestimation bias. The checklist does not add much overhead, and the agile teams at two out of three case companies integrated developed checklists with their estimation processes. In case of the third company, the team did not incorporate the developed checklist in their estimation process. However, the project manager continues to use checklist factors in prioritizing product backlog.

**Contributions:**

This study proposed a method to develop and evolve checklist to improve expert judgment based software effort estimation process. The proposed method was applied in three software companies, and the usefulness of the proposed checklists in improving the estimation process and the estimates’ accuracy was demonstrated.

## 1.8 Discussion

This thesis aimed to improve effort estimation process in agile context. This thesis attempted to achieve this aim in three stages: 1) First, by identifying how effort estimation is practiced in agile context, 2) Second, by organizing the identified evidence as a taxonomy of effort estimation in agile context, providing a mechanism to researchers and practitioners to effectively characterize their effort estimation process, 3) Third, based on the taxonomy, proposed a checklist to improve expert judgment based effort estimation in agile projects, and also provided a process support for developing and evolving such checklists.

We discuss these stages further in the following subsections:

### 1.8.1 RQ1: How is effort estimated in ASD context?

This thesis aggregated the reported evidence on effort estimation in ASD context with an SLR (Chapter 2). The SLR identified that the estimation techniques that rely on the
subjective judgment of experts are used the most in ASD context. These techniques include planning poker, use case points estimation methods, estimation by analogy etc. Furthermore, task size and team related cost drivers, such as team’s prior experience and skill level, are the most frequently used effort predictors.

To complement the findings of SLR, the thesis attempted to elicit the state of the practice on effort estimation in ASD through an industrial survey (Chapter 3). The survey included the responses of 60 agile practitioners from 16 different countries. The survey identified that most agile practitioners use techniques that rely on the subjective assessment of experts to arrive at effort estimates. Agile practitioners cited that they use point-based size measures, such as story points and use case points, as size measures. As for the cost drivers, surveyed agile practitioners cited they consider team related cost drivers (such as team’s skill level, their prior experience), non-functional requirements, customer collaboration and task size during effort estimation. The survey also identified that issues related to requirements, project management, and the team are the main reasons for inaccurate effort estimates.

To account for the limited research on effort estimation in large-scale projects involving agile teams, this thesis attempted to understand how effort estimation is practiced in a large project, and what factors impact the accuracy of the effort estimates in this context. An industrial case study at Ericsson was performed involving the development of a very large telecommunication product that has been in development for the last 15 years or so. The case project involved 24 agile teams located in four different countries (Sweden, India, Italy, and USA). The studied explored the impact of these four factors on the accuracy of the effort estimates: size, team maturity, customer priority and multi-site development. The study found that the immature teams are more likely to incur larger magnitude of effort overruns. Further, the requirements having a large size, low priority or multi-site development setting are related to the larger magnitude of effort overruns. The study also found that the re-estimation of the initial estimates at the quotation stage improves the accuracy of the effort estimates.

1.8.2 RQ2: How to organize knowledge on effort estimation in ASD context?

To organize the identified knowledge as part of RQ1, this thesis proposed a method to design taxonomies in software engineering. This thesis organized the evidence, identified in studies reported in Chapter 2 and 3, as a taxonomy of effort estimation in the agile context using the proposed taxonomy design method. The proposed taxonomy can be used by both researchers and practitioners, in characterizing their effort estimation process. The applicability of the proposed taxonomy was demonstrated on projects.
from literature and also from the real industrial cases. The industry participants found
taxonomy useful in documenting their effort estimation experience. However, the tax-
onomy was not used as an instrument during effort estimation to facilitate experts in
consistently considering the same set of factors and activities to arrive at better effort
estimates.

1.8.3 RQ3: How to improve expert judgment based effort estima-
tion process in ASD context?

As stated above the taxonomy was used only to characterize the effort estimation, as
it provided a mechanism to document the context and the factors used during effort
estimation.

This thesis contributed to the improvement of the expert estimation of software
development effort in agile contexts by developing estimation checklists for three dif-
ferent software companies. Chapter 7 reports this multi-case study for development
and evolution of estimation checklist based on our previous works as part of RQ2 and
RQ3. The study also proposed a process support for developing estimation checklists
in other agile contexts as well.

First case company is Infoway, a Brazilian medium-sized software company in
eHealth domain. Two Infoway teams, working on two different products in the same
domain, were studied. Second case company is Diyatech, a Pakistan medium-sized
software company developing products in the performance enhancement related do-
main. One Diyatech team, working on a performance enhancement product to support
distributed computing and caching, was studied. Third case company is TSoft, which
is a new and small offshore development center in Pakistan of a parent company in
Norway. TSoft is developing a product in transportation domain for its parent com-
pany.

This thesis successfully demonstrated the application of a checklist development
process in all three cases. The case participants from all three companies used and ap-
preciated the estimation checklists. They experienced several benefits of using check-
lists to estimate development effort: increased confidence in effort estimates, better task
understandability, better support for new participants in the estimation process and im-
provement in the consistent consideration of relevant factors during effort estimation.
Comparison of previous and new effort estimates showed improvement in accuracy of
the effort estimates in two cases. Particularly, the results showed that the checklists
were useful in reduced the underestimation bias, which is a frequent phenomenon in
software projects.

27
1.9 Conclusions and Future work

This thesis has investigated effort estimation in ASD context. It has identified from literature and practice (Chapter 2 and 3), how, and in what context, the effort is estimated in ASD. To organize the identified evidence on effort estimation in an agile context, it has proposed a method to design taxonomies systematically. The applicability of the taxonomy design method was demonstrated by designing a taxonomy for effort estimation in ASD context. The proposed taxonomy was used to characterize the effort estimation of the selected cases from literature as well as industry. The involved industrial partners appreciated the use of taxonomy in characterizing their effort estimation activity. Using this taxonomy as basis, we plan to propose more detailed guidelines for reporting empirical studies on software effort estimation. The aim is to improve the reporting quality of estimation studies to allow for more rich analysis and synthesis.

Agile teams mostly use expert judgment based effort estimation process to estimate developed effort. This thesis proposed a process to develop and evolve estimation checklists to improve expert judgment based effort estimation process. The proposed method was applied in three different case companies. The use of checklists in the three case companies resulted in several benefits such as improvement in accuracy, increased confidence in effort estimates, better task/requirement understanding, better support for new team members, consistent consideration of factors during effort estimation, and reduced chances of missing tasks. In two out of three companies, development teams continue to use the checklists during effort estimation. In case of the third company, the checklist factors have been used by the project manager, instead of the development team, to prioritize the backlog.

As part of our future, we plan to collect more data and feedback from the three case companies. Our goal is to use the feedback to improve the checklists further and use the collected data to further empirically validate the benefits of using estimation checklists. Furthermore, we plan to work with more companies to help them develop and use customized estimation checklists to improve their estimation processes. We have recently presented our idea to a large Swedish software company in Telecommunication domain, who have agreed to work with us on this plan.
1.10 References


Chapter 1. Introduction


Chapter 1. Introduction


Chapter 1. Introduction


Chapter 2

Effort Estimation in Agile Software Development: A Systematic Literature Review

2.1 Introduction

Effort estimation is an integral part of software project management. There is a plethora of research in effort estimation in the form of models, techniques, methods, and tools. Different reviews of such literature are detailed in [1–7]. Effort estimation, like other software engineering activities, is performed under the umbrella of a development process. We have different types of development processes, e.g. traditional, agile etc., each having a different perspective on planning and estimation. During the past 10 years or so, there has been lot of interest and acceptability of agile processes, leading to the formal use of the term Agile Software Development (ASD). Under an ASD process, software is developed incrementally in small iterations and customer feedback serves as an important input for subsequent iterations. This also implies that estimations and plans need to be done progressively. This is achieved in ASD by planning iteratively at three levels i.e. release planning, iteration planning and the current day planning [8]. Given the high value of iterations in ASD, Planning and estimation are performed differently from traditional software development [8]. Some of the common techniques for estimation in ASD are Expert Opinion, Analogy and Disaggregation [8]. Planning Poker, introduced by Grenning in [9], is another technique often used for estimation
Planning poker has been used in a few agile effort estimation studies. Haugen [10] investigated its usefulness to estimate effort for an XP team during release planning. Results indicated planning poker produced more accurate estimates than unstructured group estimates when the team had related experience with similar projects. Two other studies [11, 12], conducted in a Scrum environment, compared group consensus estimates calculated by planning poker and statistical combination of individual expert estimates. Results showed that group consensus estimates were less optimistic and more accurate than statistical combination of individual estimates. Results of another study [13] also conducted in a Scrum environment do not fully support the results reported in [11, 12]. However, it was observed in [13] that optimism bias and estimation accuracy can be improved when planning poker is used by experienced practitioners.

Besides planning poker, other estimation techniques have also been investigated for ASD. A detailed incremental prediction model is proposed in [14] for effort prediction in agile and iterative development environments. This model was compared with a traditional global prediction model in an Extreme Programming (XP) environment, and results showed that the new incremental model outperformed the traditional estimation model in the early phases of development. Parvez [15] modified the use case point estimation method by inserting a new layer consisting of efficiency and risk factors, in order to estimate testing effort in of an agile project. The modified method was applied on four real projects to estimate the testing effort and presented more accurate estimates than the original use case method. A Constructive Agile Estimation Algorithm (CAEA) is proposed in [16] to incorporate factors believed vital to accurately investigate the cost, size and duration of a project.

The aforementioned studies are a sample of the range of studies that have been published on effort estimation in ASD. However, despite the current number of studies in such context, their evidence has not yet been aggregated and synthesized in a single place. Estimation reviews, such as [1–7] did not review the literature from an ASD perspective. In addition, although there are a number of reviews that provide details on agile studies [17–21], these reviews do not aggregate the evidence in the literature from an effort estimation perspective. To the best of our knowledge there is no systematic review, which reviews the literature from both perspectives together, i.e. effort estimation and agile software development. Therefore the goal and main contribution of this paper is to report the state of the art in the field of effort estimation in ASD by means of a Systematic Literature Review (SLR). We followed the guidelines proposed by Kitchenham and Charters [22]. This chapter details our SLR and also points out the gaps and future directions for research in effort estimation for ASD. Note that given
that we already knew about several studies in effort estimation in ASD and also that the evidence we wanted to gather from each study was quite detailed, we chose to carry out a SLR rather than a mapping study.

The remainder of the chapter is organized as follows: Section 2.2 describes the steps performed in this SLR; the SLR results are presented in Section 2.3 and discussed in Section 2.4. Finally, conclusions and directions for the future research are given in Section 2.5.

2.2 Systematic Literature Review

A systematic literature review is conducted methodically by following a set of guidelines to collect and analyze all available evidence about a specific question in an unbiased and repeatable manner [22]. We started the process by defining the protocol that was designed and pilot tested by the first author (student) and reviewed by the second author (supervisor). We now describe steps performed during this SLR.

2.2.1 Research Questions

Following research questions were framed to guide the SLR:

- **RQ1**: What techniques have been used for effort or size estimation in agile software development?
  - **RQ1a**: What metrics have been used to measure estimation accuracy of these techniques?
  - **RQ1b**: What accuracy level has been achieved by these techniques?

- **RQ2**: What effort predictors (size metrics, cost drivers) have been used in studies on effort estimation for agile software development?

- **RQ3**: What are the characteristics of the dataset/knowledge used in studies on size or effort estimation for agile software development?

- **RQ4**: Which agile methods have been investigated in studies on size or effort estimation?
  - **RQ4a**: Which development activities (e.g. coding, design) have been investigated?
  - **RQ4b**: Which planning levels (release, iteration, current day) have been investigated?

Based on the research questions, PICOC [22] is described below:
Chapter 2. Effort Estimation in Agile Software Development: A Systematic Literature Review

- Population (P): Software projects and applications developed using any of the agile software development methods
- Comparison (C): No comparison intervention
- Outcomes (O): Accuracy of estimation techniques for ASD.
- Context (C): ASD.

2.2.2 Search Strategy

A search strategy starts with the identification of major key terms from PICOC and their alternatives and synonyms. These terms are used to form a query string that is used to derive the rest of the search process.

Query String

The formation of a query string is an iterative process. Initially we followed the SLR guidelines [22] to create an initial string using Boolean OR/AND operators (OR is used to incorporate all synonyms and alternate spellings of each term and then these terms are ANDed to form one string). We piloted the initial string on search engines such as Scopus, IEEE explore, and Science Direct to check whether the string retrieved relevant primary studies and the studies we already knew about. The keywords from the known primary studies and newly fetched ones were also included if not already part of the string. We also studied the titles, abstracts and author keywords from some already known primary studies to identify search terms. Table 2.1 lists the keywords used by some primary studies on the effort estimation in ASD.

The term “agile software development” has a large number of synonyms and alternate terms that are used in literature; some are listed in table 1 above. Given that as we added more studies to our set of known primary studies more alternate terms for agile software development were found, we decided to simply use one word (i.e. “agile”) to cater for all of its possible alternate terms and ANDed it with the word “software” to filter out completely irrelevant studies from other domains. Dybå and Dingsøyr in their SLR [17] on agile software development have also used a similar approach for the term “agile”. Another SLR on usability in agile software development by Silva et al. [21] also uses the term “agile” in the search string instead of trying to add all of its alternatives. In addition, the set of known primary studies was also used as a quasi-gold standard as suggested in [44] to assess the accuracy of the search string. The final
search string we used is presented below. Note that this string had to be customized accordingly for each of the databases we used.

(Agile OR "extreme programming" OR Scrum OR “feature driven development” OR “dynamic systems development method” OR “crystal software development” OR “crystal methodology” OR “adaptive software development” OR “lean software development” AND (estimat* OR predict* OR forecast* OR calculat* OR assessment OR measur* OR sizing) AND (effort OR resource OR cost OR size OR metric OR user story OR velocity) AND (software)

Table 2.1: Keywords from known primary studies

<table>
<thead>
<tr>
<th>Keywords</th>
<th>References</th>
</tr>
</thead>
<tbody>
<tr>
<td>Estimat*(Estimation, Estimating, Estimate)</td>
<td>[10, 11, 13, 14, 23–37]</td>
</tr>
<tr>
<td>Predict* (Prediction, Predicting)</td>
<td>[14, 38–40]</td>
</tr>
<tr>
<td>Forecasting</td>
<td>[13]</td>
</tr>
<tr>
<td>Effort</td>
<td>[13, 14, 23, 25, 28, 29, 33, 35, 36, 38–40]</td>
</tr>
<tr>
<td>Cost</td>
<td>[23, 31, 33–35]</td>
</tr>
<tr>
<td>Size, Sizing</td>
<td>[16, 29, 30, 33, 41, 42]</td>
</tr>
<tr>
<td>Velocity</td>
<td>[16, 26, 31, 38]</td>
</tr>
<tr>
<td>User Story</td>
<td>[10, 13, 27, 29, 31, 37, 40, 42]</td>
</tr>
<tr>
<td>Metrics</td>
<td>[26, 32, 33, 39, 41, 43]</td>
</tr>
<tr>
<td>Measur* (measure, measurement, measuring)</td>
<td>[11, 28, 30, 32, 33, 39, 41, 43]</td>
</tr>
<tr>
<td>Agile Software Development</td>
<td>[13, 25, 29, 31, 32, 36, 39, 43]</td>
</tr>
<tr>
<td>Agile Methods</td>
<td>[16, 30, 40]</td>
</tr>
<tr>
<td>Agile Processes</td>
<td>[41]</td>
</tr>
<tr>
<td>Agile Development</td>
<td>[31, 34, 37, 41]</td>
</tr>
<tr>
<td>Agile Estimation, Agile Estimating</td>
<td>[16, 25, 28, 42]</td>
</tr>
<tr>
<td>Extreme Programming</td>
<td>[10, 14, 33, 35, 38]</td>
</tr>
<tr>
<td>Scrum</td>
<td>[13, 23, 26, 28, 34]</td>
</tr>
<tr>
<td>Agile Project Management</td>
<td>[27, 29]</td>
</tr>
<tr>
<td>Agile Projects</td>
<td>[27, 30]</td>
</tr>
<tr>
<td>Agile Teams</td>
<td>[32]</td>
</tr>
<tr>
<td>Agile Methodology</td>
<td>[33, 37]</td>
</tr>
<tr>
<td>Agile Planning</td>
<td>[37]</td>
</tr>
</tbody>
</table>
Chapter 2. Effort Estimation in Agile Software Development: A Systematic Literature Review

Primary and Secondary Searches

As part of the primary search strategy, search strings were applied on different databases in the first week of December 2013 to fetch the primary studies since 2001 (we chose 2001 as the starting date because this is when the Agile Manifesto\(^1\) was published). Databases, search results, before and after removing duplicates, are listed in Table 2.2. A master library was formed in Endnote\(^2\) wherein results from all databases were merged and duplicates were removed. In the end, after removal of duplicates, we ended up with 443 candidate primary studies.

<table>
<thead>
<tr>
<th>Database</th>
<th>Before removing duplicates</th>
<th>After removing duplicates</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scopus</td>
<td>302</td>
<td>177</td>
</tr>
<tr>
<td>IEEE Explore</td>
<td>181</td>
<td>154</td>
</tr>
<tr>
<td>EI Compendex</td>
<td>76</td>
<td>69</td>
</tr>
<tr>
<td>Web of Science</td>
<td>134</td>
<td>23</td>
</tr>
<tr>
<td>INSPEC</td>
<td>48</td>
<td>9</td>
</tr>
<tr>
<td>Science Direct</td>
<td>7</td>
<td>2</td>
</tr>
<tr>
<td>ACM DL</td>
<td>27</td>
<td>9</td>
</tr>
<tr>
<td>Springer Link</td>
<td>12</td>
<td>0</td>
</tr>
<tr>
<td>Total</td>
<td>787</td>
<td>443</td>
</tr>
</tbody>
</table>

The databases we used cover all major software engineering conferences and journals thus providing a comprehensive coverage of this SLR’s topic. We also ensured that these databases would index all the venues where known primary studies in our topic had been published. Other SLRs, such as [6, 19, 45], have also used these databases for searching relevant primary studies.

The secondary search was performed in the next phase by going through references of all the primary studies retrieved in the first phase.

2.2.3 Study Selection Criteria

The inclusion and exclusion criteria were defined in the light of the SLR objectives and the research questions.

---

\(^1\)http://agilemanifesto.org/
\(^2\)http://endnote.com/
Inclusion Criteria

It was decided that studies that:

- Report effort or size estimation related (technique or model or metric or measurement or predictors) AND
- Are based on any of the agile software development methods AND
- Described in English AND
- Are reported in peer reviewed workshop or conference or journal AND
- Are evidence-based (empirical studies)

will be included as primary studies.

Exclusion Criteria

- Are not about effort or size estimation OR
- That are not conducted using any of the agile software development methods OR
- Are not described in English OR
- Are not published in a peer reviewed conference or journal or workshop OR
- Are not empirically-based OR
- Deal with software maintenance phase only OR
- Deal with performance measurement only i.e. velocity measurement

will be excluded.

2.2.4 Study Selection Process

The study selection process was performed in two phases, as follows:

- Title and Abstract level screening: In the first phase the inclusion/exclusion criteria was applied to the title and abstracts of all candidate primary studies identified by applying search strategy. Three researchers (the first three authors) performed this step independently, on all 443-search results, to deal with the problem of researcher bias. To arrive at a consensus, two Skype meetings were arranged
between these three researchers that resulted in 30 papers passing the inclusion criteria out of 443 search results, with another 13 marked as doubtful. It was decided that the first author would go through the introduction, background and conclusion sections of these 13 doubtful papers to resolve the doubts. After consultation between the first two authors, two out of these 13 doubtful papers eventually passed the inclusion criteria increasing the number of studies to 32 that would be screened in the next phase.

• Full text level screening: In the second phase, the inclusion and exclusion criteria were applied on full text of 32 papers that passed phase 1. The first author performed this task for all 32 papers; the other authors performed the task on 10% (3 papers by each one) of the total number of papers. At the end results were compared and consensus was reached for all common papers via Skype and face-to-face meetings. Whenever we did not have access to the primary study due to database access restrictions, we emailed that paper’s authors. However, two papers could not be obtained by any means.

2.2.5 Quality Assessment

The study quality assessment checklist in Table 2.3 was customized based on the checklist suggestion provided in [22]. Other effort estimation SLRs [6, 7] have also customized their quality checklists based on the suggestions given in [22]. Each question in the QA checklist was answered using a three-point scale i.e. Yes (1 point), No (0 point) and Partial (0.5 point). Each study could obtain 0 to 13 points. We used the first quartile (13/4= 3.25) as the cutoff point for including a study, i.e., if a study scored less than 3.25 it would be removed from our final list of primary studies.

2.2.6 Data Extraction and Synthesis Strategies

A data extraction form was designed which, besides having general fields (year, conference, author etc.), contained fields corresponding to each research question (e.g. estimation technique used, size metrics, accuracy metrics, agile method etc.). Data extraction and quality assessment were both performed by the authors in accordance with the work division scheme described in Section 2.2.4. The extracted data were then synthesized to answer each of the research questions.
2.3 Results

In this section we describe the results for the overall SLR process and for each of the research questions as well. Table 2.4 describes the number of studies passing through different stages of the SLR. Details of the excluded papers during the full text screening and the QA phases are: eight papers [27, 38, 42, 46–50] were excluded due to not passing the inclusion criteria, two due to a low quality score [23, 51] and one [52] due to reporting a study already included in another paper [14]. Breakup of the eight papers excluded on inclusion and exclusion criteria is as under:

- 2 studies were not empirically based
- 3 studies were not conducted in an agile software development context
- 3 studies met the last exclusion criterion i.e. velocity measurement.

Table 2.3: Quality assessment checklist adopted from [7, 22]

<table>
<thead>
<tr>
<th>Questions</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Are the research aims clearly specified?</td>
<td>Y/N/P</td>
</tr>
<tr>
<td>2. Was the study designed to achieve these aims?</td>
<td>Y/N/P</td>
</tr>
<tr>
<td>3. Are the estimation techniques used clearly described and their selection justified?</td>
<td>Y/N/P</td>
</tr>
<tr>
<td>4. Are the variables considered by the study suitably measured?</td>
<td>Y/N/P</td>
</tr>
<tr>
<td>5. Are the data collection methods adequately described?</td>
<td>Y/N/P</td>
</tr>
<tr>
<td>6. Is the data collected adequately described?</td>
<td>Y/N/P</td>
</tr>
<tr>
<td>7. Is the purpose of the data analysis clear?</td>
<td>Y/N/P</td>
</tr>
<tr>
<td>8. Are statistical techniques used to analyze data adequately described and their use justified?</td>
<td>Y/N/P</td>
</tr>
<tr>
<td>9. Are negative results (if any) presented?</td>
<td>Y/N/P</td>
</tr>
<tr>
<td>10. Do the researchers discuss any problems with the validity/reliability of their results?</td>
<td>Y/N/P</td>
</tr>
<tr>
<td>11. Are all research questions answered adequately?</td>
<td>Y/N/P</td>
</tr>
<tr>
<td>12. How clear are the links between data, interpretation and conclusions?</td>
<td>Y/N/P</td>
</tr>
<tr>
<td>13. Are the findings based on multiple projects</td>
<td>Y/N/P</td>
</tr>
</tbody>
</table>

As part of the secondary search, references of 19 primary studies identified in the primary search process were screened to identify any additional relevant papers. Six
Chapter 2. Effort Estimation in Agile Software Development: A Systematic Literature Review

Table 2.4: No. of papers in study selection and QA

<table>
<thead>
<tr>
<th>Questions</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>a. Search results</td>
<td>443</td>
</tr>
<tr>
<td>b. After titles and abstracts screening</td>
<td>32</td>
</tr>
<tr>
<td>c. Inaccessible papers</td>
<td>2</td>
</tr>
<tr>
<td>d. Excluded on inclusion exclusion criteria</td>
<td>8</td>
</tr>
<tr>
<td>e. Duplicate study</td>
<td>1</td>
</tr>
<tr>
<td>f. Excluded on low quality score</td>
<td>2</td>
</tr>
<tr>
<td>g. Final Papers (b-c-d-e-f)</td>
<td>19</td>
</tr>
</tbody>
</table>

papers passed this initial screening in the secondary search, out of which four were excluded on inclusion and exclusion criteria applied on the full text, one for having a low quality score, leaving only one additional paper, which increased the total number of primary studies to 20. Some of the papers reported more than one study, as follows: one paper reported four studies; three papers reported two studies (one study is common in two of these three papers and is counted only once). Therefore, although we had 20 papers, they accounted for 25 primary studies. We present this SLR’s results arranged by the 25 primary studies reported in 20 primary papers.

2.3.1 RQ1: Estimation Techniques

Four studies, out of the 25 primary studies, investigated techniques for estimating size only (S1, S7, S11, S17); three did not describe the estimation technique used (S5, S14) while the remaining 19 studies used effort estimation techniques. Table 2.5 lists the different estimation techniques (both size and effort) along with their corresponding frequencies (no of studies using the technique) denoted as F in last column. Size estimation techniques are grouped with effort estimation techniques because within an agile context an effort estimate is often derived from a size estimate and velocity calculations [8]. Expert judgment, planning poker and use case points method (both original and modifications) are estimation techniques frequently used in an ASD context. All of these frequently used effort estimation techniques require a subjective assessment by the experts, which suggests that, in a similar way as currently occurs within non-agile software development contexts, within an agile context the estimation techniques that take into account experts’ subjective assessment are well received. Note that Table 2.5 also shows that other types of effort estimation techniques have also been investigated e.g. regression, neural networks. The frequency column (F) does not add up to 25 as many studies have used more than one technique.
Table 2.5: Estimation techniques investigated

<table>
<thead>
<tr>
<th>Technique</th>
<th>Study ID</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>Expert judgment</td>
<td>S2a, S2b, S4a, S4b, S4c, S4d, S6, S20</td>
<td>8</td>
</tr>
<tr>
<td>Planning poker</td>
<td>S1, S6, S8, S16a, S16b, S17</td>
<td>6</td>
</tr>
<tr>
<td>Use case points (UCP) method</td>
<td>S4a, S4b, S4c, S4d, S10, S12</td>
<td>6</td>
</tr>
<tr>
<td>UCP method modification</td>
<td>S4a, S4b, S4c, S4d, S11</td>
<td>5</td>
</tr>
<tr>
<td>Linear regression, robust regression, neural nets (RBF)</td>
<td>S2a, S2b, S3</td>
<td>3</td>
</tr>
<tr>
<td>Neural nets (SVM)</td>
<td>S2a, S2b</td>
<td>2</td>
</tr>
<tr>
<td>Constructive agile estimation algorithm</td>
<td>S13, S18</td>
<td>2</td>
</tr>
<tr>
<td>Statistical combination of individual estimates (S8), COSMIC FP (S7), Kalman Filter Algorithm (S9), Analogy (S15), Silent Grouping (S1), Simulation (S19)</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>Not described</td>
<td>S5, S14</td>
<td>2</td>
</tr>
</tbody>
</table>

F = Frequency

RQ1a: Accuracy Metrics

RQ1a investigates which prediction accuracy metrics were used by the primary studies. The entries shown in Table 2.6 are based solely on the 23 primary studies (see table 2.5) that used at least one estimation technique. Both the Mean Magnitude of Relative Error (MMRE) and the Magnitude of Relative Error (MRE) are the most frequently used accuracy measures. The use of MMRE is in line with the findings from other effort estimation SLRs [1, 7]. Two important observations were identified herein: the metric Prediction at n% (Pred(n)) was used in two studies only and secondly 26% (6 out of 23 studies included in Table 2.6) of the studies that employed some estimation technique have not measured their prediction accuracy. Finally, some studies have used other accuracy metrics, described in the category “other” in Table 2.6. Two of which (S5, S8) employed the Balanced Relative Error (BRE) metric on the grounds that BRE can more evenly balance overestimation and underestimation, when compared to MRE.

RQ1b: Accuracy Levels Achieved

This sub-question looks into the accuracy levels achieved by different techniques. Due to the space limitations, Table 2.7 only includes those techniques that have been applied
Table 2.6: Accuracy metrics used

<table>
<thead>
<tr>
<th>Metrics</th>
<th>Study ID</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>MMRE</td>
<td>S2a, S2b, S6, S9, S14</td>
<td>5</td>
</tr>
<tr>
<td>MRE</td>
<td>S3, S4a, S4b, S4c, S4d</td>
<td>5</td>
</tr>
<tr>
<td>Pred (25)</td>
<td>S2a, S2b</td>
<td>2</td>
</tr>
<tr>
<td>Not used</td>
<td>S1, S10, S13, S17, S18, S20</td>
<td>6</td>
</tr>
<tr>
<td>Other</td>
<td>S11(SD, Variance, F test), S12(comparison with actual), S16a and S16b (Box plots to compare effort estimates), S8(BRE, REbias), S5(BREbias), S6(MdMRE), S2a and S2b(MMER), S7 (Kappa index)</td>
<td>10</td>
</tr>
</tbody>
</table>

in more than one study and for which the accuracy level was reported. Other than Studies 4a to 4d that report good MRE values for UCP methods (modified and original) and expert judgment, no other technique has achieved accuracy level of at least 25% [53]. Studies 4a to 4d are all reported in a single paper and focus solely on testing effort, thus clearly prompting the need for further investigation. The lack of studies measuring prediction accuracy and presenting good accuracy represents a clear gap in this field.

Table 2.7: Accuracy achieved by frequently used techniques

<table>
<thead>
<tr>
<th>Technique</th>
<th>Accuracy achieved % (Study ID)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Planning poker</td>
<td>MMRE: 48.0 (S6), Mean BRE: 42.9-110.7 (S8)</td>
</tr>
<tr>
<td>UCP method</td>
<td>MRE: 10.0-21.0 (S4a to S4d)</td>
</tr>
<tr>
<td>UCP modified method</td>
<td>MRE: 2.0-11.0 (S4a to S4d)</td>
</tr>
<tr>
<td>Expert judgment</td>
<td>MRE: 8.00-32.00 (S4a to S4d), MMRE: 28-38 (S2a, S2b, S6), Pred (25%): 23.0-51.0 (S2a, S2b)</td>
</tr>
<tr>
<td>Linear regression</td>
<td>MMRE: 66.0-90.0 (S2a, S2b), Pred(25): 17.0-31.0 (S2a, S2b), MMER: 47.0-67.0 (S3), MRE: 11.0-57.0 (S3)</td>
</tr>
<tr>
<td>Neural nets (RBF)</td>
<td>MRE: 6.00-90.00 (S3)</td>
</tr>
</tbody>
</table>

2.3.2 RQ2: Effort Predictors

This question focuses on the effort predictors that have been used for effort estimation in an ASD context.
Table 2.8: Size metrics used

<table>
<thead>
<tr>
<th>Metrics</th>
<th>Study ID</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>Story points</td>
<td>S1, S2a, S8, S13, S17, S18</td>
<td>6</td>
</tr>
<tr>
<td>Use case points</td>
<td>S4a, S4b, S4c, S4d, S10, S11, S12</td>
<td>7</td>
</tr>
<tr>
<td>Function points</td>
<td>S9, S10, S17</td>
<td>3</td>
</tr>
<tr>
<td>Lines of code</td>
<td>S3</td>
<td>1</td>
</tr>
<tr>
<td>Other</td>
<td>Number of User Stories (S19), COSMIC FP (S7),</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>Length of User Story (S2b)</td>
<td></td>
</tr>
<tr>
<td>Not described</td>
<td>S5, S6, S15, S16a, S16b</td>
<td>5</td>
</tr>
</tbody>
</table>

Table 2.9: Cost drivers

<table>
<thead>
<tr>
<th>Cost drivers</th>
<th>Study ID</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dev. Team Skills/Expertise</td>
<td>S15, S16a, S16b</td>
<td>3</td>
</tr>
<tr>
<td>Task Size</td>
<td>S6, S11</td>
<td>2</td>
</tr>
<tr>
<td>Team’s Prior Experience</td>
<td>S6, S15</td>
<td>2</td>
</tr>
<tr>
<td>Test Efficiency Factor, Test Risk Factor</td>
<td>S4a, S4b, S4c, S4c</td>
<td>4</td>
</tr>
<tr>
<td>Project domain, Performance requirements,</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Configuration requirements, Volume of data</td>
<td></td>
<td></td>
</tr>
<tr>
<td>transactions, Complex processing, Operational</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ease for users, Multiple sites, Security</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>S19 (12 XP Practices), S2a</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&amp; S2b (Keywords in user story), S3 (Chidamber &amp;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Kemerer design metrics), S5</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(Customer Communication Frequency), S11</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(Use case Elements)</td>
<td></td>
</tr>
<tr>
<td>Other</td>
<td></td>
<td>6</td>
</tr>
<tr>
<td>Not investigated</td>
<td>S3, S7, S8, S9, S10, S12, S14, S17, S20</td>
<td>9</td>
</tr>
</tbody>
</table>
Chapter 2. Effort Estimation in Agile Software Development: A Systematic Literature Review

RQ2a: Size Metrics

Table 2.8 presents the size metrics that have been used in the primary studies focus of this SLR. Use case and story points are the most frequently used size metrics; very few studies used more traditional size metrics such as function points and lines of code, which in a way is not a surprise given our findings suggest that whenever the requirements are given in the form of stories or use case scenarios, story points and UC points are the choices commonly selected in combination with estimation techniques such as planning poker and UCP method. Finally, to our surprise, 20% of the studies (five) investigated herein did not describe any size metrics used during the study.

RQ2b: Cost Drivers

This sub-question investigates what cost drivers have been used by effort estimation studies in ASD context. Table 2.9 shows that different cost drivers have been used by the primary studies investigated herein, and without a common pattern. 24% of the studies (6 out of 25) used a cost driver that was not used by any other primary study. Two testing effort-related drivers are used in four studies (S4a to S4b), all reported in the same paper. Development team skills, size of the task to be estimated and team’s prior experience are reported in multiple studies. In the ASD context, where most of the knowledge is not explicit, skills and experience of the developers seems to play a crucial role in all tasks, including effort estimation. Note that 32% (9) of the studies detailed herein did not use any cost drivers for effort estimation, thus also suggesting a research gap and a possible avenue for future research.

2.3.3 RQ3: Characteristics of the Dataset or Knowledge Used

This question looks into the characteristics, i.e. domain (industrial or academic or both) and type (within-company or cross-company), of the datasets or knowledge used in primary studies. Table 2.10 shows that 76% (19) of the studies used an industrial dataset. This is a good sign as use of the industrial data increases the usefulness of such results for practitioners.

Table 2.11 details the type of dataset used herein; 72% (18) used within-company data, only 8% (2) used cross-company data, and one study (S9) used log-simulated data to predict completion time of project logs. Another 16% (4) did not describe the type of dataset used. We believe these results are quite interesting as they suggest that within the scope of ASD, companies have focused on their own project data, rather than looking for data from cross-company datasets. There are contexts (e.g. Web development) where estimation models based on within-company data have clearly presented
superior accuracy, when compared to models built from cross-company datasets [6]; however, the lack of primary studies herein using cross-company datasets does not necessarily mean that such datasets would not be useful within an ASD context; perhaps they have not been used because most of the publicly available datasets in software engineering contain only data on legacy systems. We believe this is also a clear research gap that provides further avenues for future research in effort estimation for ASD context.

Table 2.11: Type of datasets/knowledge used

<table>
<thead>
<tr>
<th>Type</th>
<th>Study ID</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>Within Company</td>
<td>S2a, S2b, S3, S4a-S4d, S5, S6, S10, S11, S12, S13, S15, S16, S17, S18, S20</td>
<td>18</td>
</tr>
<tr>
<td>Cross company</td>
<td>S1, S7</td>
<td>2</td>
</tr>
<tr>
<td>Not described</td>
<td>S8, S16a, S16b, S19</td>
<td>4</td>
</tr>
<tr>
<td>Log simulated</td>
<td>S9</td>
<td>1</td>
</tr>
</tbody>
</table>

2.3.4 RQ4: Agile Methods Used

This question is designed to identify the specific agile methods used in effort estimation studies in an ASD context. Table 2.12 details the methods used in each of the studies, and corresponding frequency of method usage. Scrum and XP were the most frequently used agile methods in studies on effort estimation. However it is unfortunate to note that 40% (10) of the studies only mention that they are using an agile method without specifying the exact agile method used.

3https://code.google.com/p/promisedata/

49
Chapter 2. Effort Estimation in Agile Software Development: A Systematic Literature Review

Table 2.12: Agile methods used

<table>
<thead>
<tr>
<th>Agile method</th>
<th>Study ID</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>XP</td>
<td>S2a, S2b, S3, S6, S14, S19, S20</td>
<td>7</td>
</tr>
<tr>
<td>SCRUM</td>
<td>S1, S4a-S4d, S8, S12, S17</td>
<td>8</td>
</tr>
<tr>
<td>Not described</td>
<td>S5, S7, S9, S10, S11, S13, S15, S16a, S16b, S18</td>
<td>10</td>
</tr>
</tbody>
</table>

RQ4a: Development Activity

The RQ4a investigates the development activities to which the effort estimate applies. The results are summarized in Table 2.13. None of the primary studies estimated effort for analysis and design activities only; 16% of the primary studies (four studies reported in one paper) investigated testing effort estimation, 12% (3) investigated implementation effort and another 12% (3) investigated both implementation and testing effort. Close to a third of the primary studies in this SLR (9) estimated effort considering all of the development activities (analysis, design, implementation and testing activities) of an agile project.

Table 2.13: Development activity investigated

<table>
<thead>
<tr>
<th>Dev. Activity</th>
<th>Study ID</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>Analysis</td>
<td>None</td>
<td>0</td>
</tr>
<tr>
<td>Design</td>
<td>None</td>
<td>0</td>
</tr>
<tr>
<td>Implementation (I)</td>
<td>S2a, S2b, S6</td>
<td>3</td>
</tr>
<tr>
<td>Testing (T)</td>
<td>S4a-S4d</td>
<td>4</td>
</tr>
<tr>
<td>Both I &amp; T</td>
<td>S8, S16a, S16b</td>
<td>3</td>
</tr>
<tr>
<td>All</td>
<td>S3, S5, S9, S10, S11, S12, S13, S18, S19</td>
<td>9</td>
</tr>
<tr>
<td>Not described</td>
<td>S1, S7, S14, S15, S17, S20</td>
<td>6</td>
</tr>
</tbody>
</table>

RQ4b: Planning Level

Planning can be performed at three different levels in an agile context i.e. release, iteration and daily planning [8]. This question looks into which planning levels were used in this SLR’s primary studies. Table 2.14 details the results showing the studies and their count (denoted as F in last column) against each planning level. A quarter of the studies (6) dealt with iteration level planning; and release planning is investigated in 20% (5) of the studies. Both iteration and release planning levels have received almost
equal coverage indicating the importance of planning at both levels; however, to our surprise, 48% (12) of the primary studies did not describe the planning level they were dealing with.

Table 2.14: Planning levels investigated

<table>
<thead>
<tr>
<th>Planning level</th>
<th>Study ID</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>Daily</td>
<td>None</td>
<td>0</td>
</tr>
<tr>
<td>Iteration (I)</td>
<td>S2a, S2b, S3, S12, S14, S17</td>
<td>6</td>
</tr>
<tr>
<td>Release (R)</td>
<td>S1, S6, S8, S9, S20</td>
<td>5</td>
</tr>
<tr>
<td>Testing (T)</td>
<td>S4a-S4d</td>
<td>4</td>
</tr>
<tr>
<td>Both I &amp; R</td>
<td>S13, S18</td>
<td>2</td>
</tr>
<tr>
<td>Not described</td>
<td>S4a-S4d, S5, S7, S10, S11, S15, S16a, S16b, S19</td>
<td>12</td>
</tr>
</tbody>
</table>

2.4 Discussion

The results of this SLR address four research questions (see Section 2.1), which cover the following aspects:

- Techniques for effort or size estimation in an Agile Software Development (ASD) context.
- Effort predictors used in estimation studies in ASD context.
- Characteristics of the datasets used.
- Agile methods, activities and planning levels investigated.

We observed that a variety of estimation techniques have been investigated since 2001, ranging from expert judgment to neural networks. These techniques used different accuracy metrics (e.g. MMRE; Kappa index) to assess prediction accuracy, which in most cases did not turn out to meet the 25% threshold [53]. In addition, with regard to the effort predictors used in the primary studies, we see that there is little consensus on the use of cost drivers, i.e. only a few cost drivers are common across multiple studies. We believe that further investigation relating to effort predictors is warranted, as it seems that the focus of previous studies mostly on size metrics may have had an effect on the poor prediction accuracy reported. When combining the results and discussion on two questions, it can be said that much needs to be done at both technique and cost drivers level to improve effort estimation in ASD. Currently there are no guidelines for
practitioners to decide as to which estimation technique or cost driver to choose in a particular agile method or context.

When it comes to the most frequently used estimation techniques, our results showed that the top four (expert judgment, planning poker, UCP method original and modified) have one common facet i.e. some form of subjective judgment by experts. Other techniques have also been investigated but not in this proportion. We can say that subjective estimation techniques are frequently used in agile projects.

With regard to prediction accuracy metrics, MMRE and MRE are the frequently used metrics; however two recent studies have recommended the use of BRE as accuracy metric on the basis that it balances over and under estimation better than MMRE. Some other alternate metrics have also been used e.g. boxplots (two studies) and MMER (two studies). Despite the existing criticism with regard to both MMRE and MRE for being inherently biased measures of accuracy (e.g. \cite{54, 55}), they were the ones used the most within the context of this SLR. In our view this is a concern and such issues should be considered in future studies. It is surprising that 26% (6) of the studies that have applied an estimation technique did not measure that technique’s prediction accuracy; in our view such studies may be of negligible use to researchers and practitioners as they failed to provide evidence on how suitable their effort estimation techniques are.

The SLR results showed that story points and use case points were the most frequently used size metrics while traditional metrics (FP or LOC) were rarely used in agile projects. Both story points and use case points seem to synchronize well with the prevalent methods for specifying the requirements under agile methods i.e. user stories and use case scenarios. There appears to be a relationship between the way the requirements are specified (user stories, use cases), frequently used size metrics (story points, UC points) and frequently used estimation techniques (Planning poker, use case points methods). What is lacking is a consensus on the set of cost drivers that should be used in a particular agile context. In the absence of correct and appropriate cost drivers, accuracy will be compromised and we believe that this may be the reason for poor accuracy levels identified in this SLR. We believe much needs to be done in the cost drivers area.

Industrial datasets are used in most of the studies, which may be a good indication for the acceptability of the results. However, most of the data sets, i.e. 72% (18), are of within company type. This may be due the fact that cross-company datasets specifically for agile projects are not easily available. With the increasing number of companies going agile, we believe that some efforts should be made to make cross-company datasets available for ASD context in order to support and improve effort estimation.

XP and Scrum were the two most frequently used agile methods, and we have
not found any estimation study that applied any other agile method other than these two. We do not know how estimation is performed in agile methods other than XP and Scrum, which points out the need to conduct estimation studies with other agile methods as well. With respect to the development activity being investigated, only implementation and testing have been investigated in isolation. About 36%(9) of the primary studies took into all activities during effort prediction, which may be due to the fact that in an ASD process, the design, implementation and testing activities are performed very closely to each other.

During this SLR, we found very few studies reporting all the required elements appropriately e.g. 40%(10) of the studies have not described the exact agile method used in the study, 24%(6) of them have not described the development activity that is subject to estimation, and 26%(6) of the studies have not used any accuracy metric. Quality checklists help in assessing the quality of the studies’ evidence; however it proved challenging to assess the quality of all types of studies by means of a generic checklist. A potential threat to validity for this SLR is the issue of coverage i.e. Are we able to cover all primary studies? We have aimed to construct a comprehensive string, following the guidelines proposed in evidence based SE literature, which covers all relevant key terms and their synonyms. Search strings were applied to multiple databases that cover all major SE conferences and journals. Lastly we have also performed a secondary search in order to make sure that we did not miss any primary study. In regard to the quality of studies’ selection and data extraction, we have also used a process in which a good percentage of studies (details in section 2.2.4 above) were selected and extracted separately by more than one researcher, and later aggregated via consensus meetings, in order to minimize any possible bias.

### 2.5 Conclusions

This study presents a systematic literature review on effort estimation in Agile Software Development (ASD). The primary search fetched 443 unique results, from which 32 papers were selected as potential primary papers. From these 32 papers, 19 were selected as primary papers on which the secondary search process was applied resulting in one more paper added to the list of included papers. Results of this SLR are based on 20 papers that report 25 primary studies. Data from these 25 studies were extracted and synthesized to answer each research question.

Although a variety of estimation techniques have been applied in an ASD context, ranging from expert judgment to Artificial Intelligence techniques, those used the most are the techniques based on some form of expert-based subjective assessment. These techniques are expert judgment, planning poker, and use case points method. Most of
the techniques have not resulted in good prediction accuracy values. It was observed that there is little agreement on suitable cost drivers for ASD projects. Story points and use case points, which are close to prevalent requirement specifications methods (i.e. user stories, use cases), are the most frequently used size metrics. It was noted that most of the estimation studies used datasets containing data on within-company industrial projects. Finally, other than XP and Scrum, no other agile method was investigated in the estimation studies.

Practitioners would have little guidance from the current effort estimation literature in ASD wherein we have techniques with varying (and quite often low) level of accuracy (in some case no accuracy assessment reported at all) and with little consensus on appropriate cost drivers for different agile contexts. Based on the results of this SLR we recommend that there is strong need to conduct more effort estimation studies in ASD context that, besides size, also take into account other effort predictors. It is also suggested that estimation studies in ASD context should perform and report accuracy assessments in a transparent manner as per the recommendations in the software metrics and measurement literature. Finally, we also believe that further investigation into the use of cross-company datasets for agile effort estimation is needed given the possible benefits that the use of such datasets would perhaps bring to companies that employ ASD practices but that do not yet have their own datasets on software projects to use as basis for effort estimation.
2.6 References


Chapter 2. Effort Estimation in Agile Software Development: A Systematic Literature Review


Chapter 2. Effort Estimation in Agile Software Development: A Systematic Literature Review


Appendix: List of included primary studies


Chapter 2. Effort Estimation in Agile Software Development: A Systematic Literature Review


Chapter 3

Effort Estimation in Agile Software Development: A Survey on the State of the Practice

3.1 Introduction

In Agile Software Development (ASD) planning is carried out iteratively at different levels such as release planning, sprint planning and current day planning [3]. Effective planning needs estimates that are relatively accurate and reliable. Effort estimation in ASD is an active research area wherein different studies have already been reported in the literature. A recent Systematic Literature Review (SLR) [1] aggregated and described the state of the art related to estimation techniques, effort predictors, accuracy measures and agile methods used, gathered from 20 papers reporting 25 studies. Due to its focus on fully refereed empirically-based research, the SLR did not include works that were not peer reviewed, white papers, technical reports, and practitioners’ opinion in articles/forums/blogs. As a follow up research, the survey detailed herein gathered evidence on the state of the practice on effort estimation in ASD, in order to complement the findings from the SLR and provide a combined and detailed understanding relating to effort estimation in ASD.

There have been three previous surveys that investigated topics related to estimation
Chapter 3. Effort Estimation in Agile Software Development: A Survey on the State of the Practice

in ASD; however they fell short of presenting the state of the practice with respect to the specific estimation techniques and predictors used, accuracy level achieved by different techniques and the development context in which estimation is practiced in ASD. Further description of these surveys and their pitfalls with respect to our study are given in Section 2.

The remainder of the chapter is organized as follows: Section 3.2 presents the related work; the research methodology is described in Section 3.3, followed by the presentation of the results in Section 3.4. Next, a discussion of the results and the conclusions are given respectively in Sections 3.5 and 3.6.

3.2 Related Work

We identified three surveys investigating topics related to estimation in ASD.

In the first survey, published in 2005 [15], project managers from 18 Norwegian companies were interviewed. Data about 52 projects was used to analyze the differences in schedule and effort overruns between projects using flexible (incremental, agile) and sequential process models. The study concluded that the projects that used a flexible process model experienced less effort overruns when compared to those projects that used sequential models. The data gathered from this survey was also used in a study [19] on different aspects of software estimation (effort and schedule overruns, choice of estimation method, level of estimation skill etc.) in the Norwegian industry. However, this study made no distinction between projects/companies using agile or traditional development processes.

The second survey [11] investigated the relationship between customer collaboration and the effort overruns. It is based on interviews with project managers from a medium sized Norwegian company employing an agile methodology. The data covered 18 different projects. This survey concluded that a lesser magnitude of effort overruns is experienced whenever customer collaboration is supported via daily communication with the customer. The third survey [17] explored the concept of “expected implementation time” of a user story as a granularity characteristic. The results showed that developers face more problems, e.g., estimation related issues, when they dealt with coarse grained user stories. None of these surveys reported the state of the practice with respect to the specific estimation techniques and predictors used, accuracy of the used techniques and the development context in which estimation is practiced specifically in the ASD. To achieve this aim, we conducted this empirical study that is based on practitioners from any country, from software companies practicing agile methods or practices.
3.3 Research Methodology

In this section we describe research questions, study type, design, execution and validity threats.

3.3.1 Research Questions

Five research questions are investigated in this survey:

- **RQ1**: Which effort estimation techniques, and in which combinations, are used in practice in ASD?
- **RQ2**: Which effort predictors (size metrics, cost drivers), and in which combinations, are used in practice for effort estimation in ASD?
- **RQ3**: What is the agile development context in which these techniques are used in practice?
- **RQ4**: How accurate are the effort estimates in ASD?
- **RQ5**: What are the reasons for inaccurate estimates in ASD?

3.3.2 Study Type, Design and Execution

Our goal was to gather retrospective data from as large as possible a sample of practitioners who have been involved in effort estimation in ASD. Therefore a survey investigation was carried out [6, 7, 9]. An on line web-based questionnaire, hosted at surveymonkey.com, was used as instrument for data collection as it offers a low cost and efficient mechanism to gather data from respondents [7, 10]. The questionnaire was accessible between August and October of 2014. The software practitioners who have been involved in the effort estimation in ASD represented the survey population. The survey participants were recruited initially through personal contacts and via posting the survey link to the agile and measurement-related forums at LinkedIn. A snowball sampling approach [7] was also used via our initial industry contacts. Finally, the questionnaire’s URL and goal were advertised by one of the authors at three conferences (PROMISE 2014, RE 2014, ICGSE 2014). Convenience sampling was used [7].

A total of 82 respondents started the questionnaire, whereof 60 (73%) completed all mandatory questions. Therefore, we only used data from these 60 respondents. The questionnaire was organized into two parts: part 1 gathered demographic data, while part 2 collected data on the respondents’ practical experience in effort estimation in
ASD. The questions from part 2 correspond in the following way to the RQs described in Subsection 3.3.1:

- **Survey Question(s) for RQ1**: Which of the following effort estimation technique(s) are used in your company to estimate effort? Respondents could select multiple items from a list of possible techniques. In addition, a text field was provided for further answers.

- **Survey Question(s) for RQ2**: (1) Which of the following size metric(s) are used in your company in estimating the effort? A list of metrics plus a text field were provided for answering. (2) Which of the following factors do you believe are fundamental to consider when estimating the effort? A list of factors (cost drivers) plus a text field were provided for answering.

- **Survey Question(s) for RQ3**: (1) What are the agile development activities in your company for which effort is estimated? A list of development activities plus a text field were provided for answering. (2) What are the most common situations when effort is estimated for agile projects in your company? A list of planning stages/levels plus a text field were provided for answering.

- **Survey Question(s) for RQ4**: In your experience the effort estimates for a sprint/iteration, when compared to the actual effort, are on average: 1) Spot on (0–5%), 2) Under-estimated by (5–25%), 3) Over-estimated by (5–25%), 4) Under-estimated by (25–50%), 5) Over-estimated by (25–50%), 6) Under-estimated by 50% or more, and 7) Over-estimated by 50% or more. The seven point ordinal scale was used for answering this question. Participants had to select exactly one option. No additional text field was provided.

- **Survey Question(s) for RQ5**: If estimates are on average within or above 35% from actuals, what do you think are the possible reasons for such difference between actual and estimated effort? Only a free text field was provided for answering.

### 3.3.3 Validity Threats

This section discusses validity threats using the four types of threats suggested in [2, 8].

- **Construct validity** is concerned with issues that may arise due to improper design of the survey instrument, which then may not be measuring properly what it is supposed to measure. We believe that this issue has been mitigated, since the questionnaire was iteratively designed and updated by the authors based on
the results from the SLR [1]. Further, another two researchers with experience in ASD validated the questionnaire regarding its completeness and readability.

- **Internal validity** is concerned with issues, such as confounding factors or irrelevant respondents, which could introduce a systematic error or bias in the study results. A number of steps were taken to mitigate this threat: (i) It was clearly stated in the survey introduction that only practitioners with actual experience in effort estimation in ASD should participate. In addition, besides asking respondents about their experience in the software industry, we also asked them about their experience in ASD to ensure that all respondents were also agile practitioners. All respondents (60) answered this question; (ii) Respondents were assured of their anonymity to avoid evaluation apprehension; and (iii) We matched the respondents’ countries, as provided in the survey, against the IP addresses collected by the survey tool, and found no mismatch.

- **External validity**, also called generalizability, refers to the extent to which findings in a study are applicable outside of the study context. Since the sample used herein was a convenience sample, our results are only generalizable to those agile teams and companies that share similar characteristics to our survey respondents’ teams and companies. However, a number of steps, such as advertising our survey on ASD-related on-line forums and conferences and a snowball approach, were taken to obtain a broad sample representative of our population of interest. We believe that these steps contributed to obtaining a sample that is quite heterogeneous in terms of experience, job role and country.

- **Conclusion Validity** is related with the possibility of reaching incorrect conclusions about association in observations due to errors such as use of inadequate statistical tests or measures. In this study, we only used frequencies and percentages to identify common patterns or practices to point out potential areas or relationships for future research efforts. In addition, we only considered complete responses in our analysis.

3.4 **Results**

This section presents the results of our survey organized by research questions.

3.4.1 **Respondents’ Demographics**

To understand the respondents’ context and background, several demographic questions were asked.
Chapter 3. Effort Estimation in Agile Software Development: A Survey on the State of the Practice

Country

Respondents from 5 continents and 16 countries participated in the survey (see Table 3.1). Most responses were from Brazil and the USA. However, in relation to the number of respondents per continent, the majority came from Europe.

Table 3.1: Respondents by continents and countries

<table>
<thead>
<tr>
<th>Continent</th>
<th>Countries</th>
<th>F (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Europe</td>
<td>Sweden(4), UK(2), Italy(2), NL(2), Denmark(2), France(2), Latvia(1), Ireland(1), Estonia(1)</td>
<td>17 (28%)</td>
</tr>
<tr>
<td>S. America</td>
<td>Brazil(15), Argentina(2)</td>
<td>17 (28%)</td>
</tr>
<tr>
<td>N. America</td>
<td>USA(13), Canada(1)</td>
<td>14 (23%)</td>
</tr>
<tr>
<td>Asia</td>
<td>Pakistan(4), India(2)</td>
<td>6 (10%)</td>
</tr>
<tr>
<td>Australia</td>
<td>Australia(6)</td>
<td>6 (10%)</td>
</tr>
</tbody>
</table>

Job Role

Figure 3.1 presents the respondents’ job roles, along with the corresponding frequencies and percentages. Results suggest that a variety of roles, both managerial and non-managerial, participate in effort estimation activities in agile projects.

- **Managerial roles**: Managerial roles constitute 50% of the total respondents and include project managers, program managers, product owners, scrum masters and team leads.

- **Non-Managerial roles**: Non managerial roles constitute around 32% of the total respondents and include developers, testers, analysts and designers.

- **Other roles**: A total of 11 (18%) respondents described other roles in the free text field provided for this question, such as estimation expert, agile coach, project coach, consultant, and business analyst.

Estimation and planning are done at different levels in ASD, e.g., release planning, sprint/iteration planning and daily planning [3], which demands the involvement of people both in managerial and non-managerial positions at different project stages.
Agile Experience

Figure 3.2 depicts the information about respondents’ experience in ASD along with the corresponding frequencies and percentages. A total of 62% of the respondents have more than three years of experience in ASD, while only 10% have less than 1 year of ASD experience. We also asked the respondents to provide information regarding their total professional experience, agile or otherwise, in the software industry. The majority of the respondents (about 82%) have more than 5 years of experience in the software industry, suggesting that most of the survey respondents are senior industry practitioners. The high number increases our confidence in the validity of the responses and also highlights the practice of involvement of seniors in the estimation process in ASD.

Team Size, System Types and Agile Methods

Most respondents work in teams of up to 10 team members; 38% reported a team size of 1–5, and 40% a team size of 6–10. Only few reported a team size greater than 20. Most of the respondents’ companies develop business applications (62%). Some companies also develop E-commerce (27%), data processing (22%) and financial (20%) applications, few develop safety critical (7%) and embedded systems (13%). Scrum, Kanban and eXtreme Programming are the most commonly used agile methods, while
Chapter 3: Effort Estimation in Agile Software Development: A Survey on the State of the Practice

3.4.2 RQ1: Estimation Techniques

Figure 3.3 details the responses on estimation techniques practiced in ASD along with the corresponding frequencies and percentages. In our sample, planning poker (63%) is the most frequently practiced effort estimation technique in ASD, followed by estimation by analogy (46%) and expert judgment (38%). All of the techniques that are selected by more than 10% of the participants share one common aspect: these techniques use subjective assessment of experts in some way to arrive at an effort estimate. These subjective assessment based techniques (planning poker, analogy, expert judgment) seem to align well with the agile philosophy, where much of the knowledge is managed tacitly and more emphasis is placed on people and their interaction. Other techniques reported in the text field include SLIM (2) and methods based on IFPUG (1) and COSMIC function points (1).

To identify the combinations in which effort estimation techniques are practiced in ASD, respondents were allowed to select multiple techniques. 52% (31) of the respondents selected multiple effort estimation techniques, while 48% (29) selected a single technique. Planning Poker was selected by 38 (63%) agile practitioners in total, out of which it is practiced alone by 18 (47%) and in combination with other techniques.
by 20 (53%). Planning poker was reported as being practiced in combination with following techniques:

- Estimation by analogy (6 agile practitioners)
- Estimation by analogy and expert judgment (5 agile practitioners)
- Estimation by analogy and Delphi (2 agile practitioners)
- Estimation by analogy and COSMIC function points (1 agile practitioner)
- Estimation by analogy, expert judgment, use case points method and SLIM (1 agile practitioner)
- Expert judgment (3 agile practitioners)
- Use case points method (2 agile practitioners)

It can be inferred from the above data that when planning poker is practiced in combination with other techniques, estimation by analogy and expert judgment are the preferred
choices for agile practitioners. Besides planning poker, other techniques are also practiced in combination, e.g., estimation by analogy and expert judgment (selected by 9 practitioners). It is interesting to note that estimation by analogy alone was selected by a single agile practitioner only, while expert judgment by 5 agile practitioners. The results suggest that these techniques (i.e. estimation by analogy and expert judgment) are mostly used in combination with other techniques in ASD. On the other hand, Planning poker is used both solo and in combination with other estimation techniques in ASD.

### 3.4.3 RQ2: Effort Predictors

Size metrics and cost drivers constitute effort predictors, which are described in the following subsections.

#### Size Metrics

61% (37) of the respondents in this survey selected story points as the size metric (see Figure 3.4). This finding aligns well with the results on the estimation techniques presented above, wherein planning poker was found to be the most frequently practiced estimation technique in the ASD context. Other size metrics are also used to some degree – function points (17%) and use case points (10%). Respondents did not report any additional size metrics in the text field provided for this question.

![Figure 3.4: Size Metrics](image)

Based on the respondents’ data, we identified some patterns in the application of size metrics in ASD during the effort estimation process. According to 68% (25 out
of 37) of the respondents story points are used solo, while 32% (12 out of 37) selected story points in combination with the following metrics:

- Function points (6 agile practitioners)
- Function points and use case points (2 agile practitioners)
- Function points and LOC (1 agile practitioner)
- Function points, use case points and lines of code (LOC) (1 agile practitioner)
- Use case points (2 agile practitioners)

Respondents’ data suggest that other size metrics are rarely used solo; function points are used solo according to 4 (out of 17) respondents only and LOC is never used solo. Our data suggest that most agile practitioners consider story points as the most preferred choice when measuring size for ASD projects. Story points are mostly used solo but when used in combination with other size metrics, they are mostly combined with function points and/or use case points.

Cost Drivers

Figure 3.5 details the responses in terms of frequencies and percentages for the different cost drivers. Team’s expertise level and team’s prior experience are considered to be the most important cost drivers by most agile practitioners in our sample. Task size, project domain and non-functional requirements are also given considerable importance by agile practitioners. In the text field, respondents mentioned cost drivers such as complexity, team stability, integration issues and team velocity. It was interesting to observe that 10 respondents did not select any cost driver, however they selected size metric(s) as follows: story points (5), story and function points (2), function points (2) and use case points (1).

Results also show that most of the time a combination of cost drivers are used to arrive at an effort estimate. The two most highly used cost drivers – team’s expertise level and team’s prior experience, are rarely used solo according to our sample data. Both were selected as solo cost drivers by only one respondent each, but together in combination by 36 respondents. Other cost drivers that are also, at times, used in combination with these top two cost drivers are task size, project domain and non-functional requirements. Both of the most frequently used cost drivers are related to teams, which resonates well with agile philosophy, wherein individuals and teams are highly valued. We believe that agile managers should always take into account these team related cost drivers when estimating the effort in ASD.
3.4.4 RQ3: Estimation and Agile Development Context

Through this research question we wanted to understand the agile development context in which the process of effort estimation is carried out in ASD. Two aspects related to the ASD context are considered in this question: Planning levels (release, sprint or current day planning) and development activities (i.e. analysis, design, implementation, testing or maintenance).

Effort Estimation and Agile Planning Levels

Respondents were asked to select the planning levels or stages at which they perform the effort estimation process. Figure 3.6 details the responses against each planning level along with the frequency and percentage. A total of 82% (49) of the respondents said that effort estimation is carried out during sprint/iteration planning. Release planning was selected by about 42% (25) respondents. Our results suggest that estimation plays a pivotal role during sprint and release planning for most agile teams. It
is interesting to note that a relatively a small percentage of respondents selected daily planning (about 17%) and project bidding (about 17%), indicating that at least for our sample, few teams estimate effort in the daily planning or project bidding stage. Five respondents provided answers in the text field: feasibility study, weekly task planning and product road map planning.

Figure 3.6: Estimation at agile planning levels

After analyzing the data for estimation versus planning levels at a broader level, we next present the results at the individual technique level. This is done to highlight how specific techniques are used by agile teams at different planning levels. We only consider planning poker, estimation by analogy and expert judgment for this discussion as they are the ones that are used most frequently, see Section 3.4.2 above. Table 3.2 presents the planning levels with different estimation techniques along with the frequency and percentage for each case. The trends for each technique are in line with the overall trend described in the preceding paragraph, i.e. most agile teams, using any of the estimation techniques listed in Table 3.2, estimate effort at the sprint planning level, or at both sprint and release planning levels. The entries in boldface in Table 3.2 highlight these trends. Since it is difficult to estimate in absence of detailed information at the project bidding stage only few respondents (8) selected bidding, indicating that few agile teams are able to estimate effort at the bidding or start up phase. In 7 out of these 8 cases, expert judgment, in combination with some other technique, is used to estimate effort at the bidding level.
Chapter 3. Effort Estimation in Agile Software Development: A Survey on the State of the Practice

Table 3.2: Planning levels versus estimation techniques

<table>
<thead>
<tr>
<th>Planning Level</th>
<th>PP</th>
<th>Analogy</th>
<th>EJ</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bidding (B)</td>
<td>0(0%)</td>
<td>0(0%)</td>
<td>0(0%)</td>
</tr>
<tr>
<td>Release (R)</td>
<td>2(5%)</td>
<td>3(11%)</td>
<td>1(5%)</td>
</tr>
<tr>
<td>Sprint (S)</td>
<td>17(45%)</td>
<td>7(25%)</td>
<td>6(27%)</td>
</tr>
<tr>
<td>Daily (D)</td>
<td>0(0%)</td>
<td>1(4%)</td>
<td>2(9%)</td>
</tr>
<tr>
<td><strong>R &amp; S</strong></td>
<td>10(26%)</td>
<td>11(39%)</td>
<td>5(23%)</td>
</tr>
<tr>
<td>R, S &amp; D</td>
<td>2(5%)</td>
<td>1(4%)</td>
<td>1(5%)</td>
</tr>
<tr>
<td>S &amp; D</td>
<td>2(5%)</td>
<td>0(0%)</td>
<td>0(0%)</td>
</tr>
<tr>
<td>S &amp; B</td>
<td>3(8%)</td>
<td>2(7%)</td>
<td>4(18%)</td>
</tr>
<tr>
<td>R &amp; D</td>
<td>1(3%)</td>
<td>0(0%)</td>
<td>0(0%)</td>
</tr>
<tr>
<td>B, R &amp; S</td>
<td>0(0%)</td>
<td>2(7%)</td>
<td>2(9%)</td>
</tr>
<tr>
<td>All</td>
<td>1(3%)</td>
<td>1(4%)</td>
<td>1(5%)</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td>38</td>
<td>28</td>
<td>22</td>
</tr>
</tbody>
</table>

PP=Planning Poker, EJ=Expert Judgment

Effort Estimation and Agile Development Activities

Software projects progress through different development activities (analysis, design, implementation etc.) that all contribute to the effort required to complete a project. Respondents were asked to select activities for which effort is estimated in their organizations. Figure 3.7 details the responses along with the frequencies and percentages of responses against each development activity. It is clear from the figure that most agile teams estimate effort for implementation (about 87%) and testing (about 68%) activities. Two thirds (40 of 60) of the respondents selected implementation and testing activities together in combination. Since test driven development is a common practice in ASD many agile teams consider both implementation and testing when doing effort estimation. Analysis and design activities are selected by a relatively small number of respondents, i.e. 26 (about 43%) and 32 (about 53%), respectively. On the other hand, also maintenance was selected by few respondents (about 23%). This might indicate that only few agile teams take into account maintenance activities when they estimate effort on an agile development project.

Next, Table 3.3 shows, for the most frequently used estimation techniques, the development activities and their combinations for which effort is estimated according to our survey respondents. Results show that, irrespective of the estimation technique, most of the agile teams take into account multiple activities to estimate effort. Imple-
mentation (I) and testing (T) are the leading activities that are mostly considered for estimating the effort, irrespective of the estimation technique. The majority of the respondents who selected implementation also selected the testing activity. This trend is consistent across all techniques, i.e. planning poker (78%), analogy (77%) and expert judgment (80%). The pattern I&T appears frequently and is emphasized in Table 3.3 (in boldface). There are only two respondents with all three estimation techniques that did not select the implementation activity. On the other hand, the testing activity was never selected solo. Our results suggest that irrespective of the effort estimation technique, a good number of agile teams do not take into account analysis, design and maintenance activities during the effort estimation process.

### 3.4.5 RQ4: Estimation Accuracy

The RQ4 aims to identify respondents’ beliefs regarding the amount of estimation error (actual effort – estimated effort), based on their experiences in ASD. Figure 3.8 presents the results arranged by the residual or error range. Note that the same residual range was also employed in the survey instrument. Close to 22% (13) of the respondents believe that the estimates in their company or team are spot on (i.e. 0–5% error).
Chapter 3. Effort Estimation in Agile Software Development: A Survey on the State of the Practice

Data suggests that the tendency to underestimate effort is higher than the tendency to overestimate, which supports some of the existing body of evidence in software effort estimation [20]. According to 35% (21) of the respondents, effort estimates are on average underestimated by 25% or more in their company, while 7% (4) reported underestimation by 50% or more. The tendency to underestimate development effort could be attributed to overoptimism and team members’ inexperience in ASD, as inexperienced developers may exhibit the tendency to consider the best case scenario only.

Table 3.3: Development activities versus estimation techniques

<table>
<thead>
<tr>
<th>Development Activity</th>
<th>PP</th>
<th>Analogy</th>
<th>EJ</th>
</tr>
</thead>
<tbody>
<tr>
<td>Analysis (A) only</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Design (D) only</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td><strong>Implementation (I) only</strong></td>
<td>2</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>Testing (T) only</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Maintenance (M) only</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>All (A&amp;D&amp;I&amp;T&amp;M)</td>
<td>5</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>A&amp;I only</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>D&amp;I only</td>
<td>2</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>I&amp;T only</td>
<td>8</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>I&amp;M only</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>A&amp;D&amp;I only</td>
<td>2</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>D&amp;I&amp;T only</td>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>I&amp;T&amp;M only</td>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>A&amp;I&amp;T only</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>A&amp;D&amp;I&amp;T only</td>
<td>5</td>
<td>6</td>
<td>4</td>
</tr>
<tr>
<td>D&amp;I&amp;T&amp;M only</td>
<td>4</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>D&amp;I&amp;T&amp;Management only</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td>38</td>
<td>28</td>
<td>22</td>
</tr>
</tbody>
</table>

PP=Planning Poker, EJ=Expert Judgment

Table 3.4 provides an aggregated view of the estimation error or residual and shows that, according to 52% of the respondents, effort estimates in ASD are out of range by factor of 25% or more.

Our results suggest that the effort estimates of around half of the agile teams are inaccurate by a factor of 25% or more. It seems worthwhile to investigate further the respondents’ estimation accuracy responses with respect to the estimation techniques and effort predictors used to check whether there is an association between the choice of
estimation techniques and predictors and accuracy levels. Table 3.5 presents a detailed view of the different estimation error ranges against three different factors: estimation techniques, size metrics and cost drivers. The estimation error ranges are divided into two broad groups: error range of 0–25% (both over and under) and 25% and above. The first group represents an acceptable accuracy level, while the second one represents a potential concern for the agile teams. In the following, we discuss estimation error ranges in the light of these three factors.

- **Estimation error vs estimation techniques**: When we look at the percentages in the techniques columns in the Table 3.5, we can see that for the high accuracy group (0–25% error range), multiple estimation techniques were selected about twice as often (66%) as single techniques (34%). For the low accuracy
group (25% and above error range) this is the opposite: 61% of the respondents selected a single technique, while 39% selected multiple technique. Our data therefore suggests that the use of a solo or combination (ensemble) of estimation techniques may have an effect upon estimation accuracy. However, further investigation is needed before any conclusions or generalizations can be made. A similar result was reported in the domain of web effort estimation [16], where it was observed that ensembles of estimation techniques consistently performed better than solo techniques.

**Estimation error vs size metrics:** Regarding the possible impact of the choice of size metric on the estimation error, we can observe the following: (1) A total of 76% of the respondents in the high accuracy group (0–25% error range) selected story points as the size metric, whereas 48% in the low accuracy group. (2) For size metrics other than story points, the distribution is the opposite; 31% of the respondents in the high accuracy group selected another size metric, while 42% of the respondents in the low accuracy group selected another size metric. Combining point 1 and 2, it can be said that story points are a preferred choice for most of the respondents in the high accuracy group only. (3) Regarding multiple size metrics, 28% of the respondents in the high accuracy group selected multiple metrics, while 19% of respondents in low accuracy group selected multiple size metrics.

Empirical evidence shows that size metrics correlate with effort, thus suggesting their use as effort predictors. However, to our surprise, some respondents did not select any size metric indicating the possibility of arriving at an effort estimate without using any size metric. Only 14% of the respondents in the high accuracy group did not select any size metric, while 26% of the respondents in the low accuracy group did not select any size metric. Such results suggest that not considering the size of the system to be developed during effort estimation could have a detrimental effect upon the accuracy of the estimates. Combining these points suggests that in the ASD context story points seem to be related with better accuracy, while other size metrics are also employed but only in combination with story points. Lastly it may not be advisable to completely ignore system size during effort estimation.

**Estimation error vs cost drivers:** There is not much difference in the usage of cost drivers for both groups, i.e. respondents in both high and lower accuracy groups mostly selected multiple cost drivers. Most of the agile teams use multiple cost drivers during the effort estimation process.
Table 3.5: Estimation error versus techniques and predictors

<table>
<thead>
<tr>
<th>Estimation Error</th>
<th>Techniques</th>
<th>Size Metrics</th>
<th>Cost Drivers</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>M  S</td>
<td>SPs Other</td>
<td>M  No</td>
</tr>
<tr>
<td>Spot on (0–5%) (13)</td>
<td>9  4</td>
<td>9  3  2  3</td>
<td>11 0 2</td>
</tr>
<tr>
<td>U by 5% or more (11)</td>
<td>6  5</td>
<td>10  6  6  0</td>
<td>9  0 2</td>
</tr>
<tr>
<td>O by 5% or more (5)</td>
<td>4  1</td>
<td>3  1  0  1</td>
<td>4  1 0</td>
</tr>
<tr>
<td><strong>Error (0–25%) (29)</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Frequency</td>
<td>19 10</td>
<td>22 9 8 4</td>
<td>23 1 4</td>
</tr>
<tr>
<td>Percentage</td>
<td>6634</td>
<td>76312814</td>
<td>79314</td>
</tr>
<tr>
<td>U by 25% or more (21)</td>
<td>10 11</td>
<td>11 9 4 4</td>
<td>16 2 3</td>
</tr>
<tr>
<td>O by 25% or more (6)</td>
<td>1  5</td>
<td>2  2 0 2</td>
<td>4  1 1</td>
</tr>
<tr>
<td>U by 50% or more (4)</td>
<td>1  3</td>
<td>2  2 2 2</td>
<td>1  1 2</td>
</tr>
<tr>
<td><strong>Error (25+%) (31)</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Frequency</td>
<td>12 19</td>
<td>15 13 6 8</td>
<td>21 4 5</td>
</tr>
<tr>
<td>Percentage</td>
<td>39 61</td>
<td>48 421926</td>
<td>681316</td>
</tr>
</tbody>
</table>

SPs = Story points, M = Multiple, S = Single

3.4.6 RQ5: Reasons for Inaccurate Estimates in ASD

A total of 28 respondents provided feedback for this open ended question, discussing a range of possible reasons for inaccurate estimates in ASD. These reasons were organized into seven categories, which are presented below and summarized in Table 3.6.

- The category representing requirements-related reasons received most contributions (9 respondents). The following specific reasons were suggested: poorly specified user stories, missing requirements, overlooking the non-functional requirements (NFR) and requirements change. Requirements are used as input to estimate size, which is then employed as input to the estimation process. The accuracy of estimates can be improved whenever requirements and user stories are well understood, and when NFRs are taken into account. The survey results support previous evidence where both changing and new requirements were found to be one of the top reasons for effort overruns (e.g., [12]).

- The second category represents project management-related issues (7 respondents) including the following reasons for inaccurate estimates: poor change control, scrum master not guiding the team, scope creep and high employee turnover. When such issues are not managed properly, they may have a negative impact on the development time and project cost, and therefore increase the risk for
estimation errors.

- Team-related issues (5 respondents) represent the third category. Respondents provided the following specific reasons for inaccurate estimates: insufficient experience of the development team, knowledge sharing and management problems in the team and the presence of unskilled members in the team. The lack of team expertise and experience were also found to be the reasons for inaccurate estimates in ASD in [18]. It is important that these issues are considered when estimating effort and managed during project execution to mitigate issues that may have a detrimental effect upon estimation accuracy.

- Overoptimism (3 respondents) comprises two specific reasons. The first reason was considering only the best case scenario and purposefully underestimating the effort to obtain a contract. It is important to be mindful of the tendency of developers to understate the effort by only considering the best case scenario. The overoptimism trend in the ASD context is also observed in other studies, such as [13, 14, 17]. The second reason regarding the purposeful underestimation is an unfair practice and is a clear breach of the code of ethics for software engineers described in [4].

- Lack of a formal estimation process was reported by 3 respondents. Such results are in line with the findings from another study in agile cost estimation [18]. In the ASD context people and their interactions are valued more than processes and tools, but this does not mean that agile practices advocate for the estimation process dimension to be completely ignored. We suggest that agile managers should ensure that a process, that is aligned with the agile philosophy, is in place for effort estimation.

- The sixth category represents ignoring the testing effort (2 respondents). Effort is underestimated when developers doing the estimation mainly consider implementation effort, and ignore the testing effort. It is important for the agile managers to ensure that estimators are considering all development activities to obtain a reliable effort estimate. One way for ensuring this is to include people from the testing teams in the estimation sessions.

- The seventh and last category (2 respondents), represents the lack of customer involvement during story sizing sessions. User stories often lack the required details making it challenging to arrive at a reliable size estimate. In such situations, the presence of customer helps resolve ambiguities. Lack of customer collaboration and communication has also been associated with effort overruns
in the ASD context elsewhere [11]. Most of the studies included in a systematic review by Grimstad et al. [12] reported that customer characteristics, such as involvement and commitment, are considered important for estimation accuracy.

Table 3.6: Reasons for inaccurate estimates in ASD

<table>
<thead>
<tr>
<th>#</th>
<th>Category of reasons for inaccurate estimates</th>
<th>F(%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Requirements’ related issues (Missing and changing Reqs., Poor user stories, Overlooking NFRs etc.)</td>
<td>9 (32%)</td>
</tr>
<tr>
<td>2</td>
<td>Project management issues (Poor change control, SM not guiding the team, Scope creep)</td>
<td>7 (25%)</td>
</tr>
<tr>
<td>3</td>
<td>Team related issues (Inexperience, knowledge sharing problem in team, Unskilled team members)</td>
<td>5 (18%)</td>
</tr>
<tr>
<td>4</td>
<td>Overoptimism (Best case scenario, Purposely underestimating to obtain work)</td>
<td>3 (10%)</td>
</tr>
<tr>
<td>5</td>
<td>Lack of formal estimation process</td>
<td>3 (10%)</td>
</tr>
<tr>
<td>6</td>
<td>Ignoring testing effort</td>
<td>2 (7%)</td>
</tr>
<tr>
<td>7</td>
<td>Insufficient customer involvement during story sizing</td>
<td>2 (7%)</td>
</tr>
</tbody>
</table>

3.5 Discussion

This section provides a detailed discussion within the context of our research questions, which correspond to the following topics:

- Effort estimation techniques practiced in ASD
- Effort predictors (size metrics and cost drivers) practiced in ASD
- Agile contexts in which estimation is practiced in ASD
- Accuracy of effort estimates in ASD.

Our results show that planning poker, estimation by analogy and expert judgment are the techniques that are most frequently practiced in ASD. In ASD much of the knowledge is managed tacitly with higher emphasis on people and their interactions [21]. This probably is the reason due to which most frequently used estimation techniques in ASD, as per our survey respondents, are the ones that rely on subjective assessment of the experts to arrive at an estimate. These findings are in line with the
Chapter 3. Effort Estimation in Agile Software Development: A Survey on the State of the Practice

results of an SLR [1] on effort estimation in ASD. A total of 52% of the respondents use a combination of estimation techniques. Results also show that planning poker is the only technique that is practiced both in combination with other techniques and solo also; it is combined either with estimation by analogy or with expert judgment. As for other estimation techniques, respondents rarely used them solo, which indicates that these techniques are not considered comprehensive on their own by most of the respondents. Note that none of the primary studies included in the SLR [1] investigated the application of effort estimation techniques in combination.

According to our survey results, story points are most frequently used size metrics by agile teams followed by function points (FPs) and use case points (UCPs). These findings also corroborate those from the SLR [1]. Story points are used solo by most agile teams (68%), while in some cases they are used in combination with other metrics such as FPs/UCPs. Other size metrics, e.g., FPs or UCPs, are rarely used solo in ASD, as per our respondents. Our results suggest that planning poker and story points are considered a reliable combination by most agile teams. Other estimation techniques and metrics, when used, act in aid of this combination.

Team related cost drivers, i.e. team’s expertise level and prior experience, are the ones that are considered to be the most important by the respondents. This finding is in line with the results of the SLR [1], where these team related cost drivers were the ones that are identified in most of the primary studies. This corresponds well with the agile philosophy wherein the team members are more important than tools or processes. However we also identified some cost drivers in the survey presented here, e.g. project domain and non functional requirements, that are considered important by several respondents (about 40%); these findings are contrary to those from the SLR [1]. Results also show that multiple cost drivers are used by most agile teams.

We now discuss the agile context, in terms of agile planning levels and development activities, in which estimation is practiced in ASD. Effort estimation is carried out mostly at sprint planning (82% of the respondents) and release planning (42%) levels. Few respondents selected daily planning and project bidding and feasibility study stages as well where estimation is carried out. These findings also corroborate those from the SLR [1]. It is also identified in this survey that most agile teams, irrespective of the estimation technique and predictors used, take into account implementation and testing activities for effort estimation; and relatively fewer teams consider activities such as analysis and design as well. This may be one of the reasons of the under estimation trend in ASD, as discussed in section 4.5.

According to 52% of the respondents effort estimates in ASD are out by an estimation error of 25% and above, wherein the dominating trend is of under estimation. Estimation error is on the lower side in the cases when respondents’ teams are practicing multiple estimation techniques. There appears to be an association between use
of story points and low estimation error i.e. the majority of the respondents (76%) in the high accuracy group selected the usage of story points as size metrics, while 48% of respondents in low accuracy group selected story points. These finding regarding the associations between estimation techniques or size metrics with accuracy of estimates needs further investigation before we can make any generalizations. Most respondents cited requirements related issues as one of the main reason for inaccuracy of effort estimates in ASD. Incomplete user stories and overlooking the non-functional requirements (NFRs) can lead to underestimating the total effort. NFRs consume considerable design and implementation effort; it is important to incorporate their impact in size and effort estimates otherwise it is possible that team’s productivity may turn out to be much lower than expected because they are required to meet the demanding performance or security or other NFRs [5, chapter 22].

3.6 Conclusions

The chapter presented the results of an empirical study conducted to understand state of the practice on effort estimation in ASD. The study was performed as a survey employing an online questionnaire as instrument to collect data from agile practitioners, across different companies and countries, who have some experience in effort estimation. The results indicate that most agile teams use estimation techniques that rely on experts’ subjective assessment to arrive at an estimate e.g. planning poker, analogy and expert judgment. It was also observed that the usage of a combination of techniques is related with relatively accurate estimates. Story points are the most frequently practiced size metric while other metrics, such as function points or LOC, are rarely used. Regarding the cost drivers it was found that most of the agile teams use team related cost drivers (e.g., team expertise level, team’s prior knowledge) when estimating the effort. According to 52% of the respondents effort estimates in agile are over/under estimated by a factor 25% or more, wherein dominant trend is that of under estimation. Requirements, management and team related issues are cited as the main reasons by some agile practitioners for the larger difference between actual and estimated effort.
Chapter 3. Effort Estimation in Agile Software Development: A Survey on the State of the Practice

3.7 References


Chapter 3. Effort Estimation in Agile Software Development: A Survey on the State of the Practice
Chapter 4

Taxonomies in Software Engineering: A Systematic Mapping Study and a Revised Taxonomy Development Method

4.1 Introduction

In science and engineering, a systematic description and organization of the investigated subjects helps to advance the knowledge in this field [22]. This organization can be achieved through the classification of the existing knowledge. Knowledge classification has supported the maturation of different knowledge fields mainly in four ways:

- Classification of the objects of a knowledge field provides a common terminology, which eases the sharing of knowledge [4, 5, 22].
- Classification can provide a better understanding of the interrelationships between the objects of a knowledge field [22].
- Classification can help to identify gaps in a knowledge field [4, 5, 22].
- Classification can support decision making processes [22].

Summarizing, classification can support researchers and practitioners in generalizing, communicating and applying the findings of a knowledge field [8].
Chapter 4. Taxonomies in Software Engineering: A Systematic Mapping Study and a Revised Taxonomy Development Method

Software Engineering (SE) is a comprehensive and diverse knowledge field that embraces a myriad of different research subareas. The knowledge within many subareas is already classified, in particular by means of taxonomies [24, 25, 28–30]. According to the Oxford English Dictionary [18], a taxonomy is "a scheme of classification". A taxonomy allows for the description of terms and their relationships in the context of a knowledge area. The concept of taxonomy was originally proposed by Carolus Linnaeus [19] to group and classify organisms by using a fixed number of hierarchical levels. Nowadays, different classification structures (e.g., hierarchy, tree and faceted analysis [21]) have been used to construct taxonomies in different knowledge fields, such as Education [13], Psychology [12] and Computer Science [11].

Taxonomies have contributed to mature the SE knowledge field. Nevertheless, likewise the taxonomy proposed by Carolus Linnaeus that keeps being extended [20], SE taxonomies are expected to evolve over time incorporating new knowledge. In addition, due to the wide spectrum of SE knowledge, there is still a need to classify the knowledge in many SE subareas.

Although many SE taxonomies have been proposed in the literature, it appears that taxonomies have been designed or evolved without following particular patterns, guidelines or processes. A better understanding of how taxonomies have been designed and applied in SE could be very useful for the development of new taxonomies and the evolution of existing ones.

To the best of our knowledge, no systematic mapping or systematic literature review has been conducted to identify and analyze the state-of-the-art of taxonomies in SE. In this chapter, we describe a systematic mapping study [26, 27] aiming to characterize the state-of-the-art research on SE taxonomies.

The main contribution of the study reported in this chapter is a characterization of the state-of-the-art of taxonomies in SE. Our results also show that most taxonomies are developed in an ad-hoc way. We therefore revised a taxonomy development method in the light of the findings of this mapping study, our own experience and literature from other research fields with more maturity regarding taxonomies (e.g., psychology and computer science).

The remainder of this chapter is organized as follows: Section 4.2 describes related background. Section 4.3 presents the employed research methodology. The current state-of-the-art on taxonomies in SE, as well as the validity threats associated with the mapping study, are presented in Section 4.4. In Section 4.5, we present a revised method for developing SE taxonomies, along with an illustration of the revised method and its limitations. Finally, our conclusions and view on future work are provided in Section 4.6.
4.2 Background

In this section, we discuss important aspects related to taxonomy design that serve as motivation for the research questions described in Section 4.3.

4.2.1 Taxonomy definition and purpose

Taxonomy is neither a trivial nor a commonly used term. According to the most cited English dictionaries, a taxonomy is mainly a classification mechanism:

- The *Cambridge* dictionary\(^1\) defines taxonomy as “a system for naming and organizing things, especially plants and animals, into groups that share similar qualities”.

- The *Merriam-Webster* dictionary\(^2\) defines taxonomy as “Orderly classification of plants and animals according to their presumed natural relationships”.

- The *Oxford* dictionaries\(^3\) define taxonomy as “The classification of something, especially organisms” or “A scheme of classification”.

Since taxonomy is mainly defined as a classification system, one of the main purposes to develop a taxonomy should be to classify something.

4.2.2 Subject matter

The first step in the design of a new taxonomy is to clearly define the units of classification. In software engineering this could be requirements, design patterns, architectural views, methods and techniques, defects etc. This requires a thorough understanding of the subject matter to be able to define clear taxonomy classes or categories that are commonly accepted within the field [14, 15].

4.2.3 Descriptive bases / terminology

Once the subject matter is clearly defined or an existing definition is adopted, the descriptive terms, which can be used to describe and differentiate subject matter instances, must also be specified. An appropriate description of this bases for classification is important to perform the comparison of subject matter instances. Descriptive bases can

\(^1\)www.dictionary.cambridge.org
\(^2\)www.merriam-webster.com
\(^3\)www.oxforddictionaries.com
also be viewed as a set of attributes that can be used for the classification of the subject matter instances [14, 15].

### 4.2.4 Classification procedure

Classification procedures define how subject matter instances (e.g., defects) are systematically assigned to classes or categories. Taxonomy’s purpose, descriptive bases and classification procedures are related and dependent on each other. Depending upon the measurement system used, the classification procedure can be qualitative or quantitative. **Qualitative** classification procedures are based on nominal scales. In the qualitative classification systems, the relationship between the classes can not be determined. **Quantitative** classification procedures, on the other hand, are based on numerical scales [15].

### 4.2.5 Classification structure

As aforementioned, a taxonomy is mainly a classification mechanism. According to Rowley and Hartley [1] there are two main approaches to classification: enumerative and faceted. In enumerative classification all classes are fixed, making a classification scheme intuitive and easy to apply. It is, however, difficult to enumerate all classes in immature or evolving domains. In faceted classification aspects of classes are described that can be combined and extended. Kwasnik [21] describes four main approaches to structure a classification scheme (classification structures): hierarchy, tree, paradigm and faceted analysis.

**Hierarchy** [21] leads to taxonomies with a single top class that “includes” all sub- and sub-sub classes, i.e. a hierarchical relationship with inheritance (“is-a” relationship). Consider, for example, the hierarchy of students in an institution wherein the top class “student” has two sub-classes of “graduate student” and “undergraduate student”. These sub-classes can further have sub-sub classes and so forth. A true hierarchy ensures the mutual exclusivity property, i.e an entity can only belong to one class. Mutual exclusivity makes hierarchies easy to represent and understand; however, it cannot represent multiple inheritance relationships though. Hierarchy is also not suitable in situations when researchers have to include multiple and diverse criteria for differentiation. To define a hierarchical classification, it is mandatory to have good knowledge on the subject matter to be classified; the classes and differentiating criteria between classes must be well defined early on.

**Tree** [21] is similar to the hierarchy, however, there is no inheritance relationship between the classes of tree-based taxonomies. In this kind of classification structure, common types of relationships between classes are “part-whole”, “cause-effect” and
“process-product”. For example, a tree representing a whole-part relationship between a country, its provinces and cities. Tree and hierarchy share similar strengths and limitations.

**Paradigm** [21] leads to taxonomies with two-way hierarchical relationships between classes. The classes are described by a combination of two attributes at a time. For example, paradigm would be suitable if we have to also represent gender in the “student” hierarchy example above-mentioned. It can also be viewed as a two-dimensional matrix whose vertical and horizontal axes allow for the inclusion of two attributes of interest. This type of classification structure shares similar strengths and limitations with the hierarchy structure.

**Faceted analysis** [6, 21] leads to taxonomies whose subject matters are classified using multiple perspectives (facets). The basic principle in faceted analysis is that there are more than one perspectives to view and classify a complex entity. Each facet is independent and can have its own classes, which enable facet-based taxonomies to be easily adapted so they can evolve smoothly over time. In order to properly classify CASE tools, for example, multiple facets need to be considered. These facets may include supported platform(s), license type, SE area, web support etc. Table 4.1 depicts the application of these multiple facets to classify two hypothetical CASE tools. Faceted analysis is suitable for new and evolving fields, since it is not required to have the complete knowledge related to the selected subject matter to design a facet-based taxonomy. However, it can be challenging to define an initial set of facets. In addition, although it is possible to define relationship between facets, in most cases the facets are independent and have no meaningful relationship between each other.

Table 4.1: Faceted analysis example.

<table>
<thead>
<tr>
<th>Tool name</th>
<th>Platform(s)</th>
<th>License type</th>
<th>SE area</th>
<th>Web support</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tool 1</td>
<td>Linux</td>
<td>FOSS</td>
<td>Testing</td>
<td>Yes</td>
</tr>
<tr>
<td>Tool 2</td>
<td>Windows</td>
<td>Proprietary</td>
<td>Construction</td>
<td>No</td>
</tr>
</tbody>
</table>

### 4.2.6 Validation

Validation strengthens reliability and usefulness of taxonomies. Taxonomies can be validated in three ways:

- **Orthogonality demonstration** – The orthogonality of the taxonomy dimensions and categories is demonstrated [15, 29].
• **Benchmarking** – The taxonomy is compared to similar classification schemes [29].

• **Utility demonstration** – The utility of a taxonomy is demonstrated by actually classifying subject matter examples [15, 29]. The utility of a taxonomy can be demonstrated or exemplified by classifying existing literature or expert opinion, or by employing more rigorous validation approaches such as a case study or experiment.

## 4.3 Research methodology

We chose the systematic mapping study method (SMS) to identify and analyze the state-of-the-art towards taxonomies in SE, because this method works well for broad and weakly defined research areas [26, 27]. We employed the guidelines by Kitchenham and Charters [26] and partly implemented the mapping process provided by Petersen *et al.* [27]. The employed mapping process is summarized in Figure 4.1 and described further in Subsections 4.3.1–4.3.5.

Figure 4.1: Employed systematic mapping process.

### 4.3.1 Research questions

The following research questions were formulated to guide this SMS:

- **Question 1 (RQ1)** – What taxonomy definitions and purposes are provided by publications on SE taxonomies?
- **Question 2 (RQ2)** – Which subject matters are classified in SE taxonomies?
- **Question 3 (RQ3)** – How is the utility of SE taxonomies demonstrated?
- **Question 4 (RQ4)** – How are SE taxonomies structured?
- **Question 5 (RQ5)** – To what extent are SE taxonomies used?
- **Question 6 (RQ6)** – How are SE taxonomies developed?

The main idea behind RQ1 is to identify how and why the term “taxonomy” is used in primary studies that claim to present a taxonomy. RQ2 focuses on identifying the subject matters classified by means of taxonomies in SE. RQ3 focuses on identifying the approaches used to demonstrate the utility of SE taxonomies, which is one of the ways of validating a taxonomy (see Section 4.2). With RQ4 we intend to identify the classification structures, related descriptive bases and classification procedures employed to design SE taxonomies. RQ5 focuses on the extent to which proposed SE taxonomies are used. Finally, RQ6 addresses in which ways SE taxonomies are developed, i.e. whether there are guidelines, methods, and processes that guide the development of taxonomies in a systematic way.

### 4.3.2 Search process

The search process employed in this work is displayed in Figure 4.2 and has 6 activities.

![Search process diagram](image)

**Figure 4.2: Search process.**

First, we defined the terms to be included in our search string. We selected all SWEBOK knowledge areas [28] to be included as terms, except for the three knowledge areas on related disciplines (Computing Foundations, Mathematical Foundations and Engineering Foundations). We also included the term “Software Engineering”, to augment the comprehensiveness of the search string. Finally, to reduce the scope of the search string to studies that report SE taxonomies, we included the term “taxonomy”.

95
Since some of the knowledge areas are referred by the SE community through other terms (synonyms), we also included their synonyms. Specifically, the following synonyms were included into the search string:

- **Requirements** – requirements engineering.
- **Construction** – software development.
- **Design** – software architecture.
- **Management** – software project management, software management.
- **Process** – software process, software life cycle.
- **Models and methods** – software model, software methods.
- **Economics** – software economics.

The selected SWEBOK knowledge areas and the term “Software Engineering” were all linked using the operator OR. The term “taxonomy” was linked with the other terms using the operator AND. The final search string is shown below.

```
(“software requirements” OR “requirements engineering” OR “software design” OR “software architecture” OR “software construction” OR “software development” OR “software testing” OR “software maintenance” OR “software configuration management” OR “software engineering management” OR “software project management” OR “software management” OR “software engineering process” OR “software process” OR “software life cycle” OR “software engineering models and methods” OR “software model” OR “software methods” OR “software quality” OR “software engineering professional practice” OR “software engineering economics” OR “software economics” OR “software engineering”) AND (taxonomy OR taxonomies)
```

Although SE knowledge classification could be named in different ways, e.g., taxonomy, ontology, classification and classification scheme [22], we limited the scope
of this study to taxonomies. Extending our search string to include the terms “ontology” and “classification scheme” would have led to an excessive number of search results that would have been infeasible to handle. Using alternative terms would also force the authors to interpret whether the primary studies’ authors actually intended to present a taxonomy when they do not explicitly refer to taxonomies. To mitigate this threat to validity, we restricted the scope to taxonomies.

Once the search string was designed, we selected the primary sources to search for relevant studies. Scopus, Compendex/Inspec and Web of Science were selected because they cover most of the important SE databases, such as IEEE, Springer, ACM and Elsevier. In addition, the selected primary sources are able to handle advanced queries. The search string was applied on metadata (i.e. title, abstract and author keywords) in August 2014 on the selected data sources. We later on updated the search results by applying the search string again in February 2016, to fetch studies published between September 2014 and December 2015. Table 4.2 presents the number of search results for each data source.

Table 4.2: Summary of search results

<table>
<thead>
<tr>
<th>Database/Search Engine</th>
<th>Search Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scopus</td>
<td>932</td>
</tr>
<tr>
<td>Compendex and Inspec</td>
<td>683</td>
</tr>
<tr>
<td>Web of Science</td>
<td>335</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>1950</strong></td>
</tr>
<tr>
<td><strong>Total without duplicates</strong></td>
<td><strong>1517</strong></td>
</tr>
</tbody>
</table>

4.3.3 Study selection process

The selection process employed in this work is displayed in Figure 4.3 and detailed as follows.

First, the following inclusion and exclusion criteria were defined:

- **Inclusion criteria**

4Inclusion of the terms “ontolog*” and “classification” returned 10474 hits in total just for Scopus.

5www.scopus.com

6www.engineeringvillage.com

7apps.webofknowledge.com
Chapter 4. Taxonomies in Software Engineering: A Systematic Mapping Study and a Revised Taxonomy Development Method

Figure 4.3: Selection process.

Figure 4.4: Extraction process.
1. Studies that propose or extend a taxonomy AND
2. Studies that are within Software Engineering (SE), according to SWE-BOK’s KAs (see Subsection 4.3.2).

- **Exclusion criteria**

1. Studies where the full-text is not accessible OR;
2. Studies that do not propose or extend a SE taxonomy OR;
3. Studies that are not written in English OR;
4. Studies that are not reported in a peer-reviewed workshop, conference, or journal.

The selection of primary studies was conducted using a two-stage screening procedure. In the first stage, only the abstracts and titles of the studies were considered. In the second stage, the full texts were read. Note that we used in both stages an inclusive approach to avoid premature exclusion of studies, i.e. if there was doubt about a study, such a study was to be included.

For the first stage (level-1 screening), the total number of 1517 studies were equally divided between the two first authors. As a result, 507 studies were judged as potentially relevant.

<table>
<thead>
<tr>
<th>Reason</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not proposing or evolving a SE taxonomy</td>
<td>1167</td>
</tr>
<tr>
<td>Duplicate</td>
<td>433</td>
</tr>
<tr>
<td>Full-text not accessible</td>
<td>50</td>
</tr>
<tr>
<td>Non-peer reviewed</td>
<td>21</td>
</tr>
<tr>
<td>Not written in English</td>
<td>9</td>
</tr>
<tr>
<td><strong>Total retrieved</strong></td>
<td>1950</td>
</tr>
<tr>
<td><strong>Total excluded</strong></td>
<td>1680</td>
</tr>
<tr>
<td><strong>Total included after study selection</strong></td>
<td>280</td>
</tr>
<tr>
<td><strong>Total included after data extraction</strong></td>
<td>270</td>
</tr>
</tbody>
</table>

To increase the reliability of the level-1 screening result, the third author screened a random sample of 10.30% (78 studies) from the studies screened by the first author and the fourth author screened a random sample of 10.28% (78 studies) from the studies
screened by the second author. The first and third authors had the same judgment for 91% (71) of the studies. The second and fourth authors had the same judgment for 93.6% (73) of the studies.

To evaluate the reliability of the inter-rate agreement between the authors, we calculated the Cohen’s kappa coefficient [17]. The Cohen’s kappa coefficient between the first and third authors was statistically significant (significance level = 0.05) and equal to 0.801. The Cohen’s kappa coefficient between the second and fourth authors was also statistically significant (significance level = 0.05) and equal to 0.857. According to Fleiss et al. [17], Cohen’s kappa coefficient values above 0.75 mean excellent level of agreement.

The level-2 screening (second stage), performed by the first and second authors, consisted on applying the selection criteria on the full-text of the studies selected during the level-1 screening. The total number of 507 studies were equally divided between the first two authors. As a result, 280 studies were judged as relevant.

To increase the reliability of the level-2 screening, a two-step validation was performed, as follows:

1. The first author screened 27.67% (70) of the studies deemed as relevant by the second author during the level-2 screening (randomly selected) and vice-versa. No disagreements were found between the authors.

2. Nine studies were randomly selected from each of the two sets allocated to the first two authors for further validation. The third author applied the study selection process on these 18 studies (about 6.43% of 280) for validation purposes. No disagreements were found with respect to the study selection (i.e. include/exclude) decisions.

During the entire screening process (stages 1 and 2), we tracked the reason for each exclusion, as presented in Table 4.3.

### 4.3.4 Extraction process

The extraction process employed in this work is summarized in Figure 4.4 and consists of four main steps: Define a classification scheme, define an extraction form, extract data, and validate the extracted data.

We designed classification scheme by following Petersen et al.’s guidelines [27]. It has the following facets:

- **Research type** – This facet is used to distinguish between different types of studies (adapted from Wieringa et al. [16]).
Table 4.4: Data extraction form

<table>
<thead>
<tr>
<th>Data item(s)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Citation data</td>
<td>Title, author(s), year and publication venue</td>
</tr>
<tr>
<td>Taxonomy definition</td>
<td>Definition of taxonomy that is used or referred to</td>
</tr>
<tr>
<td>Purpose</td>
<td>Text that states the purpose for the taxonomy</td>
</tr>
<tr>
<td>Purpose keyword</td>
<td>Key word used in the paper to describe the purpose (e.g. classify, understand, describe)</td>
</tr>
<tr>
<td>Subject matter</td>
<td>The name of the thing/concept are that is taxonomized</td>
</tr>
<tr>
<td>Descriptive bases</td>
<td>Is the subject matter defined in sufficient detail/clarity to enable classification (Yes/No)</td>
</tr>
<tr>
<td>Classification structure</td>
<td>Hierarchy, tree, paradigm, or faceted analysis, according to Kwasnik [21]</td>
</tr>
<tr>
<td>Classification procedure</td>
<td>The criteria for putting items in different classes (qualitative, quantitative or no details provided)</td>
</tr>
<tr>
<td>Classification procedure description</td>
<td>Do the authors explicitly describe the classification procedure (Yes/No)</td>
</tr>
<tr>
<td>Design method</td>
<td>Did the authors employ any systematic approach to design the reported taxonomy? If so, which approach?</td>
</tr>
<tr>
<td>Presentation approach</td>
<td>Textual or graphical</td>
</tr>
<tr>
<td>Utility demonstration</td>
<td>Is the utility of the taxonomy demonstrated? If so, how (e.g. illustration, case study, experiment)?</td>
</tr>
<tr>
<td>Primary knowledge area</td>
<td>Primary knowledge area as per SWEBOK v3 [28]</td>
</tr>
<tr>
<td>Secondary knowledge area</td>
<td>Secondary knowledge area as per SWEBOK v3 (if applicable)</td>
</tr>
<tr>
<td>Number of citations</td>
<td>Number of times a primary study is cited by other studies, as per Google Scholar</td>
</tr>
</tbody>
</table>

- **Evaluation research** – A study that reports a taxonomy implemented in practice, i.e. evaluation in a real environment, in general by means of the case study method.
- **Validation research** – A study that reports a taxonomy that was not implemented in practice yet, although it was validated in laboratory environment, in general by means of experiment.
- **Solution proposal** – A study that reports a taxonomy that was neither implemented in practice nor validated although it is supported by a small example (illustration) or a good line of argumentation.
• **SE knowledge area** – This facet is used to distinguish between the SE knowledge areas in which taxonomies have been proposed. The categories of this facet follow the SWEBOK [28]: software requirements, software design, software construction, software testing, software maintenance, software configuration management, software engineering management, software engineering process, software engineering models and methods, software quality, software engineering professional practice and software engineering economics.

• **Presentation approach** – This facet is used to classify the studies according to the overall approach used to present a taxonomy: textual and graphical, respectively.

For the data extraction, the relevant studies (280) were equally divided between the first and second authors. For each paper, data was collected and later on stored in a spreadsheet using the data extraction form shown in Table 4.4.

To increase the reliability of the extracted data, a two-step validation was performed, as follows:

1. The first author independently re-extracted the data of 50% (70) of the studies originally extracted by the second author (randomly selected) and vice-versa. Five disagreements were identified and all of them were related to the item “classification structure”.

2. Eighteen studies were randomly selected from the studies originally extracted by the first and second authors (9 studies from each author). Those studies were independently re-extracted by the third author. Twenty three disagreements were identified; 2 on the “taxonomy purpose”, 10 on “classification structure”, 2 on “classification procedure type”, 3 on “classification procedure description” and 6 on “validation approach”.

All disagreements except for “classification structure” were easily resolved. We believe that the high level of disagreement on the item “classification structure” was due to the fact that none of the studies explicitly stated and motivated the employed classification structure, which demanded the inference of such data from the text in each paper.

To improve the reliability of the extracted data, we decided to re-screen all 280 papers, focusing only on the item “classification structure”. First, we discussed classification structures in detail (based on Kwasnik [21]) to come to a common understanding.
of the terms. Second, three of us did an independent re-assessment of the classification structure of 52 papers. As a result, we reached full agreement on 50 papers (3 identical results) and partial agreement on 2 papers (2 reviewers agreeing). There were no primary studies without full or partial agreement. Third, the remaining 228 studies were re-assessed by the first and second authors and they reached agreement on 216 papers. The remaining 12 papers were independently re-assessed by the third author, who did not know the results from the other two reviewers. In the end, full agreement was achieved for 50 studies and partial agreement was achieved for 230 studies.

During the re-assessment of the primary studies, 10 studies were excluded because they do not present taxonomies, reducing the final number of primary studies to 270<sup>8</sup> (see Table 4.3).

### 4.3.5 Analysis process

Figure 4.5 presents the analysis process conducted herein. First, we classified the extracted data using the scheme defined in Subsection 4.3.4. This led to the results detailed in Section 4.4. We also performed a quantitative analysis of the extracted data to answer the research questions of this study. Finally, the overall result of the data analysis (see Section 4.4), along with information from additional literature ([14, 15, 21]), was used to revise an existing method previously proposed to design SE taxonomies [10], as detailed in Section 4.5.

### 4.4 Results

In this section, we describe the results of the mapping study reported herein, which are based on the data extracted from 270 papers reporting 271 taxonomies (one paper presented two taxonomies). The percentages in Sections 4.4.1 and 4.4.7 reflect the number of papers (270), whereas the percentages in all other subsections reflect the number of taxonomies (271).

#### 4.4.1 General results

Figure 4.6 shows that SE taxonomies have been proposed since 1987, with an increasing number of these published after the year 2000, which suggests a higher interest in this research topic.

<sup>8</sup>The full list of the included 270 primary studies is available at [http://tinyurl.com/jrdaxhh](http://tinyurl.com/jrdaxhh).
Table 4.5 displays that 53.7% (145) of the studies were published in relevant conferences in the topics of maintenance (International Conference on Software Maintenance), requirements engineering (Requirements’ Engineering Conference) or general SE topics (e.g. International Conference on Software Engineering). Taxonomies were published at 99 unique conferences with 78 featuring only a single SE taxonomy publication. These results further indicate a broad interest in SE taxonomies in a wide range of SE knowledge areas.

Table 4.5 also shows that 33.7% (91) of the primary studies were published as journal articles in 44 unique journals. Taxonomies have been published frequently in relevant SE journals (e.g. IEEE Transactions on Software Engineering and Information and Software Technology). We believe that this has been the case because the scope of these journals is not confined to a specific SE knowledge area.

Primary studies were published also in 28 unique workshops (34 – 12.6%). As for journals and conferences, the results indicate an increasing interest in SE taxonomies in a broad range of SE knowledge areas.

Figures 4.7a–h depict the yearly distribution of SE taxonomies by knowledge area for the KAs with 10 or more taxonomies. Note that most knowledge areas follow an

---

Figure 4.5: Analysis process.
Figure 4.6: Year and venue wise distributions.
increasing trend after 2000, with many taxonomies for construction, design, and quality in the 1980s and 1990s.

### 4.4.2 Classification scheme results

In this section, we present the results corresponding to the three facets of the classification scheme described in Section 4.3, i.e. SE knowledge area (KA), research type and presentation approach.

The vertical axis in Figure 4.8 depicts the **SE knowledge areas** in which taxonomies have been proposed. Construction and design are the leading SE knowledge areas.
Figure 4.7: Yearly distribution of primary studies by KAs. Horizontal axes represent the years (starting 1987), while vertical axes denote the number of taxonomies.
areas each with 53 (19.55%) taxonomies. These are relatively mature SE fields with a large body of knowledge and a high number of subareas.

A high number of taxonomies have also been proposed in the requirements (42 – 15.50%), maintenance (32 – 11.81%) and testing (27 – 9.96%) knowledge areas. Few taxonomies have been proposed in economics (3 – 1.11%) and professional practice (3 – 1.11%), which are more recent knowledge areas.

The results show that most SE taxonomies (76.37%) are proposed in the requirements, design, construction, testing and maintenance knowledge areas, which correspond to the main activities in a typical software development process [32].

The horizontal axis in Figure 4.8 shows the distribution of taxonomies by research types, according to Wieringa et al. [16]. Most taxonomies are reported in papers that are classified as “solution proposals” (135 – 49.82%), wherein the authors propose a taxonomy and explain or apply it with the help of an illustration. Ninety one taxonomies (33.58%) are reported in “philosophical papers”, wherein authors propose a taxonomy, but do not provide any kind of validation, evaluation or illustration. Relatively fewer taxonomies are reported in “evaluation papers” (34 – 12.54%) and “validation papers” (11 – 4.06%).

Figure 4.8 also depicts the classification of the taxonomies using 2 aspects of the classification scheme, i.e. SE knowledge area and research type.

Taxonomies in the knowledge areas construction and design are mostly reported either as solution proposals (construction – 27; design – 31) or philosophical papers (construction – 20; design – 17). Taxonomies in the knowledge areas requirements, maintenance and testing are better distributed across different research types, wherein besides the solution proposal and the philosophical research types, a reasonable percentage of taxonomies are reported as evaluation or validation papers.

The horizontal axis in Figure 4.9 shows the distribution of taxonomies by presentation approach. Most taxonomies (57.93%) are presented purely as text or table, while 42.07% of the taxonomies are presented through some graphical notation in combination with text.

Figure 4.9 also displays the classification of the identified taxonomies in terms of SE knowledge area and presentation approach. The results show 2 different trends:

- For knowledge areas such as design, quality, models and methods, and process, both textual and graphical approaches are used an almost equal number of times. This suggests that the taxonomies in the KAs that involve a lot of modeling might be better presented using graphical modeling approaches.

- Most taxonomies in construction (35 out of 53), maintenance (23 out of 32), testing (15 out of 27) and software management (7 out of 10) are textually presented.
Figure 4.8: Systematic map – knowledge area vs research type.
Figure 4.9: Systematic map – knowledge area vs presentation type.
4.4.3 RQ1 – Taxonomy understanding

We extracted data about the following two aspects to answer RQ1:

- **Taxonomy definition**: We investigated from each study whether or not the authors made any attempt to communicate their understanding about the concept of taxonomy by citing or presenting any definition of it.

- **Taxonomy purpose**: We identified from each study the stated (if any) main purpose for designing a taxonomy.

As stated earlier, taxonomy is not a trivial concept. It has been defined in multiple ways (see Section 4.2 for some definitions). This RQ aims to identify whether authors make an explicit effort to share their perspective on taxonomy by adopting/using a definition. The results show that only 6.3% (17) of the taxonomies were reported with a definition for the term “taxonomy”. Out of these 17 taxonomies, three use the Cambridge dictionary’s definition (see Section 4.2), eight studies do not provide an explicit source and the remaining six have other unique references: The American heritage dictionary, Carl Linnaeus [19], Whatis, the IEEE standard taxonomy for SE standards, Doty and Glick [33] and Fleishman et al. [34]. For the remaining 93.7% (254) taxonomies, no definition of “taxonomy” was provided.

To identify the purpose of each taxonomy, we extracted the relevant text, referred here as purpose descriptions, from each of the primary studies, using a process similar to open coding [23, 31]. As codes we used the keywords used in the primary studies to describe a taxonomy’s purpose.

For about 56% of the taxonomies, the authors used “classify” (48.80%) or “categorize” (7.74%) to describe the purpose of their taxonomy. For 5.9% of the taxonomies, it was not possible to identify a specific purpose. For the remaining taxonomies (38.37%), we found 41 different terms for describing the purpose, e.g., “identify”, “understand”, and “describe”.

4.4.4 RQ2 – Subject matters

In total, we identified 263 unique subject matters for the 271 taxonomies, e.g., technical debt, architectural constraints, usability requirements, testing techniques and process models.

The high number of unique subject matters means that almost each taxonomy dealt with a unique subject matter. This might be due to the following reasons:

---

9www.ahdictionary.com/
10www.whatis.com
11For full list see: http://tinyurl.com/z4mqfnr
Existing taxonomies do not fit their purpose well. Therefore there is a need to define new taxonomies.

The subject matters for existing taxonomies are so narrowly defined that they are not suitable for usage outside their original context. New taxonomies are therefore developed constantly.

SE researchers do not reuse or extend existing taxonomies when there is need for organizing SE knowledge, but rather propose new ones.

One indicator for taxonomy use is the number of times each primary study is cited. This analysis is detailed in Subsection 4.4.7.

The list of subject matters contains mainly technical aspects of SE. Only few taxonomies deal with people-related subject matters, e.g., stakeholder-related and privacy-related issues.

The results for this research question suggest that taxonomies are rarely revisited, revised or extended. However, many taxonomy papers are highly cited, which shows that there is a strong interest in taxonomies in the SE field.

The mapping of SE taxonomies presented in this study supports SE researchers in identifying and evolving existing taxonomies. This may lead to the development of a more consistent terminology.

### 4.4.5 RQ3 – Utility demonstration

Table 4.6 displays the approaches used to demonstrate the utility of SE taxonomies. Illustration is the most frequently used approach (124 – 45.76%). Illustration includes approaches such as example, scenario and case.

Case studies have also been used to demonstrate the utility of 34 taxonomies (12.54%). Experiments have been used to demonstrate the utility of 11 taxonomies (4.06%), while the utility of a few taxonomies have also been demonstrated through expert opinion (7 – 2.58%) or survey (4 – 1.48%). Note that 33.9% (83) of the taxonomies did not have their utility demonstrated.

The results related to RQ3 show that a few taxonomies have their utility demonstrated through methods like case study or experiment, while the utility of a large number of taxonomies (33.58%) is not demonstrated by any means. We do not believe that one particular approach would be the best for all contexts; however we believe that in most cases would not be enough just to propose a taxonomy.
Table 4.6: Approaches for utility demonstration.

<table>
<thead>
<tr>
<th>Approach</th>
<th>F</th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Illustration</td>
<td>124</td>
<td>45.76</td>
</tr>
<tr>
<td>Case study</td>
<td>34</td>
<td>12.54</td>
</tr>
<tr>
<td>Experiment</td>
<td>11</td>
<td>4.06</td>
</tr>
<tr>
<td>Expert opinion</td>
<td>7</td>
<td>2.58</td>
</tr>
<tr>
<td>Survey</td>
<td>4</td>
<td>1.48</td>
</tr>
<tr>
<td>No Validation</td>
<td>91</td>
<td>33.58</td>
</tr>
</tbody>
</table>

4.4.6 RQ4 – Taxonomy structure

To answer RQ4, the following data was gathered: classification structure, descriptive bases, classification procedure and classification procedure description.

Table 4.7 shows the classification structures identified for the identified taxonomies. Hierarchy was the most frequently used classification structure (144 – 53.14%), followed by faceted-based structures (107 – 39.48%), tree (14 – 5.17%) and paradigm (6 – 2.21%).

Table 4.7: Classification structure.

<table>
<thead>
<tr>
<th>Classification structure</th>
<th>F</th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hierarchy</td>
<td>144</td>
<td>53.14</td>
</tr>
<tr>
<td>Faceted analysis</td>
<td>107</td>
<td>39.48</td>
</tr>
<tr>
<td>Tree</td>
<td>14</td>
<td>5.17</td>
</tr>
<tr>
<td>Paradigm</td>
<td>6</td>
<td>2.21</td>
</tr>
</tbody>
</table>

Table 4.8 displays the status of the taxonomies’ descriptive basis. The majority of the taxonomies have a sufficiently clear description of their elements (248 – 91.51%), followed by only 22 taxonomies (8.49%) without a sufficient description.

Table 4.9 presents the classification procedure types for the identified taxonomies. The majority of the taxonomies employed a qualitative classification procedure (262 – 96.68%), followed by quantitative (7 – 2.58%) and both (2 – 0.74%).

Table 4.10 displays the status of the taxonomies’ classification procedure description. The majority of the taxonomies do not have an explicit description for the
Chapter 4. Taxonomies in Software Engineering: A Systematic Mapping Study and a Revised Taxonomy Development Method

Table 4.8: Descriptive bases.

<table>
<thead>
<tr>
<th>Descriptive basis</th>
<th>F</th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sufficiently clear description</td>
<td>248</td>
<td>91.51</td>
</tr>
<tr>
<td>Without sufficient description</td>
<td>23</td>
<td>8.49</td>
</tr>
</tbody>
</table>

Table 4.9: Classification procedure types.

<table>
<thead>
<tr>
<th>Classification procedure</th>
<th>F</th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Qualitative</td>
<td>262</td>
<td>96.68</td>
</tr>
<tr>
<td>Quantitative</td>
<td>7</td>
<td>2.58</td>
</tr>
<tr>
<td>Both</td>
<td>2</td>
<td>0.74</td>
</tr>
</tbody>
</table>

classification procedure (227 – 83.76%) and only 44 taxonomies (16.24%) have an explicit description.

Table 4.10: Classification procedure descriptions.

<table>
<thead>
<tr>
<th>Procedure description</th>
<th>F</th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not explicitly described</td>
<td>227</td>
<td>83.76</td>
</tr>
<tr>
<td>Explicitly described</td>
<td>44</td>
<td>16.24</td>
</tr>
</tbody>
</table>

The results clearly indicate that hierarchical classification structures (hierarchy, tree and paradigm) are the ones mostly used to design SE taxonomies. This is not surprising, since taxonomies were originally designed using hierarchy as classification structure (see Introduction).

Nevertheless, SE is considered as a young discipline that is in constant and fast evolution. For this reason, the existing knowledge is sometimes incomplete or unstable. This is probably the reason faceted analysis is also frequently employed as classification structure to design SE taxonomies; such classification structure is the one that is most appropriate for knowledge fields with incomplete and unstable knowledge [21]. It was interesting to observe that in almost all studies the choice to employ a specific
classification structure is not explicitly motivated.

The descriptive bases are mandatory for a reasonable understanding of a taxonomy (see Section 4.2). The absence of such element can hinder the adoption of taxonomies. Not surprisingly, the 23 studies that did not provide descriptive bases are lowly cited, it might indicate the low usefulness of them.

At first glance, it appears that the main reason qualitative classification procedures are more popular is the fact that such approach is easier to be designed; the classification using such procedures is performed using nominal scales. However, such approach is harder to be applied, because it can leave room for ambiguities; it is not possible to establish the differences/similarities between classes because distance functions cannot be used to calculate the difference/similarity degree between classes [15]. Quantitative approaches are harder to be designed, but they are easier to be applied to classify subjects, since it is possible to use distance functions to identify the differences/similarities between classes [15].

To enable accurate subject classifications, the classification procedure of a taxonomy must be described sufficiently. Only 16.24% of the taxonomies (44) have an explicit description for their classification procedure. The remaining 83.76% of the taxonomies (227) are probably harder to be used by other people but the designers of the taxonomies themselves.

4.4.7 RQ5 – Taxonomy use

As an approximation for the usage level of SE taxonomies, we looked at the number of citations for each included primary study\textsuperscript{12}. Since a publication may be cited due to many different reasons, the number of citations can only be an indication of actual use of the proposed taxonomy. However, the number of citations show whether there is an interest in a subject.

As displayed by Table 4.11, 61 primary studies (22.6%) have been cited \(\geq 50\) times. There are only 26 studies that were never cited. Out of these 26 primary studies, 17 were published between 2012 and 2015.

Figure 4.10 shows the distribution of average number of citations per year for the primary studies\textsuperscript{13}. Most studies were cited 1–5 times per year on average (126 – 46.7%).

Table 4.12 lists the mean and median number of citations for the knowledge areas with at least 10 taxonomies. The data shows that for all knowledge areas, except for process, the mean value is much higher than the median, which is due to few studies

\textsuperscript{12}The number of citations for each primary study was fetched from Google Scholar during Jan-Feb 2015.

\textsuperscript{13}Average is computed by dividing citation count of a study by number of years from publication date to 2015.
Table 4.11: Number of primary studies (Frequency) by number of citations.

<table>
<thead>
<tr>
<th>Citations</th>
<th>F</th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>≥ 300</td>
<td>8</td>
<td>3.0</td>
</tr>
<tr>
<td>≥ 200</td>
<td>16</td>
<td>5.9</td>
</tr>
<tr>
<td>≥ 100</td>
<td>30</td>
<td>11.1</td>
</tr>
<tr>
<td>≥ 50</td>
<td>61</td>
<td>22.6</td>
</tr>
<tr>
<td>≥ 20</td>
<td>105</td>
<td>38.9</td>
</tr>
<tr>
<td>≥ 10</td>
<td>143</td>
<td>53.0</td>
</tr>
<tr>
<td>≥ 1</td>
<td>244</td>
<td>90.4</td>
</tr>
<tr>
<td>0</td>
<td>26</td>
<td>9.6</td>
</tr>
</tbody>
</table>

with very high numbers of citations (outliers); e.g. construction has a study with 1440 citations. Maintenance has fewer taxonomies (32) as compared to construction (53) and design (53), but it has the highest median for the number of citations.

Figure 4.10: Distribution of average number of citations per year. The x-axis shows ranges of average citations per year. The y-axis shows the number of primary studies.

The results show that most studies reporting taxonomies are cited a reasonable number of times with many primary studies that are highly cited in all knowledge areas. This indicates an interest in taxonomies by the SE community. The high number of unique subject matters (see Subsection 4.4.4) might be due to the breadth of the SE field and a need for classifying new and evolving SE subareas.
We also observed that some recently proposed taxonomies (2012–2015) were already cited many times. These results highlight that new taxonomies are proposed continuously and cited frequently. This indicates that the classification of SE knowledge through taxonomies is a relevant topic.

A more thorough analysis of the actual usage of the cited primary studies would be necessary though, to understand the contexts and reasons for taxonomy usage.

Table 4.12: Number of citations (mean and median) for primary papers by knowledge area.

<table>
<thead>
<tr>
<th>Knowledge area</th>
<th>Mean</th>
<th>Median</th>
</tr>
</thead>
<tbody>
<tr>
<td>Construction</td>
<td>92.7</td>
<td>17</td>
</tr>
<tr>
<td>Design</td>
<td>41.2</td>
<td>9</td>
</tr>
<tr>
<td>Requirements</td>
<td>38.1</td>
<td>10.5</td>
</tr>
<tr>
<td>Maintenance</td>
<td>40.9</td>
<td>21</td>
</tr>
<tr>
<td>Testing</td>
<td>28.8</td>
<td>8</td>
</tr>
<tr>
<td>Quality</td>
<td>46.6</td>
<td>14.5</td>
</tr>
<tr>
<td>Models &amp; Methods</td>
<td>68.2</td>
<td>9</td>
</tr>
<tr>
<td>Process</td>
<td>10.1</td>
<td>5</td>
</tr>
</tbody>
</table>

### 4.4.8 RQ6 – Taxonomy development

To address RQ6, we investigated whether the development process of a taxonomy was described or explained in some way. The majority of SE taxonomies have been developed in an ad-hoc manner. Bayona-Oré et al. [10], however, described a systematic approach to develop their taxonomy. However, the method proposed by them has some issues and thus can be enhanced. Therefore, in the rest of this section we describe Bayona-Oré et al.’s taxonomy design method and its limitations.

**Bayona-Oré et al.’s method**

Bayona-Oré et al. proposed a taxonomy development method that was used to create a taxonomy of critical success factors for software process deployment. They have developed the method based on a literature review of methods and guidelines used for developing taxonomies. Each author identified, reviewed and analyzed the activities included in the reviewed literature.
Chapter 4. Taxonomies in Software Engineering: A Systematic Mapping Study and a Revised Taxonomy Development Method

The identified activities were aggregated according to their similarities. As a result, the authors organized the activities in five groups, which are considered as the phases of the proposed method. The resulting method consists of five phases and 24 activities. The phases are described as follows (the activities are presented in Table 4.13):

- **Planning** - In this phase, plans are established to define the activities for the design and implementation of the taxonomy. It has eight activities.

- **Identification and extraction** - In this phase, the work plan and the information required by the organization are aligned. It has two activities.

- **Design and construction** - In this phase, the taxonomy is designed and constructed using the inventory of terms in order to determine the final structure of the taxonomy. It has six activities.

- **Testing and validation** - In this phase, it is ensured that the designed taxonomy will be useful for users to achieve their goals. It has two activities.

- **Deployment** - In the final phase, the taxonomy is deployed throughout the organization. It has six activities.

The activities of the method are to be carried out in sequence. Furthermore, the method is built upon the assumption that the taxonomy to be developed has hierarchical-based classification structure.

Bayona-Oré *et al.*’s method issues

Based on literature from more mature fields (e.g., [14, 15, 21]), the lessons learned during the conduction of the mapping study, and our previous experience with designing SE taxonomies/classification schemes [2, 3, 7, 8], we identified the following issues with Bayona-Oré *et al.*’s method:

- **Issue 1** – Six out of nine sources on which Bayona-Oré *et al.*’s method is based are either gray literature or are not available in English.

- **Issue 2** – The proposed method is a simple aggregation of all the activities suggested in the nine sources, thus some of these activities are only used in one of the peer-reviewed sources.

- **Issue 3** – The scope of some of the suggested activities goes beyond taxonomy design, as they cover steps related to project planning, recruitment, training and deployment, which are not necessarily relevant when designing taxonomies.
• **Issue 4** – The method does not have activities to deal with classification structures, i.e. it is only able to deal with hierarchical-based classification structures. As per the results of our mapping study, 39.48% of the identified taxonomies have a facet-based classification structure, which suggests the need for a method that allows for designing facet-based taxonomies.

• **Issue 5** – The method does not have activities to deal with classification procedures. As per existing literature [15], it is mandatory to define clear classification criteria, which can be better done if a method is used to systematize the classification procedure design.

We addressed the aforementioned issues by revising Bayona-Oré et al.’s method. The revised method is described in Section 4.5.

4.4.9 **Threats to validity related to the mapping study**

The following three major validity threats were identified and mitigated in relation to the systematic mapping study:

- **Coverage of the search string** – This threat refers to the effectiveness of the applied search string to find a sufficiently large number of relevant primary studies. To mitigate this threat, the search string was designed to be as comprehensive as possible. To achieve this goal, we not only included the SWEBOK knowledge areas related to SE; but also main synonyms of the included knowledge areas. The search string still cannot be considered complete as we did not include an exhaustive list of synonyms for all knowledge areas. The search string returned a high number of primary studies (1517 unique studies). The accuracy of the search string was considered as fair, since 18.46% of the 1517 were included after the selection process. An additional mitigation action was to apply the search string on multiple databases. This mapping study includes only those works as primary studies whose authors explicitly claim their work as taxonomies. While this is a limitation, we decided to do so to avoid misclassifying studies as taxonomies when authors themselves did not name their classifications explicitly as taxonomies.

- **Study selection** – This threat refers to the likelihood of disagreements between reviewers regarding study inclusion or exclusion. The first mitigation strategy was to define clear selection criteria. The selection criteria were discussed by all the authors to ensure shared understanding of the criteria. Whenever a paper was excluded, a reviewer had to provide a reason for the exclusion. The second
mitigation action was to perform cross-validations in both level-1 and level-2 screenings, as described in Subsection 4.3.3.

- **Data extraction** – This threat refers to the possibility that different reviewers interpret and handle data extraction in different ways. This could lead to different data extracted from the same primary study, depending on the reviewer who carried out the data extraction. The first action to mitigate this threat was to design a spreadsheet to be used by all authors during the data extraction. The spreadsheet, as well as the definitions of the data that had to be extracted was discussed by the reviewers to ensure a common understanding. The second mitigation action was to perform a cross-validation of the extracted data, as detailed in Subsection 4.3.4. Cross-validation showed that the item “classification structure” was especially difficult to extract, since none of the papers provided that item explicitly. It had to be inferred from the text, which led to a high level of disagreement between reviewers. Thus, the third mitigation action, performed jointly by the first three authors, was to re-screen all 280 studies, focusing on the “classification structure”, as described in Subsection 4.3.4.

### 4.5 A revised taxonomy development method

As aforementioned, we identified some issues associated with Bayona-Oré et al.’s taxonomy design method. Thus, we propose the following changes to address the method’s limitations:

1. Exclude phases and activities that were out of the scope of taxonomy development (Issue 3).
2. Exclude activities related to Issues 1 and 2 that were not supported by the results of our mapping study and literature from other research fields.
3. Incorporate new activities to address Issues 4 and 5.
4. Analyze the phases and activities to identify additional aspects that could be further improved, such as the sequence of the activities and whether activities could be merged to facilitate the use of the method.

As a result, we revised Bayona-Oré et al.’s method, which resulted in a revised method that has 4 phases with 13 activities in total (the original method has five phases and 24 activities). A side-by-side comparison between the original and the revised versions of Bayona-Oré et al.’s method is shown in Table 4.13. A more detailed description of each activity of the revised method is provided in Subsections 4.5.1–4.5.1.
Table 4.13: Comparison between the original and the revised versions of Bayona-Oré et al.’s method.

<table>
<thead>
<tr>
<th>Phase</th>
<th>Activity according to Bayona-Oré et al. [10]</th>
<th>Revised activity</th>
<th>Rationale for change</th>
</tr>
</thead>
<tbody>
<tr>
<td>Planning</td>
<td>A1: Identify the area of the study</td>
<td>B1: Define SE knowledge area</td>
<td>Similar to A1</td>
</tr>
<tr>
<td></td>
<td>A2: Define the objectives of the taxonomy</td>
<td>B2: Describe the objectives of the taxonomy</td>
<td>Combination A2 and A4</td>
</tr>
<tr>
<td></td>
<td>A3: Develop a survey of user needs</td>
<td>—</td>
<td>Deleted due to Issue 3</td>
</tr>
<tr>
<td></td>
<td>A4: Define the scope of the proposed taxonomy</td>
<td>—</td>
<td>Part of B1 and B2</td>
</tr>
<tr>
<td></td>
<td>A5: Define the team in charge of developing the taxonomy</td>
<td>—</td>
<td>Deleted due to Issue 3</td>
</tr>
<tr>
<td></td>
<td>A6: Identify the required resources</td>
<td>—</td>
<td>Moved to next “phase”</td>
</tr>
<tr>
<td></td>
<td>A7: Document the plan</td>
<td>—</td>
<td>Deleted due to Issue 3</td>
</tr>
<tr>
<td></td>
<td>A8: Obtain commitment and support</td>
<td>—</td>
<td>Deleted due to Issue 3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>B3: Describe the subject matter to be classified</td>
<td>Based on results of this study, from Wheaton [15] and Glass and Vessey [14]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>B4: Select classification structure type</td>
<td>Based on results of this study, from Kwasnik [21] and Glass and Vessey [14]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>B5: Select classification procedure type</td>
<td>Added to address Issue 5.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>B6: Identify the sources of information</td>
<td>Same as A9; considered as a planning activity</td>
</tr>
<tr>
<td>Identification and extraction</td>
<td>A9: Identify the sources of information</td>
<td>—</td>
<td>Moved to planning phase</td>
</tr>
<tr>
<td></td>
<td>A10: Extract all terms and identify candidate categories</td>
<td>B7: Extract all terms</td>
<td>Partly corresponds to A10; categories’ identification is covered by B10</td>
</tr>
<tr>
<td></td>
<td></td>
<td>B8: Perform terminology control</td>
<td>Combines A11 and A13</td>
</tr>
<tr>
<td>Design and Construction</td>
<td>A11: Check the list of terms and define the criteria</td>
<td>—</td>
<td>Corresponds to B8</td>
</tr>
<tr>
<td></td>
<td>A12: Define the first level of the taxonomy design</td>
<td>—</td>
<td>Part of B9 and B10</td>
</tr>
<tr>
<td></td>
<td>A13: Perform terminology control</td>
<td>—</td>
<td>Moved to identification and extraction phase</td>
</tr>
<tr>
<td></td>
<td>A14: Define the subsequent levels of the taxonomy</td>
<td>—</td>
<td>Covered in B10</td>
</tr>
<tr>
<td></td>
<td>A15: Review and approve the taxonomy by stakeholders and experts</td>
<td>—</td>
<td>Deleted due to Issue 3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>B9: Identify and describe taxonomy dimensions</td>
<td>Added to address Issue 4. Based on results of this study, from Kwasnik [21] and Glass and Vessey [14]</td>
</tr>
</tbody>
</table>
4.5.1 Phases and activities of the revised method

Here we describe the phases and activities of the revised method. Note that we adjusted the description of the phases and activities to reflect the changes we made in the method.

Planning

The planning phase has 6 activities that define the taxonomy’s context and initial setting, i.e. SE knowledge area, objective, subject matter, classification structure type, classification procedure type and sources of information.

In activity B1, one selects and makes clear the SE knowledge area for the new taxonomy. This makes it easier to understand the context of the taxonomy and thus to apply it.
In activity B2, the objectives and scope of the taxonomy are defined clearly, so that it is applied within the intended boundaries.

In activity B3, the subject matter for the classification is described in detail. This is a fundamental step in taxonomy development as discussed in Sections 4.2 and 4.4.

In activity B4, an appropriate classification structure type is selected, which can be hierarchy, tree, paradigm or facet-based (see Sections 4.2 and 4.4).

In activity B5, one determines an appropriate classification procedure type, which can be qualitative, quantitative or both (see Sections 4.2 and 4.4).

In activity B6, the data sources and data collection methods are identified to facilitate the prospection of knowledge related to the subject matter and taxonomy. Examples of data collection methods are interviews, observation, archival research, survey and simulation [9].

**Identification and extraction**

This phase comprises 2 activities for extracting and controlling the terms associated with the taxonomy.

In activity B7, the terms relevant to the new taxonomy are extracted from the collected data.

As aforementioned, taxonomies are specially useful for organizing knowledge areas that have evolving and volatile knowledge. Very often such knowledge areas lack a common terminology; different terms can be used interchangeably or a single term is used to represent different things. In activity B8 such redundancies and inconsistencies are identified and removed.

**Design and construction**

Design and construction comprises 4 activities to support identification and description of the dimensions, categories and relationships of the new taxonomy. In addition, guidelines for using and evolving the taxonomy are provided.

In activity B9, top-level dimensions are identified and described. They represent the main perspectives or top categories under which subject matter entities are classified. In taxonomies that employ hierarchy or tree as classification structure, exactly one dimension is expected (the root of the tree or hierarchy). Taxonomies that employ paradigm as the classification structure must have exactly two dimensions. Taxonomies that employ a facet-based classification structure must have at least two dimensions.

Activity B10 identifies and describes the categories for each of the dimensions; each dimension must have at least two categories.
In activity B11, the relationships between dimensions and categories are identified and described clearly. Note that in some cases there is no relationship between dimensions, i.e. this activity might be skipped.

To facilitate the adoption and evolution of the taxonomy, guidelines should be provided, which is done in activity B12.

**Validation**

To ensure that the selected subject matter is clearly, concisely and thoroughly classified, it is necessary to validate the new taxonomy, which is done in activity B13. A taxonomy can be validated through orthogonality demonstration, benchmarking and utility demonstration (see Section 4.2).

### 4.5.2 Illustrating the revised method

In this chapter, to illustrate how the revised method can be used by SE researchers, we mapped an existing taxonomy into the activities of the revised method, showing the expected outcome of each activity. Chapter 5 presents the application of the revised method to develop a taxonomy of effort estimation for agile software development.

We selected Mendes et al.’s taxonomy [8] to illustrate our method. One of the designers of the selected taxonomy has participated in the study reported in this chapter as well, which enabled us to map it to our revised method with the required level of detail.

Mendes et al. have proposed a taxonomy of hypermedia and Web application size metrics. The taxonomy has been designed based on the results of a literature survey. The authors used the taxonomy to classify existing literature. The mapping between the revised method and Mendes et al.’s taxonomy is presented in Table 4.14.

In the planning phase, we mapped six activities (B1, B2, B3, B4, B5 and B6). The knowledge area associated with the selected taxonomy is software engineering project management (outcome of B1). The main objective of the proposed taxonomy is to define a set of categories that enable classification of the hypermedia and Web application size metrics reported in the existing literature (outcome of B2). The taxonomy was designed using a facet-based classification structure (outcome of B4), as there are multiple perspectives using which the web size metrics can be classified. The procedure used to classify the size metrics is qualitative in nature (outcome of B5), wherein the authors subjectively assign the size metrics to the relevant categories. Finally, the basis of the taxonomy consists of software measurement concepts drawn from literature in software size metrics and measurement (outcome of B6).
Table 4.14: An illustration of the revised method

<table>
<thead>
<tr>
<th>Phase</th>
<th>Id</th>
<th>Mendes et al.'s taxonomy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Planning</td>
<td>B1</td>
<td>The software engineering knowledge area associated to the designed taxonomy is software engineering project management.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The main objective of the proposed taxonomy is to define a set of categories that enables to classify hypermedia and Web application size metrics reported in the existing literature.</td>
</tr>
<tr>
<td></td>
<td>B2</td>
<td>The subject matter of the designed taxonomy is hypermedia and web application size metrics.</td>
</tr>
<tr>
<td></td>
<td>B3</td>
<td>The taxonomy was designed using a facet-based classification structure.</td>
</tr>
<tr>
<td></td>
<td>B4</td>
<td>The procedure used to classify the size metrics is qualitative.</td>
</tr>
<tr>
<td></td>
<td>B5</td>
<td>The basis of the taxonomy consists of software measurement concepts drawn from literature in software size metrics and measurement.</td>
</tr>
<tr>
<td>Identification and extraction</td>
<td>B7</td>
<td>All the dimensions and categories were extracted from the literature.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>It was not necessary to perform terminology control, because the dimensions and categories are the result of an aggregation of concepts from existing literature.</td>
</tr>
<tr>
<td></td>
<td>B8</td>
<td>The following eight dimensions were identified: harvesting time, metric foundation, class, entity, measurement scale, computation, validation and model dependency.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Twenty five categories were identified, as follows: Harvesting time (Early / Late); Metric foundation (Problem-oriented / Solution-oriented); Class (Length / Functionality / Complexity); Entity (Web hypermedia application / Web software application / Web application / Media / Program-script); Measurement scale (Nominal / Ordinal / Interval / Ratio / Absolute); Computation (Directly / Indirectly); Validation (Validated empirically / Validated theoretically / Both / None); and Model dependency (Specific / Nonspecific).</td>
</tr>
<tr>
<td>Design and construction</td>
<td>B9</td>
<td>The classification structure of the taxonomy is facet-based, the dimensions are orthogonal, i.e. they are isolated from each other. The categories of each dimension are also orthogonal.</td>
</tr>
<tr>
<td></td>
<td>B10</td>
<td>No guidelines were proposed by the authors, neither for using nor for evolving the taxonomy. However, the authors used the taxonomy to classify existing literature, which gives some guidance about how to use the taxonomy.</td>
</tr>
<tr>
<td>Validation</td>
<td>B13</td>
<td>The taxonomy was used to classify size metrics reported in the existing literature (utility demonstration). No benchmarking was carried out. The orthogonality of the dimensions and categories was ensured by design.</td>
</tr>
</tbody>
</table>

In the **identification and extraction phase**, we mapped two activities (B7 and B8). All the categories were extracted from existing literature (outcome of B7). The authors carried out terminology control to aggregate the concepts identified in existing literature (outcome of B8).

In the **design and construction phase**, we mapped four activities (B9, B10, B11 and B12). As a result of B9, eight dimensions were identified, and as result of B10, 25 categories were identified (see Table 4.14 for the identified dimensions and categories). Regarding B11, due to the classification structure employed (facet-based), there is no relationship between the dimensions. Finally, the authors did not carried out B12, i.e.
they did not designed any guidelines for using and evolving the taxonomy. Nevertheless, they have illustrated the taxonomy, which provides some guidance about how to use the proposed taxonomy. In the validation phase, they only demonstrated the utility of the proposed taxonomy by classifying size metrics reported in the existing literature (outcome of B13).

### 4.5.3 Limitations of the revised method

In relation to the revised method presented here, we identified the following limitations:

- It may be the case that there are other taxonomy design methods/processes that were not captured by us.

- The new activities introduced in our revised method were based on the lessons learned during the conduction of the mapping study and the literature from other research areas. We used the revised method to design a taxonomy of effort estimation for agile software development (see Chapter 5). However, we believe that it must be used by other researchers that have not participated in the study presented in this chapter to further improve the method.

### 4.6 Conclusion

In this chapter we reported the results of a systematic mapping study on taxonomies in SE discipline. The initial search returned 1517 unique results. The application of a two-phased study selection and validation processes resulted in the inclusion of a final set of 270 primary studies.

We observed a rising trend in publishing and using taxonomies in SE in recent years. Taxonomies have been published mostly in domain specific SE conferences (e.g. Requirements’ Engineering Conference) and highly reputed SE journals (e.g. IEEE Transactions in Software Engineering and Information and Software Technology). Regarding the research type, most studies were categorized as either solution proposal or philosophical paper.

Over 75% taxonomies are proposed in 5 SE KAs (requirements, design, construction, testing and maintenance) that are known as framework activities in software process [32]. We identified 263 unique subject matters, wherein the majority of them are related to technical aspects of SE (e.g. requirements, defects, tools and techniques). More than half of the primary studies presented the taxonomies in a textual manner.
However, we noticed that it was easier to understand the taxonomies presented in a graphical manner.

Although taxonomies are vastly recognized as classification mechanisms, a good number of primary studies stated taxonomy purposes that are not very closely related to classification. However, those taxonomies are still able to classify the aimed subject matters.

We identified that a large number of primary studies have either used an illustration (45.76%) to validate the proposed taxonomy or have not performed any validation (33.58%) at all. We believe that researchers should, whenever possible, select a suitable approach for validating the proposed taxonomy considering factors such as taxonomy purpose and audience.

We also identified that hierarchy (53.14%) and facet-based (39.48%) were the most frequently used classification structures. Majority of the studies used a qualitative procedure to assign subject matter instances to classes.

The overall result of the mapping study indicates that mostly SE taxonomies have been developed without following any systematic approach. Bayona-Oré et al. have recently proposed a method to develop SE taxonomies. Nevertheless, in the light of results presented in this paper, we identified a need to revise this method. Therefore, we revised Bayona-Oré et al.’s method, which is another contribution of the study reported in this chapter.
4.7 References


Chapter 5

An Effort Estimation Taxonomy for Agile Software Development

5.1 Introduction

Software is developed incrementally and iteratively in Agile Software Development (ASD). Like other activities in ASD, planning and estimation are also performed iteratively at various stages such as during release or sprint planning. Estimation in ASD has been investigated in a number of studies. We have recently aggregated these works in a Systematic Literature Review (SLR) [1]. In order to get a more broader understanding on how effort estimation is practiced by agile teams, we decided to carry out a follow up study to elicit the state of the practice on effort estimation in ASD. The SLR included only peer reviewed empirical works as primary studies, and did not include practitioners’ opinion expressed in white papers, blogs, forums etc. The follow up study [2], on the other hand, directly elicited opinion of agile practitioners on effort estimation in ASD to complement the findings of the SLR. These studies identified and aggregated knowledge on effort estimation in ASD from the literature (SLR) and the industry (survey). The knowledge includes aspects such as techniques used to estimate effort in ASD, employed size measures and other cost drivers and the context in which estimation is performed in ASD.

We believe that this body of knowledge on effort estimation in ASD needs to be organized to facilitate both future research and practice in this area. Taxonomies have been used to organize the body of knowledge in software engineering [4] and in other disciplines [5] as well, and have played an important role in maturing these fields. The
concept of taxonomy was originally used by the Swedish scientist Carl Linnaeus in the 18th century wherein he used hierarchical structure to classify the nature (animals, plants and minerals) based on common physical characteristics [9]. Taxonomy mainly is a classification mechanism, and is defined as follows in two well known dictionaries:

- The Cambridge dictionary\(^1\) defines taxonomy as “a system for naming and organizing things, especially plants and animals, into groups that share similar qualities”.

- The Oxford dictionaries\(^2\) define taxonomy as “The classification of something, especially organisms” or “A scheme of classification”.

Classifications offer both theoretical and practical advantages. On one side they serve as a tool to explore the relationships between entities and identify gaps in the existing knowledge, while on the other side they can support the selection of the appropriate entities in a real scenario [7]. In this work we propose a taxonomy to classify and characterize the effort estimation activities of software projects conducted in an ASD context. Few classifications of effort estimation techniques have been proposed in the literature. They are briefly described in Section 5.2. These classifications are about estimation techniques only. Our taxonomy, on the other hand, covers the whole estimation activity that also includes, besides techniques, effort predictors, estimation context and effort estimates. Following are the contributions of the proposed taxonomy:

- It organizes the body of knowledge on effort estimation in ASD.

- It provides a mechanism to classify and characterize the effort estimation in ASD. The taxonomy dimensions can be used by researchers to assess and improve the reporting of research on effort estimation in ASD.

- It supports effort estimation practice in ASD. The taxonomy dimensions can be used to document important estimation related information and thereby helping in externalizing the otherwise tacit knowledge of estimation sessions. The taxonomy can also help the agile practitioners during estimation sessions by reminding them of the important aspects that should be considered when effort is being estimated.

The taxonomy was designed by following a taxonomy design method, which is briefly described in Section 5.3. The taxonomy was used to characterize 10 estimation

---

\(^1\)www.dictionary.cambridge.org

\(^2\)www.oxforddictionaries.com
cases reported in the literature. The usefulness of the taxonomy was also demonstrated by classifying and characterizing the estimation activities of five agile teams.

The rest of the chapter is organized as follows: Section 5.2 describes the related work; Section 5.3 details the research questions and methods used to design and evaluate our agile estimation taxonomy; Section 5.4 presents the results; Section 5.5 compares the proposed taxonomy with existing work and also discusses the validity threats; Section 5.6 concludes the chapter.

5.2 Related Work

In this section we describe the background work that lead to the design of the proposed taxonomy, and existing classifications of effort estimation techniques.

5.2.1 State of the art and practice on effort estimation in ASD

We conducted a Systematic Literature Review (SLR) [1] with the aim to identify the state of the art on effort estimation in ASD. The SLR was performed by following Kitchenham and Charters guidelines [10]. The SLR reported the results based on 20 papers covering 25 studies on agile effort estimation. The details relating to the planning, execution and analysis of the SLR results are described in [1]. To complement the findings of the SLR, we carried out a follow up study [2] to identify state of the practice on effort estimation in ASD. As the data and results of these two empirical studies forms the basis for the agile estimation taxonomy presented in Section 5.4, we briefly present and compare the results of these two empirical studies in this section covering the context, estimation techniques and effort predictors used to estimate effort in ASD.

Estimation context

Both empirical studies identified various contextual factors around which agile teams estimate effort. In this section, we present the findings of the two studies on three contextual factors: planning level, development activities and agile methods.

The results of the SLR and survey show that estimation is mainly performed during release and iteration planning levels in ASD. However, many primary studies (48%) in our SLR did not explicitly specify the planning level at which estimation was performed. The survey results showed that some agile teams (16.67%) estimate effort during daily meetings as well. A handful of the agile practitioners in the survey (13.33%) stated that their teams estimate effort at project bidding level as well. The results are summarized in Table 5.1.
Table 5.1: Planning levels at which agile teams estimate

<table>
<thead>
<tr>
<th>SLR</th>
<th>Survey</th>
</tr>
</thead>
<tbody>
<tr>
<td>No. of studies (%)</td>
<td>No. of responses (%)</td>
</tr>
<tr>
<td>Iteration (I) – 6 (24%)</td>
<td>Iteration (I) – 49 (81.67%)</td>
</tr>
<tr>
<td>Release (R) – 5 (20%)</td>
<td>Release (R) – 25 (41.67%)</td>
</tr>
<tr>
<td>Both I &amp; R – 2 (8%)</td>
<td>Daily – 10 (16.67%)</td>
</tr>
<tr>
<td>Not described – 12 (48%)</td>
<td>Project bidding – 8 (13.33%)</td>
</tr>
<tr>
<td>Other – 5 (8.33%)</td>
<td></td>
</tr>
</tbody>
</table>

Table 5.2: Development activities estimated

<table>
<thead>
<tr>
<th>SLR</th>
<th>Survey</th>
</tr>
</thead>
<tbody>
<tr>
<td>No. of studies (%)</td>
<td>No. of responses (%)</td>
</tr>
<tr>
<td>All – 9 (36%)</td>
<td>All – 8 (13.3%)</td>
</tr>
<tr>
<td>Testing (T) – 4 (16%)</td>
<td>All except Maintenance – 11 (18.3%)</td>
</tr>
<tr>
<td>Implementation (I) – 3 (12%)</td>
<td>All except Analysis – 5 (8.3%)</td>
</tr>
<tr>
<td>Both I &amp; T – 3 (12%)</td>
<td>Both I &amp; T – 17 (28.3%)</td>
</tr>
<tr>
<td>Not described – 6 (24%)</td>
<td>Implementation only – 7 (11.7%)</td>
</tr>
<tr>
<td>Other combination – 12 (20%)</td>
<td></td>
</tr>
</tbody>
</table>

The SLR and the survey also investigated the scope of the effort estimates with respect to the development activities, i.e., which development activities are accounted for in the effort estimates. Implementation and the testing were found to be the main estimated activities in both studies. In 36% of the primary studies of the SLR, all development activities were accounted for in effort estimates. 24% of the primary studies did not report this information. The results are listed in Table 5.2.

Scrum and eXtreme Programming (XP) were identified as the leading agile methods in the SLR studies and the survey responses. It was interesting to note that the primary studies of the SLR used only individual agile method (Scrum or XP). However, a large number of the survey respondents cited using a combination of the methods. Kanban, for example, is used mostly in combination with Scrum. These results are summarized in Table 5.3.

Some other contextual factors were also elicited in the SLR and the survey. These include, besides other factors, the team size, the project/product domain and the team setting. In most cases the team size was found to be less than 10. Agile teams in 80% of the SLR’s primary studies were co-located, while remaining 20% of the SLR studies...
Table 5.3: Agile methods practiced

<table>
<thead>
<tr>
<th>SLR</th>
<th>Survey</th>
</tr>
</thead>
<tbody>
<tr>
<td>No. of studies (%)</td>
<td>No. of responses (%)</td>
</tr>
<tr>
<td>Scrum – 8 (32%)</td>
<td>Scrum – 25 (41.67%)</td>
</tr>
<tr>
<td>XP – 7 (28%)</td>
<td>Scrum, Kanban and XP – 9 (15%)</td>
</tr>
<tr>
<td>Not described – 10 (40%)</td>
<td>Scrum and Kanban – 7 (11.67%)</td>
</tr>
<tr>
<td></td>
<td>Scrum and XP – 5 (8.3%)</td>
</tr>
<tr>
<td></td>
<td>XP – 5 (8.3%)</td>
</tr>
<tr>
<td></td>
<td>Other – 9 (15%)</td>
</tr>
</tbody>
</table>

included geographically distributed teams. In case of the survey, all respondents stated that they are currently part of co-located teams.

Estimation technique

The results of the SLR and the survey showed that the estimation techniques (see Table 5.4) that rely on the subjective decision of the practitioners are used to estimate effort in ASD context. Most frequently used techniques are expert judgment, planning poker, use case points estimation method. These findings are in line with the agile philosophy wherein people are valued more than processes, and the heavyweight models and processes are not preferred. Effort estimation in ASD is mostly a group activity, i.e., the effort is estimated by the team.

Table 5.4: Estimation techniques used in ASD

<table>
<thead>
<tr>
<th>SLR</th>
<th>Survey</th>
</tr>
</thead>
<tbody>
<tr>
<td>No. of studies (%)</td>
<td>No. of responses (%)</td>
</tr>
<tr>
<td>Expert judgment – 8 (32%)</td>
<td>Planning poker – 38 (63.33%)</td>
</tr>
<tr>
<td>Planning poker – 6 (24%)</td>
<td>Estimation by analogy – 28 (46.67%)</td>
</tr>
<tr>
<td>UCP method – 6 (24%)</td>
<td>Expert judgment – 23 (38.3%)</td>
</tr>
<tr>
<td>UCP modification – 5 (20%)</td>
<td>UCP method – 8 (13.33%)</td>
</tr>
<tr>
<td>Regression – 3 (12%)</td>
<td>Delphi – 3 (5%)</td>
</tr>
<tr>
<td>AI – 3 (12%)</td>
<td>Other – 9 (14.33%)</td>
</tr>
<tr>
<td>Other – 6 (24%)</td>
<td></td>
</tr>
</tbody>
</table>

UCP = Use Case Points

The primary studies in the SLR used various measures such as MMRE (Mean Mag-
nitude of Relative Error) to calculate the accuracy of the estimates. However according to the survey results, these measures are not very common in the industry. For most practitioners a simple comparison of the estimated and the actual effort is sufficient. The techniques achieved varying degree of accuracy in the primary studies of SLR, while the survey results indicated that there is a tendency to underestimate the effort (42% of the respondents believe that estimates are underestimated by 25% or more).

**Effort predictors**

The product size and the cost drivers constitute effort predictors. The product size is believed to be an important effort predictor. Different metrics (e.g. function points) have been used to measure product size. Our Survey and the SLR results (see Table 5.5) showed that story points, function and use case points are the most frequently used size measures during effort estimation in ASD.

**Table 5.5: Size measures used in ASD**

<table>
<thead>
<tr>
<th>SLR</th>
<th>Survey</th>
</tr>
</thead>
<tbody>
<tr>
<td>No. of studies (%)</td>
<td>No. of responses (%)</td>
</tr>
<tr>
<td>UCPs – 7 (28%)</td>
<td>Story points – 37 (61.67%)</td>
</tr>
<tr>
<td>Story points – 6 (24%)</td>
<td>Function points – 17 (28.33%)</td>
</tr>
<tr>
<td>Function points – 3 (12%)</td>
<td>UCPs – 10 (16.67%)</td>
</tr>
<tr>
<td>LOC – 1 (4%)</td>
<td>LOC – 7 (11.67%)</td>
</tr>
<tr>
<td>Other – 3 (12%)</td>
<td>Object point – 1 (1.67%)</td>
</tr>
<tr>
<td>Not described – 5 (20%)</td>
<td>Not used – 12 (20%)</td>
</tr>
</tbody>
</table>

UCP = Use Case Points, LOC = Lines of Code

With regard to the cost drivers, the survey and the SLR results identified that the team related cost drivers (see Table 5.6) play a significant role in the effort estimation in ASD. This aligns well with values of the agile development wherein the individuals and the teams are valued higher than the processes and the tools. Other cost drivers that are also considered important are non functional requirements (e.g. performance, security), project/product domain and customer communication.

**5.2.2 Recent studies on effort estimation in ASD**

Section 5.2.1 presented a summary of review on state of the art and practice on effort estimation in ASD till 2014. In this section, we discuss some recently reported studies on effort estimation in ASD.
Table 5.6: Cost drivers used in ASD

<table>
<thead>
<tr>
<th>SLR</th>
<th>Survey</th>
</tr>
</thead>
<tbody>
<tr>
<td>No. of studies (%)</td>
<td>No. of responses (%)</td>
</tr>
<tr>
<td>Team’s skill level – 3 (12%)</td>
<td>Team’s skill level – 43 (71.67%)</td>
</tr>
<tr>
<td>Team’s prior exp. – 2 (8%)</td>
<td>Team’s prior exp. – 39 (65%)</td>
</tr>
<tr>
<td>Task size – 2 (8%)</td>
<td>Task size – 25 (41.67%)</td>
</tr>
<tr>
<td>NFRs – 2 (8%)</td>
<td>Project domain – 25 (41.67%)</td>
</tr>
<tr>
<td>Other – 10 (40%)</td>
<td>NFRs – 24 (40%)</td>
</tr>
<tr>
<td>Not investigated – 9 (36%)</td>
<td>Customer communication – 14 (23.3%)</td>
</tr>
<tr>
<td></td>
<td>Distributed team issues – 5 (8.33%)</td>
</tr>
<tr>
<td></td>
<td>Other – 10 (16.67%)</td>
</tr>
</tbody>
</table>

NFRs = Non Functional Requirements

Garg and Gupta [36] proposed a new cost estimation model for agile software projects. Using project characteristics as input, the proposed model used PCA (Principal Component Analysis) to identify a reduced set of key characteristics that are relevant for the development cost. Constraint solving approach is then employed to ensure the fulfillment of the agile manifesto criteria. The model is applied on agile projects dataset collected from multiple software companies in different countries. The authors noted that the proposed model demonstrated better estimation accuracy in terms of MMRE (Mean Magnitude of Relative Error) compared to the other approaches such as planning poker.

Continuous changes in user stories in ASD lead to uncertainty, which is one of the main reasons for inaccurate time and cost estimates [37]. Popli and Chauahn [37] proposed a technique to manage this uncertainty in ASD to improve estimation. In the proposed technique the user stories are first divided into sub-stories, and then for each of the sub-stories three types of story points are calculated: fastest, practical and maximum. These three estimates are then averaged to compute the final estimated story points. The authors illustrated the proposed technique using a small example project. Zahraoui and Idrissi [35] proposed adjustments in story points measure to improve effort estimation in ASD. The new adjusted story points are calculated by adjusting the story points with the priority, size and complexity of the user stories being estimated.

Lenarduzzi et al. [38] reported a replicated case study on functional size measures and effort estimation in Scrum. The case study consisted of a small web based application developed in ASP.net by four part-time developers using Scrum as development process. The results of the study showed that the effort estimation by developers is
more accurate than estimation through functional size measures. The study concluded that the introduction of functional size measures do not help in improving accuracy of the effort estimates in Scrum.

Tanveer et al. [39] performed an industrial case study to understand and improve effort estimation in ASD. The results showed that the accuracy of the effort estimation is affected by factors such as developers’ knowledge and experience, and the complexity of the required changes in the system. These factors should be considered by developers during estimation process. They also noted that the use of tool to support the impact analysis can also improve effort estimation. The use of tool would aid practitioners in explicitly considering important factors during effort estimation process.

Ramirez-Noriega et al. [40] proposed a technique for the identification and validation of the factors that teams use to decide the complexity and importance of the tasks being estimated using planning poker in Scrum. The approach then uses Bayesian Networks (BNs), as knowledge structure, to relate these factors with each other to improve the accuracy of the estimates.

5.2.3 Classifications of effort estimation techniques

Many different classifications schemes have been proposed for effort or cost estimation techniques during last 30 years or so. Boehm [24] presented seven classes of the estimation techniques. These were: Algorithmic models, Analogy, Expert judgment, Parkinson, Price to win, Top-down and Bottom-up. Briand and Isabella [21] presented a hierarchical classification of the estimation techniques. At the top are the two main categories: Model based and Non-model based techniques. The model based techniques are further divided into generic and specific model based techniques. Mendes [22], on the other hand, divided the effort estimation techniques in three broad categories: Algorithmic models, expert based and artificial intelligence techniques.

A more detailed and recent classification of effort estimation techniques is proposed by Trendowicz and Jeffery [25]. Effort estimation techniques are divided into three top categories (Data-driven, Expert-based and Hybrid) that are further divided into sub-categories. Data-driven techniques, for instance, are divided into Proprietary and non-proprietary techniques. Model-based, memory based and composite are the sub-categories of the non-proprietary techniques.

The work presented in this study is different from these classifications. We are not proposing a new classification of effort estimation techniques. Our proposed taxonomy covers the whole effort estimation activity in ASD, whereby the actual effort estimation technique is only one facet of the proposed taxonomy. The facets of the taxonomy are described in Section 5.4. Since the recent work by Trendowicz and Jeffery [25] is very
comprehensive and also covers other aspects besides estimation techniques, we will come back to their work in Section 5.5.

5.3 Research Methodology

In this section, we describe the research questions and methods used to design and evaluate the proposed taxonomy.

5.3.1 Research Questions

Following research questions are addressed in this study:

- **RQ1**: How to organize the knowledge on effort estimation in ASD?
- **RQ2**: How can the agile effort estimation taxonomy be used to assess and improve reporting of research on effort estimation in ASD?
- **RQ3**: What is the usefulness of the taxonomy for effort estimation practice in ASD?

RQ1 is answered by organizing the knowledge on effort estimation in ASD as a taxonomy. The dimensions and facets of the taxonomy are described in Section 5.4. RQ2 is answered by applying the classification mechanism of the proposed taxonomy to characterize the effort estimation activities of the agile projects reported in literature. The characterization is detailed in Section 5.4.2. To answer RQ3, we used the taxonomy to classify and characterize the effort estimation activities of five agile teams. Further details about RQ3 are provided in Section 5.4.3.

5.3.2 Taxonomy design method

In this section, we briefly present the method used in this study to design the agile estimation taxonomy. The method is depicted in Table 5.7. The method is an update of the method presented by Bayona-Oré *et al.* [11], based on our observations and experiences from a systematic mapping study [3] on taxonomies in the software engineering discipline. The revised method consists of 4 phases and 13 activities. Although the method is presented as sequential arrangement of phases and activities, the taxonomy designers, however, may find it necessary to iterate between phases and the activities.
Chapter 5. An Effort Estimation Taxonomy for Agile Software Development

Table 5.7: Taxonomy design method from [3]

<table>
<thead>
<tr>
<th>Phase</th>
<th>Activity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Planning</td>
<td>Define SE knowledge area</td>
</tr>
<tr>
<td></td>
<td>Describe the objectives of the taxonomy</td>
</tr>
<tr>
<td></td>
<td>Describe the subject matter to be classified</td>
</tr>
<tr>
<td></td>
<td>Select classification structure type</td>
</tr>
<tr>
<td></td>
<td>Select classification procedure type</td>
</tr>
<tr>
<td></td>
<td>Identify information sources</td>
</tr>
<tr>
<td>Identification and extraction</td>
<td>Extract all the terms</td>
</tr>
<tr>
<td></td>
<td>Perform terminology control</td>
</tr>
<tr>
<td>Design and construction</td>
<td>Identify and describe the taxonomy dimensions</td>
</tr>
<tr>
<td></td>
<td>Identify and describe the categories of each dimension</td>
</tr>
<tr>
<td></td>
<td>Identify and describe the relationships</td>
</tr>
<tr>
<td></td>
<td>Define guidelines for using and updating the taxonomy</td>
</tr>
<tr>
<td>Validation</td>
<td>Validate the taxonomy</td>
</tr>
</tbody>
</table>

Phase 1: Planning

Planning is the first phase wherein basic decisions about the taxonomy design are made. It includes the following activities:

Define SE Knowledge Area (KA): In this activity the SE KA or sub-area is selected and described in which the taxonomy is designed. The Software Engineering Body of Knowledge (SWEBOK) version 3 [26] divides the SE discipline into 15 KAs (e.g., software requirements, software design, software construction, testing etc.). These are further subdivided into sub and sub-sub areas.

The taxonomy proposed in this study is about effort estimation in the ASD context. Effort estimation plays an important role in managing agile projects during release and sprint planning [12]. Effort estimation falls within the scope of the “Software engineering management” KA in SWEBOK [26].

Describe objectives of the taxonomy: The objective(s) of the taxonomy should be clearly stated. The taxonomy designers should not leave it to the audience to guess the objectives of the taxonomy.
The main purpose of the taxonomy in this study is to propose a classification scheme that can be used to characterize effort estimation activity of an agile project. A number of studies, included in SLR [1] on effort estimation in ASD, have not reported important information related to the context, techniques and predictors used during effort estimation. The results of such studies are hard to analyze and compare, and therefore not very useful for practitioners. The proposed taxonomy could be used by researchers to consistently report important aspects related to effort estimation in ASD.

Agile practitioners, on the other hand, could use the taxonomy during estimation sessions to remind themselves of important aspects that should be considered to arrive at better effort estimates. After estimation session, characterization of the estimation activities using the proposed taxonomy will also facilitate the documentation of the information related to the effort estimation sessions, which otherwise remain mostly tacit. The teams can use this documentation in future estimation sessions.

**Describe the subject matter to be classified:** The taxonomy designers should make an explicit effort in describing what exactly is classified in the proposed taxonomy. The KA specifies the broad topic of the taxonomy, while the subject matter defines what exactly is classified in the taxonomy. In a taxonomy of software testing techniques, for instance, software testing is the KA, while testing techniques are the subject matter.

Effort estimation activities of the projects that are managed with an agile method is the subject matter of this taxonomy. Estimation activities are characterized by a number of factors such as estimation technique(s) used, considered effort predictors, context etc.

**Select classification structure type:** After these initial steps the taxonomy designers have to select an appropriate classification structure. There are four basic classification structures: hierarchy, tree, paradigm and faceted analysis [6]. These structures have their own strengths and limitations, and there are situations in which one is more suitable than the rest. The taxonomy designers should also provide a clear justification for selecting a specific classification structure.

We have selected faceted classification to structure our taxonomy. Faceted classification is suitable for the evolving areas, such as effort estimation in ASD, since it is not required to have complete knowledge of the area to design a facet-based taxonomy [6]. In faceted classification based taxonomies, the subject matter is classified from multiple perspectives (facets). Each facet is independent and has its own attributes, making the facet-based taxonomies easily evolvable.
Select classification procedure type: The taxonomy designers also need to select and describe the type of the classification procedures that will be employed to assign samples of the subject matter to the most relevant categories. At a broader level there are two types of procedures depending upon the employed measurement system: qualitative and quantitative. The qualitative procedures are based on nominal scales wherein the relationship between the classes cannot be achieved. The researchers use the nominal scale to subjectively assign samples of the subject matter to the relevant classes. The quantitative classification procedures, on the other hand, are based on numerical scales [5].

Each facet of our taxonomy has a set of possible values. The details are described in Section 5.4. Based on the available data, we used the qualitative procedure to select relevant facet values to characterize a specific estimation activity. In some cases it may not be possible to assign a value simply due to insufficient data.

Identify information sources: Lastly in the planning phase, the information sources to extract the relevant data are identified.

The information sources used in the design of our agile estimation taxonomy consist of: A) The peer reviewed empirical studies on effort estimation in ASD published between 2001 and 2013. The evidence from these studies was aggregated in the SLR summarized in Section 5.2. B) Agile practitioners who participated in our survey study, which is summarized in Section 5.2.

Phase 2: Identification and Extraction

In the second phase the relevant data is identified and extracted from the information sources identified in the previous phase. It consists of the following activities:

Extract all terms: The terms and concepts are extracted from the sources identified in the planning phase. Data from the literature was extracted originally as part of the SLR [1]. The data from the agile practitioners was elicited as part of our survey study [2].

Perform terminology control: Next is to remove inconsistencies in the extracted data. It is possible that different researchers have referred to the same concept with different terms or different concepts with the same terms. First level of terminology control was already performed during the data extraction and analysis phases of the two empirical studies (SLR and survey) that are providing the data for the taxonomy.
presented herein. We, therefore only had to make few minor adjustments to consistently represent all terms and concepts.

**Phase 3: Design**

After the extraction phase, the next task is to identify the main dimensions and the categories therein along which the extracted data items could be organized. This phase consists of the following activities:

**Identify and define taxonomy dimensions:** Faceted classification based taxonomies have multiple dimensions (or perspectives) at the top along which a subject matter is classified. These dimensions can be identified in two different ways: Bottom-up and top-down. In the bottom-up approach the main dimensions and the categories emerge during the extraction and analysis process [27]. In the top-down approach, on the other hand, the taxonomy designers have a priori notion of some dimensions and categories [27] due to their knowledge of the selected knowledge area.

We have used a hybrid approach and identified following top level dimensions of our taxonomy: Estimation technique, effort predictors, effort estimate and estimation context. The first three are part of most estimation processes described in the literature, see for example estimation process described in [22]. The fourth dimension (estimation context) emerged during the data extraction and analysis phases to define and document the environment in which the effort estimation activity is carried out in an agile project. These dimensions are detailed in Section 5.4.

**Identify and describe categories of each dimension:** The next step is to identify and describe the next levels of the taxonomy, which are referred here in taxonomy design method as categories. These categories can be identified by using the same top-down, bottom-up or hybrid approaches described above.

We refer these categories as facets in our taxonomy. Most of these facets emerged when the results of the SLR [1] and survey [2] studies were analyzed. The facets related to the techniques were grouped under estimation technique dimension. Similarly cost drivers and size measures were grouped under effort predictors dimensions. The data items related to the effort estimate (e.g., unit of estimate) were categorized under the effort estimate dimension. The SLR and survey studies also identified number of factors that characterize the context of the estimation activity. The identification of these factors lead to the fourth dimension of the taxonomy, i.e., estimation context. The facets are described in detail in Section 5.4.
Identify and describe relationships, if any: The relationships between dimensions and categories are identified and described clearly. Note that in some cases there is no relationship between the dimensions, i.e., this activity might be skipped.

The objective of the taxonomy proposed in this study is to provide a classification mechanism to characterize effort estimation activity in an ASD context. Each dimension is a grouping of facets that are closely related to each other. The four dimensions together can characterize an estimation activity. However no other specific relationships between dimensions are defined, allowing for easy addition of new dimensions and the facets.

Define guidelines for using and updating the taxonomy: The aim of this activity is to describe how the taxonomy can be used. As a result of the application of the taxonomy and other future research efforts, it is highly likely that the need to update the taxonomy emerge. The taxonomy designers should also describe how the taxonomy can be updated in future.

The proposed effort estimation taxonomy can be used to assess and improve the reporting of studies on effort estimation in ASD. The taxonomy facets can be used as a checklist to ensure that all important factors about effort estimation are reported. Practitioners can document important information by using the taxonomy to characterize the specific estimation activity of an agile project. This documented information can serve as an important repository in improving future effort estimation. We applied the proposed agile taxonomy on knowledge extracted from literature and industry to demonstrate these uses. The details are provided in Section 5.4.2 and 5.4.3.

The dimensions and the facets of the proposed taxonomy are not exhaustive, and should be updated based on future research efforts and feedback from industry. At present they are based on knowledge that we aggregated in our SLR [1] and survey [2]. We have used faceted classification in our taxonomy. Faceted classification is a flexible structure wherein new facets can be added relatively easily. One likely extension is under the facet “effort predictor”. Currently we only included those cost drivers that are identified as important in the SLR and the survey. The future research on effort estimation in ASD may and should investigate more cost drivers, which could then be included in our taxonomy.

Phase 4: Validation

Validation is the last phase of this method wherein the taxonomy designers attempt to demonstrate the working or usefulness of the designed taxonomy. This phase has only one activity. A number of approaches can be used to validate a taxonomy. It includes
establishing the orthogonality of taxonomy dimensions, comparing with existing taxonomies and demonstrating its usefulness by applying it on existing literature [20].

Since there is no existing taxonomy on effort estimation in ASD, benchmarking our taxonomy against existing ones is not strictly possible. However, in Section 5.5 we compare our taxonomy with a recent and extensive review of the software effort estimation area by Trendowicz and Jeffery [25]. Their work is selected for the comparison as it is not only recent and comprehensive, but it also covers aspects related to our taxonomy dimensions. Orthogonality of the taxonomy dimensions is discussed in the Section 5.4.1. The four dimensions (Figure 5.1), representing the top level of taxonomy, are clearly distinguishable from each other. Effort estimation is performed in a certain Context, wherein agile teams use certain estimation technique(s) and effort predictors to arrive at an effort estimate.

Two methods are employed to evaluate the usefulness of the proposed taxonomy. These methods are briefly described here.

**Taxonomy evaluation: Literature** The proposed taxonomy is used to characterize ten estimation cases reported in the literature. The studies were selected from the set of primary studies included in our SLR on effort estimation in ASD. We selected only those studies that were based on real agile projects. An extraction sheet was prepared with fields corresponding to the facets of the proposed taxonomy. The data reported in the selected studies was extracted and entered in the sheet with respect to the facets of the taxonomy. The results of the evaluation are presented in Section 5.4.2.

**Taxonomy evaluation: Industry** The usefulness of the proposed taxonomy is also demonstrated by classifying and characterizing the effort estimation activities of five agile teams. We conducted interviews of four agile practitioners about effort estimation activities that their teams performed in recent sprint or release planning meeting. These four agile practitioners represents a convenience sample, i.e., they were recruited based on our personal contacts. They belong to three different companies, and provided data about effort estimation activities performed in development of five different products. Two interviewees lead effort estimation sessions, one participate as testing lead and one as developer in their respective companies. The details of the companies and products are described in Section 5.4.3. Table 5.8 lists their brief profiles.

The interviews were semi-structured and were conducted by the first author. The interviewees were first asked to describe when and how effort estimation is carried out in their respective teams. The interviewees were then asked to select recent instances of estimation sessions in which they participated. The taxonomy dimensions and the facets were used to ask questions and elicit the data required to characterize the spe-
specific estimation activity. The interviews lasted, on an average, for an hour. After the interviews, the data extraction sheets were sent to the respective interviewees for further feedback, validation and corrections, if required. The results of the evaluation are presented in Section 5.4.3.

Table 5.8: Details of the interviewees

<table>
<thead>
<tr>
<th>#</th>
<th>Current Location</th>
<th>Role</th>
<th>Exp. in current company</th>
<th>Total exp.</th>
<th>No. of estimation sessions</th>
<th>No. of teams and products</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Sweden</td>
<td>Test lead</td>
<td>1.5 years</td>
<td>6 years</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>Pakistan</td>
<td>Dev Manager</td>
<td>11 years</td>
<td>11 Years</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>Sweden</td>
<td>Product Owner</td>
<td>7 years</td>
<td>11 years</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>Sweden</td>
<td>Developer</td>
<td>1 year</td>
<td>10 years</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Together these four practitioners from three different companies provided data about six different estimation sessions corresponding to five different products and teams.

5.4 Results

In this section first we present the agile estimation taxonomy to answer RQ1. Later, the applications of the taxonomy on the literature and the industrial estimation cases are presented to answer RQ2 and RQ3 respectively.

5.4.1 RQ1: Organizing the knowledge on effort estimation in ASD

We have organized the identified knowledge on effort estimation in ASD as a taxonomy. Taxonomies are an effective tool to organize and communicate the knowledge in an area [4]. The proposed taxonomy was created by following a method that was described in Section 5.3. In this section, we present the outcome of that method, i.e. the agile estimation taxonomy.

At the top level the taxonomy has four dimensions, which are presented in Figure 5.1. A dimension encompasses all facets that are related to each other. The dimension “Effort predictors”, for example, includes effort predictors employed during estimation. These four dimensions define the first level of taxonomy, and provide an overview of the taxonomy at a broader level.

Estimation context represents grouping of those facets (or categories in more general terms) that define and characterize the context in which the estimation activity is
carried out in an agile project. It includes, beside others, agile method in use, team size, project domain etc. **Effort predictors** dimension encompasses predictors used during effort estimation. **Estimation technique** dimension includes facets related to the technique used to arrive at effort estimates. **Effort estimate** includes the facets related to effort estimates such as estimate value, estimate type etc.

![Figure 5.1: First level of the Agile estimation taxonomy](image)

In order to completely characterize a specific effort estimation activity of an agile project, facets of all dimensions need to be described. In the following, we describe each of these four dimensions and their facets in detail.

**Estimation context**

Context is an inseparable part of the software practice, and it needs to be properly captured and reported in the empirical studies to communicate the applicability of the research findings. The empirical research findings are not universal due to the unique context in which the research is conducted. The empirical research needs to be properly contextualized so that the readers can determine the relevance of the results for their contexts, i.e., what is applicable for a company, and when and where to use the reported findings [13].

The context dimension has 8 facets. These facets and their possible values are depicted in Figure 5.2. They are described in the following.

- **Planning level**: Estimation supports planning at different levels in ASD. It includes mainly release and sprint planning, while some teams may also estimate during daily meetings [12]. Project bidding is another level wherein companies have to estimate the total development effort upfront in order to bid for the projects.
Chapter 5. An Effort Estimation Taxonomy for Agile Software Development

- **Main estimated activity**: This facet describes which development activities are accounted for in effort estimates. For example does the total effort estimate include the time spent on fixing bugs, i.e., maintenance, or it is considered separately. It is possible that some team members have different understandings towards the estimation scope due to, for example, the reason that the team is newly formulated, or it includes some inexperienced developers. Explicit consideration of this facet during estimation sessions will improve shared understanding amongst team members with respect to the scope of the estimation.

- **Agile method(s)**: This facet documents the agile method practiced by the agile team. Most agile teams customize and adopt agile method(s) to suit their context.

- **Product/project domain**: This facets represents the domain of product for which the development effort is being estimated. Different domains could lead to different sets of effort predictors and their impact on the total effort. For instance, different domains would have varying non functional requirements. We have used the categories from the domain taxonomies reviewed by Glass and Vessey [4] as example facet values in Figure 5.2.

- **Project setting**: This facet captures the setting in which the agile teams are developing the product. There are two broad settings, co-located team wherein all members work at the same location, and distributed development wherein the teams are placed at different geographical locations. The latter introduces additional challenges related to communication and coordination. In case of the distributed teams many different settings, depending on the temporal and geographical distance, are possible. Smite et al. [20] proposed a global software engineering taxonomy that characterizes different settings of the global teams. We have used situations in their GSE taxonomy as example facet values in Figure 5.2. The teams in distributed development could be working either in an onshore or an offshore setting.

- **Estimation entity**: Requirements are the basic input to the estimation activity. Requirements can be specified in different ways and at varying level of granularity. In ASD requirements are mostly specified in the form of user stories. In some cases user stories are estimated, while other teams divide the stories into tasks, which are then estimated. This facet documents the entity (e.g. task, user story or use case) that is being estimated.

- **Number of entities estimated**: It records the number of user stories or tasks estimated in a specific estimation session, and thus conveys important information related to the size of the sprint or release.
**Team size**: It documents the size of the team which is responsible for developing the estimated tasks or stories.

![Figure 5.2: Estimation context dimension](image)

**Effort predictors**

Effort predictors play an important role in arriving at effort estimates. They consist of size and other cost drivers such as team capabilities, non-functional requirements, etc. Product size is considered as one of the main determinants of the required development effort [25]. Cost drivers also contribute to the required effort depending on the context.

The effort predictors dimension has 6 facets. These facets and their possible values are depicted in Figure 5.3.

- **Product size**: Development effort, in general, is strongly correlated with product size. Many research efforts have been made to empirically relate effort with product size [14]. This facet documents whether the agile team uses size as a predictor and which metric is used to represent the size. It is also possible that estimators considered the task or story size as one of the predictors without using any specific size metric.
• **Team’s prior experience**: A development team’s prior experience with similar tasks impacts the required effort. Estimated effort for sprints that involve similar and familiar tasks will be low as compared to a sprint involving unfamiliar tasks. This facet describes whether a team’s prior experience was considered in arriving at the effort estimates.

• **Team’s skill level**: Team’s skill level also impacts the required effort. This facet documents whether skill level of the team members was considered during the effort estimation session.

• **Non functional requirements**: Stringent non functional requirements increase the development effort. This facet records whether or not, and which non functional requirements were considered in arriving at effort estimates. Chung *et al.* [28] provided a review of the types and classifications of the non functional requirements. We have used few non functional requirements, from a large list included in their work, as example facet values in Figure 5.3.

• **Distributed teams’ issues**: How many sites collaboratively develop software also has an impact on the development effort due to the increased complexity of the collaboration and communication. The cultural and geographical barriers between distributed development teams can increase the development effort.

• **Customer communication**: Effective communication with the customer is an important aspect of ASD, and break down in this communication could lead to delays and problems in resolving issues. It could further lead to effort overruns. This facet guides the estimating team to see if customer communication should be used as an effort predictor in their context.

The actual facet values for some cost drivers, such as team’s skill level, in Figure 5.3 are not provided. At the first level it is important to characterize whether or not certain cost drivers are considered relevant during effort estimation. It is not simple to quantitatively measure these cost drivers. A commonly used approach in most estimation methods, such as COCOMO (COstructive COst MOdel) [23] is to rank these cost drivers on an ordinal scale such as low, medium or high. A similar approach can be used here, if specific quantitative constructs or measures are absent.

**Estimation technique**

This dimension encompasses facets related to estimation techniques that should be documented in order to characterize estimation activities of an agile team. The facets of this dimension and their corresponding example values are depicted in Figure 5.4.
Figure 5.3: Effort predictors dimension

Figure 5.4: Estimation technique dimension
• **Technique**: This documents the actual estimation technique applied by the agile team. According to the survey and the SLR results most frequently applied techniques are planning poker, expert judgment and estimation by analogy. The agile teams can also use a combination of techniques to arrive at an effort estimate.

• **Type of technique**: There are different types of estimation techniques and models such as algorithmic, expert based and artificial intelligence based techniques and models. In ASD, estimation techniques that rely on subjective assessment of experts are frequently used [1]. These techniques can be used by an individual or a group of experts. This facet documents whether the effort was estimated using a individual or group based estimation technique.

**Effort estimate**

Effort estimates are the main output of the estimation activity. The facets in this dimension characterize effort estimate. These facets and their corresponding values are depicted in Figure 5.5.

• **Estimated effort**: This facet documents the main output of the estimation session or activity, i.e., the estimated effort.

• **Actual effort**: It is also important to have the actual effort, at the end of release or sprint, to enable comparison with the estimated effort.

• **Estimate Type**: The estimate could be of different types e.g. point estimate, three point estimate or a distribution. This facet documents the type of estimate used by the agile team.

• **Estimate unit**: Effort can be estimated in terms of hours, weeks or months. The concept of ideal time has also been applied in ASD. This facet records the unit used to represent the effort estimates.

• **Accuracy level**: This facets documents the accuracy level achieved by the used technique. It is better to have the actual and estimated effort reported to have a better idea about the accuracy of the estimates. However, some studies only report accuracy levels in terms of the percentages (e.g. MMRE).

• **Accuracy measure**: This facet records whether or not, and which, measure is used to assess the accuracy of the applied estimation technique. The commonly used accuracy measures are listed as example facet values in Figure 5.5.
5.4.2 RQ2: Reporting of research on effort estimation in ASD

In order to evaluate usefulness of the proposed taxonomy in assessing the completeness of reporting, we used the taxonomy to characterize ten effort estimation cases reported in the literature. The estimation cases were elicited from eight empirical studies on effort estimation in ASD. The characterizations are presented in Table 5.9. Agile estimation taxonomy is able to describe the effort estimation cases reported in literature. It is however interesting to note that some studies do not report important information, which makes it hard to compare and analyze results of these studies. We observed the following specific issues during this exercise:

- Authors make an effort to describe the context in which estimation studies are conducted. However in many cases the context is very briefly described, and is scattered in the report. We were not able to extract project domain, planning level and the number of estimated stories/tasks in two, three and four cases respectively.

- The reporting of the studies lack the details with respect to the effort predictors used during effort estimation. In six cases we were not even able to establish whether or not, and which, cost drivers were used. It is hard to imagine a real scenario wherein effort is estimated without considering any effort predictors. It is possible that the focus of a study is not on effort predictors. However this then
Chapter 5. An Effort Estimation Taxonomy for Agile Software Development

should be explained and described explicitly.

- Two studies have not described the technique used to estimate effort. The remaining eight studies have described the used technique and corresponding accuracy measure used. However we were not able to determine in 5 cases whether the technique was used by an individual or group of experts.

- Actual and estimated effort are not reported in six cases, which makes it hard to analyze and compare the findings.

The proposed taxonomy can be used by researchers to make sure that important information related to agile effort estimation is duly reported. This, we hope, will improve the reporting and would facilitate comparing and analyzing the results of studies on effort estimation in ASD.
Table 5.9: Taxonomy application: Literature

<table>
<thead>
<tr>
<th>Facet</th>
<th>[15]</th>
<th>[16]</th>
<th>[16]</th>
<th>[30]</th>
<th>[31]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Planning level</td>
<td>Release</td>
<td>Iteration</td>
<td>Iteration</td>
<td>Iteration &amp; Release</td>
<td>?</td>
</tr>
<tr>
<td>Estimated activity</td>
<td>Implementation (I)</td>
<td>I</td>
<td>I</td>
<td>All</td>
<td>All</td>
</tr>
<tr>
<td>Agile Method(s)</td>
<td>XP</td>
<td>XP</td>
<td>XP</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>Project Domain</td>
<td>IS for an ISP</td>
<td>?</td>
<td>?</td>
<td>Web MIS</td>
<td>MIS</td>
</tr>
<tr>
<td>Project setting</td>
<td>Co-located</td>
<td>Co-located</td>
<td>Co-located</td>
<td>Distributed</td>
<td>?</td>
</tr>
<tr>
<td>Estimation unit</td>
<td>User story</td>
<td>User story</td>
<td>User story</td>
<td>User story</td>
<td>Use cases</td>
</tr>
<tr>
<td>No. of tasks/stories</td>
<td>50</td>
<td>1325</td>
<td>1325</td>
<td>?</td>
<td>12</td>
</tr>
<tr>
<td>Team size</td>
<td>16-20</td>
<td>16-20</td>
<td>16-20</td>
<td>?</td>
<td>3</td>
</tr>
<tr>
<td>Size</td>
<td>not used</td>
<td>Length of US</td>
<td>?</td>
<td>Story points</td>
<td>UC points</td>
</tr>
<tr>
<td>Team’s prior experience</td>
<td>+</td>
<td>-</td>
<td>?</td>
<td>-</td>
<td>?</td>
</tr>
<tr>
<td>Team’s skill level</td>
<td>+</td>
<td>-</td>
<td>?</td>
<td>-</td>
<td>?</td>
</tr>
<tr>
<td>Non functional reqs.</td>
<td>-</td>
<td>-</td>
<td>?</td>
<td>Performance, security</td>
<td>?</td>
</tr>
<tr>
<td>Distributed teams’ issues</td>
<td>-</td>
<td>-</td>
<td>?</td>
<td>+</td>
<td>?</td>
</tr>
<tr>
<td>Customer communication</td>
<td>-</td>
<td>-</td>
<td>?</td>
<td>-</td>
<td>?</td>
</tr>
<tr>
<td>Technique</td>
<td>Planning poker</td>
<td>Stepwise reg.</td>
<td>Exp. judgment</td>
<td>Constructive agile estimation algo</td>
<td>UCP estimation</td>
</tr>
<tr>
<td>Type</td>
<td>Group</td>
<td>Other</td>
<td>Group</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>Estimation unit</td>
<td>Pair days</td>
<td>SPs converted to mins</td>
<td>Hrs</td>
<td>?</td>
<td>Hrs</td>
</tr>
<tr>
<td>Estimate type</td>
<td>Point</td>
<td>Point</td>
<td>Point</td>
<td>Point</td>
<td>Point</td>
</tr>
<tr>
<td>Actual effort</td>
<td>?</td>
<td>?</td>
<td>?</td>
<td>?</td>
<td>2600 Hrs</td>
</tr>
<tr>
<td>Accuracy measure &amp; level</td>
<td>MMRE(0.48), MdMMRE(0.25)</td>
<td>MMRE(0.90), Pred@25% (0.17), nMER(0.67)</td>
<td>MMRE(0.28), Pred@25% (0.51), nMER(0.25)</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>Facet</td>
<td>[17]</td>
<td>[17]</td>
<td>[32]</td>
<td>[18]</td>
<td>[19]</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>------</td>
<td>------</td>
<td>------</td>
<td>------</td>
<td>------</td>
</tr>
<tr>
<td>Estimated activity</td>
<td>Testing</td>
<td>Testing</td>
<td>All</td>
<td>I&amp;T</td>
<td>?</td>
</tr>
<tr>
<td>Agile Method(s)</td>
<td>SCRUM</td>
<td>SCRUM</td>
<td>?</td>
<td>SCRUM</td>
<td>XP</td>
</tr>
<tr>
<td>Project Domain</td>
<td>Web app.</td>
<td>Web app.</td>
<td>Accounting system</td>
<td>Web student IS</td>
<td>Desktop support sys.</td>
</tr>
<tr>
<td>Estimate unit</td>
<td>Use case</td>
<td>Use case</td>
<td>User story</td>
<td>User story</td>
<td>Task</td>
</tr>
<tr>
<td>No. of tasks/stories</td>
<td>?</td>
<td>?</td>
<td>?</td>
<td>200</td>
<td>4000</td>
</tr>
<tr>
<td>Team size</td>
<td>17</td>
<td>17</td>
<td>2</td>
<td>40 (10 teams of size 4)</td>
<td>15</td>
</tr>
<tr>
<td>Size measure</td>
<td>?</td>
<td>UC points</td>
<td>Story points</td>
<td>Story points</td>
<td>?</td>
</tr>
<tr>
<td>Team’s prior experience</td>
<td>?</td>
<td>+</td>
<td>?</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>Team’s skill level</td>
<td>?</td>
<td>+</td>
<td>?</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>Non functional reqs.</td>
<td>?</td>
<td>+</td>
<td>?</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>Distributed teams’ issues</td>
<td>?</td>
<td>-</td>
<td>N/A</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>Customer communication</td>
<td>?</td>
<td>-</td>
<td>?</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>Technique Type</td>
<td>Exp. judgment</td>
<td>UCP method</td>
<td>?</td>
<td>Planning poker</td>
<td>?</td>
</tr>
<tr>
<td>Type</td>
<td>Single</td>
<td>?</td>
<td>?</td>
<td>Group</td>
<td>?</td>
</tr>
<tr>
<td>Estimation unit</td>
<td>Hrs</td>
<td>Hrs</td>
<td>Hrs</td>
<td>Ideal days</td>
<td>Ideal hrs</td>
</tr>
<tr>
<td>Estimate type</td>
<td>Point</td>
<td>Point</td>
<td>Point</td>
<td>Point</td>
<td>Point</td>
</tr>
<tr>
<td>Actual effort</td>
<td>1090</td>
<td>1090</td>
<td>380</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>Estimated effort</td>
<td>750</td>
<td>870</td>
<td>400</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>Accuracy measure &amp; level</td>
<td>MRE(0.32)</td>
<td>MRE(0.21)</td>
<td>MRE(0.05)</td>
<td>MBRE(1.1074)</td>
<td>MMRE(0.19)</td>
</tr>
</tbody>
</table>

+ = Considered, - = Not considered, ? = Data unavailable
5.4.3 RQ3: Usefulness of taxonomy for effort estimation practice in ASD

The aim of this RQ was to evaluate if the proposed taxonomy can be used by agile teams in documenting their estimation sessions for future use. We used the taxonomy to classify and characterize the estimation activity of five agile teams working in three different companies. First we provide a brief description of the companies, teams and related products. The companies are referred here as A, B and C due to privacy concerns.

Company A

Company A is a medium sized software company established in the mid 90s headquartered in Finland with offices in Sweden and India as well. A develops and offers Business Support Solutions (BSS) to telecommunication service providers. It offers BSS services related to customer, product and revenue management. We collected the effort estimation related data from the recent sprint planning meetings of two different teams at Company A. We refer to these two teams as A1 and A2. A sprint normally lasts for two weeks. Both teams are located in Sweden, and are using Scrum as development process. A1 is responsible for developing Order Management System (OMS) of BSS, while A2 is developing Customer Information System (CIS).

We interviewed a developer from team A1 and a testing lead from A2, who both participated in the sprint planning meetings of their respective teams. Each interview lasted for about an hour. The effort estimation data was entered in extraction sheets by the interviewer, which were later sent to the interviewees for corrections and adding the missing information, if any. Our agile estimation taxonomy was used (see Table 5.10) to classify the specific effort estimation activities of both teams.

Company B

Company B is a very large organization, headquartered in Sweden, providing services and support systems related to communication technology. B develops and provides software solutions for Operations and Business Support Solutions (OSS/BSS). We interviewed a product owner for two hours who leads the release plan meetings that involve both design and testing leads and other team members. The teams are based in Sweden, and are following Scrum as their development process. They are responsible for developing one subsystem of a large billing system.

We elicited effort estimation data corresponding to two different releases; one is based on customer requirements, while the other is research based internal release. We
Table 5.10: Characterizing effort estimation activity of company A

<table>
<thead>
<tr>
<th>Dimension</th>
<th>Facet</th>
<th>OMS</th>
<th>CIS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Context</td>
<td>Planning level</td>
<td>Sprint</td>
<td>Sprint</td>
</tr>
<tr>
<td></td>
<td>Main estimated activity</td>
<td>Implementation and Testing</td>
<td>Implementation and Testing</td>
</tr>
<tr>
<td>Agile Method(s)</td>
<td>Scrum and XP practices</td>
<td>Scrum</td>
<td>Scrum</td>
</tr>
<tr>
<td>Product domain</td>
<td>Telecom</td>
<td>Telecom</td>
<td>Telecom</td>
</tr>
<tr>
<td>Project setting</td>
<td>Co-located</td>
<td>Co-located</td>
<td>Co-located</td>
</tr>
<tr>
<td>Estimation unit</td>
<td>Task</td>
<td>Task</td>
<td>Task</td>
</tr>
<tr>
<td>Number of tasks/stories</td>
<td>60 tasks</td>
<td>36 tasks</td>
<td></td>
</tr>
<tr>
<td>Team size</td>
<td>8</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>Effort</td>
<td>Size</td>
<td>Not used</td>
<td>Considered without any measure</td>
</tr>
<tr>
<td>Predictors</td>
<td>Team’s prior experience</td>
<td>Considered</td>
<td>Considered</td>
</tr>
<tr>
<td></td>
<td>Team’s skill level</td>
<td>Considered</td>
<td>Considered</td>
</tr>
<tr>
<td></td>
<td>Non functional requirements</td>
<td>Considered</td>
<td>Considered</td>
</tr>
<tr>
<td></td>
<td>Distributed teams’ issues</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td>Customer communication</td>
<td>Not considered</td>
<td>Not considered</td>
</tr>
<tr>
<td>Estimation</td>
<td>Technique name</td>
<td>Planning poker</td>
<td>Planning poker</td>
</tr>
<tr>
<td>Technique Type</td>
<td></td>
<td>Group estimation</td>
<td>Group estimation</td>
</tr>
<tr>
<td>Effort</td>
<td>Estimate unit</td>
<td>Hrs</td>
<td>Hrs</td>
</tr>
<tr>
<td>Estimate</td>
<td>Estimate type</td>
<td>Point</td>
<td>Point</td>
</tr>
<tr>
<td></td>
<td>Actual effort</td>
<td>750</td>
<td>325</td>
</tr>
<tr>
<td></td>
<td>Estimated effort</td>
<td>580</td>
<td>240</td>
</tr>
<tr>
<td>Accuracy measure used</td>
<td>Comparison of actual and estimate</td>
<td>Comparison of actual and estimate</td>
<td></td>
</tr>
<tr>
<td>Accuracy level achieved</td>
<td>30% underestimated</td>
<td>35% underestimated</td>
<td></td>
</tr>
</tbody>
</table>

used the data elicited in the interview to characterize the effort estimation activities performed in release planning meeting of both releases. The details are listed in the Table 5.11.

Company C

Company C is a small to medium sized Pakistan based software company established in the late 90s. It has around 50 employees based in Islamabad, Pakistan and develops mainly performance enhancements and database applications. C follows a customized version of Scrum in all projects. We collected effort estimation data, related to our taxonomy, from their release planning meetings of two teams working on two different products. We refer to the two teams as Team C1 and C2. A release normally consists of four iterations, and takes about four to six months to complete. An iteration is usually four to six weeks long.

Data was collected in a 1.5 hours interview of the development manager of the company who is responsible for overseeing all development teams of the company. He has the additional responsibility of managing the systems’ architecture as well. He leads both the release and sprint plan meetings of both products, which are also attended by
Table 5.11: Characterizing effort estimation activity of company B

<table>
<thead>
<tr>
<th>Dimension</th>
<th>Facet</th>
<th>External release</th>
<th>Internal release</th>
</tr>
</thead>
<tbody>
<tr>
<td>Context</td>
<td>Planning level</td>
<td>Release</td>
<td>Release</td>
</tr>
<tr>
<td>Main estimated activity</td>
<td>Implementation and Testing</td>
<td>Implementation and Testing</td>
<td></td>
</tr>
<tr>
<td>Agile Method(s)</td>
<td>Scrum</td>
<td>Scrum</td>
<td></td>
</tr>
<tr>
<td>Product domain</td>
<td>Telecom (billing)</td>
<td>Telecom (Billing)</td>
<td></td>
</tr>
<tr>
<td>Project setting</td>
<td>Co-located</td>
<td>Co-located</td>
<td></td>
</tr>
<tr>
<td>Estimation unit</td>
<td>Task</td>
<td>Task</td>
<td></td>
</tr>
<tr>
<td>Number of tasks/stories</td>
<td>137 tasks</td>
<td>227 tasks</td>
<td></td>
</tr>
<tr>
<td>Team size</td>
<td>6-10</td>
<td>6-10</td>
<td></td>
</tr>
<tr>
<td>Effort Size</td>
<td>Considered without any measure</td>
<td>Story points</td>
<td></td>
</tr>
<tr>
<td>Predictors</td>
<td>Team’s prior experience</td>
<td>Considered</td>
<td>Considered</td>
</tr>
<tr>
<td></td>
<td>Team’s skill level</td>
<td>Considered</td>
<td>Considered</td>
</tr>
<tr>
<td></td>
<td>Non functional requirements</td>
<td>Considered</td>
<td>Considered</td>
</tr>
<tr>
<td></td>
<td>Distributed teams’ issues</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td>Customer communication</td>
<td>Considered</td>
<td>Not considered</td>
</tr>
<tr>
<td>Estimation Technique name</td>
<td>Planning poker</td>
<td>Planning poker</td>
<td></td>
</tr>
<tr>
<td>Technique Type</td>
<td>Group estimation</td>
<td>Group estimation</td>
<td></td>
</tr>
<tr>
<td>Effort Estimate</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Estimate type</td>
<td>Point</td>
<td>Point</td>
<td></td>
</tr>
<tr>
<td>Actual effort</td>
<td>1685</td>
<td>4325</td>
<td></td>
</tr>
<tr>
<td>Estimated effort</td>
<td>1574</td>
<td>3726</td>
<td></td>
</tr>
<tr>
<td>Accuracy measure used</td>
<td>Comparison of actual and estimate</td>
<td>Comparison of actual and estimate</td>
<td></td>
</tr>
<tr>
<td>Accuracy level achieved</td>
<td>6% underestimated</td>
<td>16% underestimated</td>
<td></td>
</tr>
</tbody>
</table>

relevant development and testing leads. The first author conducted the interview and recorded the responses in an excel file. The excel file was later sent to the interviewee for entering missing information and validation of the recorded information.

Team C1 consists of 6 members, and is responsible for developing a performance enhancement application (referred as P1) for .Net and Java development environments, while team C2 consists of 10 members and is developing a NoSQL based database product (P2). We collected the effort estimation related data from the sprint plan meetings for the first three sprints of both products. The effort estimation activities of both teams C1 and C2 are characterized using our agile estimation taxonomy in Table 5.12. We only used the data of the first sprints for both products.

Summary

The interviewed practitioners found the taxonomy useful in characterizing their estimation activities. They noted that this offers a simple mechanism to document this information, which can be used subsequently. We asked them to suggest any changes or updates in the taxonomy when we sent the extraction sheet for validating extracted data. The following three extensions were suggested in agile estimation taxonomy.
Table 5.12: Characterizing effort estimation activity of company C

<table>
<thead>
<tr>
<th>Dimension</th>
<th>Facet</th>
<th>Team C1</th>
<th>Team C2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Context</td>
<td>Planning level</td>
<td>Sprint</td>
<td>Sprint</td>
</tr>
<tr>
<td></td>
<td>Main estimated activity</td>
<td>Implementation and Testing</td>
<td>Implementation and Testing</td>
</tr>
<tr>
<td></td>
<td>Agile Method(s)</td>
<td>Customized Scrum</td>
<td>Customized Scrum</td>
</tr>
<tr>
<td></td>
<td>Product domain</td>
<td>Performance enhancement NoSql Database app.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Project setting</td>
<td>Co-located</td>
<td>Co-located</td>
</tr>
<tr>
<td></td>
<td>Estimation unit</td>
<td>Task</td>
<td>Task</td>
</tr>
<tr>
<td></td>
<td>Number of tasks/stories</td>
<td>40 tasks</td>
<td>110 tasks</td>
</tr>
<tr>
<td></td>
<td>Team size</td>
<td>6</td>
<td>10</td>
</tr>
<tr>
<td>Effort</td>
<td>Size</td>
<td>Considered without any measure</td>
<td>Considered without any measure</td>
</tr>
<tr>
<td>Predictors</td>
<td>Team’s prior experience</td>
<td>Considered</td>
<td>Considered</td>
</tr>
<tr>
<td></td>
<td>Team’s skill level</td>
<td>Considered</td>
<td>Considered</td>
</tr>
<tr>
<td></td>
<td>Non functional requirements</td>
<td>Considered</td>
<td>Considered</td>
</tr>
<tr>
<td></td>
<td>Distributed teams’ issues</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td>Customer communication</td>
<td>Not considered</td>
<td>Not considered</td>
</tr>
<tr>
<td>Estimation</td>
<td>Technique name</td>
<td>Expert judgment</td>
<td>Expert judgment</td>
</tr>
<tr>
<td>Technique Type</td>
<td>Estimation unit</td>
<td>Hrs</td>
<td>Hrs</td>
</tr>
<tr>
<td></td>
<td>Estimate type</td>
<td>Point</td>
<td>Point</td>
</tr>
<tr>
<td></td>
<td>Actual effort</td>
<td>1500</td>
<td>Not available yet</td>
</tr>
<tr>
<td></td>
<td>Estimated effort</td>
<td>1450</td>
<td>4400</td>
</tr>
<tr>
<td></td>
<td>Accuracy measure used</td>
<td>Comparison of actual and estimate</td>
<td>Comparison of actual and estimate</td>
</tr>
<tr>
<td></td>
<td>Accuracy level achieved</td>
<td>Minor underestimation</td>
<td>Not available yet</td>
</tr>
</tbody>
</table>

- When the estimation is done as part of release planning, the taxonomy should also document the number and duration of the iterations/sprints in a release.

- While capturing the context, the taxonomy should also characterize the development mode, i.e., whether the agile team is engaged in a product or project based development.

- It is not enough to capture the project/product domain alone in one factor. The taxonomy should have separate facets for industry and application domains to fully characterize product domain. Few examples of industry are transportation, telecommunication, insurance, education etc; while examples of application domain are information system, command and control, avionics, data processing etc.

These suggested extensions are added in the context dimension of the agile estimation taxonomy. The Figure 5.6 presents the extensions as new facets in the context dimension. It is relatively easy to add or remove a facet in any dimension. This is one of the benefits of using the faceted classification structure in a taxonomy.
We have used the categories of software types proposed by Forward and Lethbridge [29] as example facet values for the application type facet in Figure 5.6. These are high level categories. More specific types of the software applications are also listed in their taxonomy.

5.5 Discussion and Validity Threats

The agile estimation taxonomy is based on the knowledge identified in our SLR [1] and survey [2] studies. Companies and researchers can introduce new dimensions and facets into the taxonomy. Since we used a faceted classification structure in designing this taxonomy, it is flexible enough to accommodate changes in the future.

In this section, we compare our taxonomy with an existing work and also describe limitations of our work.

5.5.1 Comparing agile estimation taxonomy with existing work

Trendowicz and Jeffery [25] provide a comprehensive review on software effort estimation covering aspects related to effort estimation foundations, techniques, drivers,
context and guidelines to select an appropriate technique in a specific context. We compare dimensions and facets of our taxonomy with the relevant aspects of the work by Trendowicz and Jeffery\textsuperscript{3} [25]. It is important to note the following points before moving further.

- Trendowicz and Jeffery do not present their work as a taxonomy or classification. However, the book covers many important aspects related to software effort estimation, and therefore provides an opportunity for comparing our taxonomy.

- The TJ estimation reference is about software effort estimation in general, is not meant specifically for ASD.

- The evidence identified and analyzed in our SLR [1] served as one of the inputs to the agile estimation taxonomy presented in this chapter. The SLR was published in 2014, and included only peer reviewed works published before December 2013. The TJ estimation reference was published as a book in 2014. Our taxonomy and the TJ estimation reference can therefore be considered as independent works.

**Effort estimation context**

The facets corresponding to the estimation context dimension of our agile estimation taxonomy are compared with the contextual factors in the TJ estimation reference in Table 5.13. There are six contextual facets that are not covered in the TJ estimation reference: planning level, main estimated activity, project/product domain, type and number of estimation entities, development mode and iteration/sprint details. The TJ estimation reference is general and therefore some factors that are more specific for ASD are not covered. Planning levels (i.e., release planning, iteration planning), estimation entities such as user stories and tasks, and concept of sprint/iterations are more specific to the ASD context.

There are two contextual factors listed in the TJ estimation reference that are not part of our taxonomy: The Programming language and the development type (new development or enhancement). These could be added as new facets in the estimation context dimension in our taxonomy as both provide additional information that might be helpful in characterizing the estimation context.

\textsuperscript{3}We refer to their work as “TJ estimation reference” in this comparison.
Table 5.13: Comparison of facets in the estimation context dimension

<table>
<thead>
<tr>
<th>Agile estimation taxonomy</th>
<th>TJ estimation reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Planning level</td>
<td>Not covered</td>
</tr>
<tr>
<td>Main estimated activities</td>
<td>Not covered</td>
</tr>
<tr>
<td>Agile method(s)</td>
<td>Covered as development life cycle in contextual factors list</td>
</tr>
<tr>
<td>Project/product domain</td>
<td>Not covered</td>
</tr>
<tr>
<td>Project setting</td>
<td>Partly covered in team structure as effect of scale factor</td>
</tr>
<tr>
<td>No. and type of estimation entity</td>
<td>Not covered</td>
</tr>
<tr>
<td>Team size</td>
<td>Team size covered as effect of scale factor</td>
</tr>
<tr>
<td>Application type</td>
<td>Covered as application domain in contextual factors list</td>
</tr>
<tr>
<td>Development mode</td>
<td>Not covered</td>
</tr>
<tr>
<td>Iteration duration and count</td>
<td>Not covered</td>
</tr>
</tbody>
</table>

**Effort estimation techniques**

The TJ estimation reference includes a very detailed classification of the effort estimation techniques. The classification is presented as a hierarchy which includes specific techniques as leaf nodes and their types at higher levels of the hierarchy. The facets of the estimation technique dimension are compared with this work in Table 5.14. All facets of the estimation technique dimension of our taxonomy are also covered in the TJ estimation reference.

Table 5.14: Comparison of facets in the estimation technique dimension

<table>
<thead>
<tr>
<th>Agile estimation taxonomy</th>
<th>TJ estimation reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Estimation techniques</td>
<td>Covered in the classification of effort estimation techniques</td>
</tr>
<tr>
<td>Type</td>
<td>Type is also covered in the classification of techniques</td>
</tr>
<tr>
<td>Accuracy measures</td>
<td>Covered as accuracy metric as part of handling uncertainty</td>
</tr>
</tbody>
</table>

**Effort predictors**

All facets of the predictors dimension of our agile estimation taxonomy are also present in the TJ estimation reference (See Table 5.15).

There are many effort predictors, described in the TJ estimation reference as common effort drivers, that are not part of our taxonomy: software, database and architecture complexity, requirements novelty, requirements stability, project constraints,
Chapter 5. An Effort Estimation Taxonomy for Agile Software Development

Table 5.15: Comparison of facets in the predictors’ dimension

<table>
<thead>
<tr>
<th>Agile estimation taxonomy</th>
<th>TJ estimation reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Size</td>
<td>Software size covered as scale factor</td>
</tr>
<tr>
<td>Team’s prior experience with similar tasks</td>
<td>Covered as application familiarity in effort drivers list</td>
</tr>
<tr>
<td>Team’s skill level</td>
<td>Covered as team capabilities in effort drivers list</td>
</tr>
<tr>
<td>Non functional requirements</td>
<td>Covered as required software quality in effort drivers list</td>
</tr>
<tr>
<td>No. of development sites</td>
<td>Covered as project distribution in effort drivers list</td>
</tr>
<tr>
<td>Customer communication</td>
<td>Covered as customer involvement in effort drivers list</td>
</tr>
</tbody>
</table>

Schedule pressure, tool usage, CASE tools and testing tools. These effort predictors are currently not part of our taxonomy as we could not find any evidence wherein these have been investigated as drivers of project effort in ASD. There is a need to investigate whether these generic effort drivers are relevant in an ASD context.

**Effort estimate**

Apart from estimation unit and accuracy level, other facets of this dimension of our taxonomy are described in the TJ estimation reference (see Table 5.16). These are, however, described in different places in the book. Some studies only report the accuracy level achieved in terms of percentages (e.g. MMRE 30%), and do not specify the actual and estimated effort. As for estimation units, in ASD some teams use the concept of ideal time as unit of effort instead of standard person hours/month units [12].

Table 5.16: Comparison of facets in the effort estimate dimension

<table>
<thead>
<tr>
<th>Agile estimation taxonomy</th>
<th>TJ estimation reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Estimated effort</td>
<td>Described in the principles of effort estimation</td>
</tr>
<tr>
<td>Actual effort</td>
<td>Described in the principles of effort estimation</td>
</tr>
<tr>
<td>Estimate type</td>
<td>Partly described as part of handling uncertainty</td>
</tr>
<tr>
<td>Estimate unit</td>
<td>Not covered</td>
</tr>
<tr>
<td>Accuracy level</td>
<td>N/A</td>
</tr>
</tbody>
</table>

**Summary**

The main findings of this comparison are described in the following:
• There are aspects that are specific to ASD, such as some facets in the context dimension. These are not part of TJ estimation reference, which is about effort estimation in general. Agile practitioners should therefore consult estimation works, such as our taxonomy, that are specific to ASD besides considering general works on effort estimation.

• There are many aspects, such as some cost drivers, included in the TJ estimation reference that are not covered by our agile estimation taxonomy. There is a need to investigate the relevance and importance of these effort drivers in an ASD context in the future research efforts.

5.5.2 Validity threats

We evaluated the proposed taxonomy by using it to characterize estimation cases from the literature and the industry. We also compared our taxonomy with a reference work by Trendowicz and Jeffery [25]. However, this study may still have some limitations. We discuss the validity threats using the categorization proposed by Petersen and Gen-cel [34] corresponding to two main phases of research: data collection and analysis.

• Descriptive validity is concerned with issues that may arise due to poor gathering and recording of data, which would result in the inaccurate description of the “truth”. The data identified and aggregated in the SLR [1] and the survey [2] was used as main input to the taxonomy designed in this study. In the SLR the studies were identified and analyzed following Kitchenham and Charters guidelines [33]. The survey instrument (questionnaire) was designed and reviewed iteratively to ensure readability and correctness. The survey was based on a sample of 60 agile practitioners. The validity threats and the corresponding mitigating actions for the SLR and the survey are described in [1] and [2] respectively.

• Interpretive validity is concerned with ensuring that reasonable conclusions are drawn based on the collected data, and the issues such as researchers’ bias do not lead to incorrect conclusions. We applied a taxonomy design method to organize the knowledge on effort estimation in ASD as taxonomy in a systematic manner. The taxonomy was evaluated by using it to characterize estimation cases from the literature and the industry. The taxonomy was also compared with a recent detailed review on software effort estimation [25].

• The taxonomy is applied to characterize ten agile effort estimation cases from the literature and six estimation sessions of five different agile teams. The results are generalizable to only those projects and teams that have similar contexts.
The authors of this study have experience in ASD and/or effort estimation. We followed a method to design the taxonomy in a systematic way. However, we still believe that the involvement of more experts could have further improved the completeness and utility of the proposed taxonomy.

5.6 Conclusions and Future Work

The design of taxonomies helps in structuring and communicating existing knowledge, and in advancing the research [4]. We organized the existing body of knowledge on effort estimation in agile software development as a taxonomy. The taxonomy was developed in a systematic way by following a taxonomy design method. The usefulness of the taxonomy was demonstrated by applying it on data extracted from literature and industry. The taxonomy was applied to characterize 10 selected estimation cases reported in the literature. It was observed that some studies do not report important information related to effort estimates, effort predictors used and the context in which the estimation activity was performed. The taxonomy can be used by researchers in reporting their findings in a consistent way to allow the audience to compare, analyze and aggregate their findings. The taxonomy was also applied to characterize the estimation activities of agile teams of three companies. The involved practitioners noted that the taxonomy provides a useful mechanism to document their estimation sessions.

The taxonomy has not been used during effort estimation sessions. We plan to develop a checklist instrument based on our agile estimation taxonomy with the aim to support agile practitioners during estimation sessions. The idea is to employ the checklist during sprint or release plan meetings while estimation is in progress. Practitioners could use the checklist to remind themselves of important factors that should be considered while estimating user stories or tasks. It will also support in documenting the knowledge used by agile practitioners to arrive at effort estimates, which otherwise would remain tacit in most cases. Our taxonomy could also be used as a tool in the development of a repository of the effort estimation knowledge, which could be used in the long term to better understand and improve the practice of effort estimation in ASD.
5.7 References


Chapter 5. An Effort Estimation Taxonomy for Agile Software Development


[34] K. Petersen and C. Gencel, “Worldviews, research methods, and their relationship to validity in empirical software engineering research,” in *The Joint Conference of the 23rd International Workshop on Software Measurement (IWSM) and...*
Chapter 5. An Effort Estimation Taxonomy for Agile Software Development

the 8th International Conference on Software Process and Product Measurement (Mensura), 2013.


Chapter 6

Effort Estimation in Large Scale Software Development: An Industrial Case Study

6.1 Introduction

Software is increasingly developed in globally distributed projects [3, 6–8]. Large, globally distributed projects are subject to particular challenges, like geographical, temporal, and cultural distances [1, 2], which make coordination and communication more difficult and may lead to more software defects [5] and schedule and budget overruns [9, 34].

The literature indicates that practitioners have fallen short of providing accurate and reliable effort estimates in both co-located and distributed projects. A review of surveys on effort estimation by Moløkken and Jørgensen revealed schedule and/or budget overruns in 60–80% of the covered projects [34]. Inaccurate effort estimates may lead to unrealistic schedules and/or budgets and might therefore constitute a significant business risk.

Estimation and planning becomes more challenging in the context of large-scale distributed projects [17, 36]. Project managers and analysts find it hard to create and estimate user stories in large-scale distributed projects [36]. In case of distributed agile teams, it becomes more difficult to achieve technical consistency due to the varying competence of the distributed teams [36]. Furthermore, multi-site development intro-
duces additional issues related to communication and coordination [2, 9, 36]. There is a need to investigate how these challenges contribute to the effort overruns.

While a lot of research has covered effort estimation in software development projects in general [12, 14, 35], little research has been reported on effort estimation in large-scale distributed agile projects [36]. More research is therefore necessary to better understand the factors that affect effort estimation in large-scale distributed agile development. Jørgensen and Shepperd [35] also identified the need to conduct more studies on estimation methods frequently used (i.e., expert estimation) in the software industry.

To address this gap, we conducted a longitudinal exploratory industrial case study, to investigate how effort estimation is performed in large-scale distributed agile projects, and to analyze factors (cost drivers) impact the accuracy of the effort estimates. Identifying these factors would benefit managers at the case company, and in similar contexts, to improve their estimation and planning processes. For the context of this work, we adopt Dikert et al.’s definition of large-scale projects, i.e. projects that involve at least 50 people or more than 10 teams [36]. The case is a large-scale distributed agile software project with a large amount of complex legacy code at Ericsson AB\(^1\), a Swedish company that develops telecommunication-related products.

In this case study, we address the following research questions. The context for all research questions is large-scale distributed agile projects comprising a large amount of complex legacy code.

**RQ1:** How is effort estimation carried out?

**RQ2:** How accurate are the effort estimates?

**RQ3:** Which factors impact the accuracy of the effort estimates?

**RQ4:** How do the identified factors impact the accuracy of the effort estimates?

The main contributions of this study are described in the following:

- A description of a two-stage effort estimation process and its coordination in a large-scale distributed agile project.
- An analysis of the impact of re-estimation in a large-scale distributed agile project.
- An identification of the factors that affect effort estimation accuracy in a large-scale distributed agile project.

\(^1\)www.ericsson.com
• An analysis of the factors that affect effort estimation accuracy in a large-scale distributed agile project.

The remainder of the chapter is organized as follows. Related work is presented and discussed in Section 6.2. Section 6.3 describes the research methodology employed in this case study. The results of the case study are presented in Section 6.4 and discussed in Section 6.5. Threats to validity are discussed in Section 6.6. Finally, Section 6.7 presents our conclusions and view on future work.

6.2 Related work

Effort estimation has been on the agenda of software engineering researchers for many years. It becomes more challenging in the context of large-scale distributed projects involving coordination between several stakeholders and teams [36]. The teams may be geographically distributed, which adds delays due to communication and coordination challenges [44]. In context of large-scale agile development, long-term planning and estimation are more challenging [17, 36]. To provide insight to the practitioners working in such contexts, there is a need to investigate how effort estimation process is carried out at different stages of development, and how factors related to scale affect effort estimation accuracy.

A review by Jørgensen [37] covering empirical research published 1990–2002, showed that expert estimation is the most commonly used estimation strategy in software projects. Jørgensen and Shepperd [35], furthermore noted that the proportion of studies on expert judgment based estimation is increasing and that the available evidence does not support that the use of formal estimation models improves the estimation accuracy. More recent reviews [12, 14] on effort estimation in agile and global software development contexts also showed that the most frequently applied estimation methods are the ones that rely on subjective assessment by experts, such as planning poker, analogy, and use case points. In the case studied in this study as well, an expert judgment based approach is used to estimate development effort. However, the initial effort estimate is revised later at the analysis stage in the light of additional information.

In a systematic literature review on effort estimation in global software development, Britto et al. [14] identified number of cost drivers that are relevant in the context of globally distributed projects, e.g., socio-cultural, geographical and temporal distances. However, none of the primary studies included in their SLR performed an in-depth investigation of the factors that impact the accuracy of the effort estimates. Furthermore, primary studies did not describe in detail the effort estimation process used in their contexts. The SLR managed to include only five primary studies and
therefore stressed the need to conduct more empirical studies in the area.

Usman et al. [12] identified team skills and prior experience as the most frequently used cost drivers in studies on effort estimation in agile projects through a systematic literature review. However, none of the included primary studies were performed in the context of a large-scale distributed project involving multiple agile teams. Tanveer et al. [18] also showed that the accuracy of effort estimation in agile projects is affected by factors such as developers’ knowledge and experience, and the complexity of changes. This study was based on observations and interviews with three agile teams working on three different web applications. It, however, did not involve collection and analysis of quantitative data (e.g., actual and estimated effort). Furthermore, In each project, only a single team was responsible for developing a web application. In a small agile project, Lenarduzzi et al. [16] replicated a case study on functional size measures and effort estimation. Their results show that the effort estimated by developers was more accurate than the estimates obtained through functional size measures.

Evbota et al. [17] investigated the challenges of scaling up the planning game to large-scale agile development through an interview-based case study at Ericsson. The interviewees found it very challenging to arrive at long-term estimates, that are applicable for several months, due to a very large size of the product backlog items at that level. This also made them skeptical about effort estimation in general. The interviewees also suggested that it is important to monitor and manage the estimation sessions to avoid lengthy discussions without any progress. This study was based on interviews only and did not involve the collection and analysis of the quantitative data to analyze effort estimates. In another interview-based study on project management in global software development, Björndal et al. [44] identified that challenges related to the multi-site development (e.g., communication and coordination) result in delays due to extra work.

From the review of the existing literature, we identified that expert judgment based effort estimation is the most commonly used approach in the software industry. We have not found any empirical study in the context of a large-scale distributed project that has looked into the affect of re-estimation at a later stage on effort estimates. We have not found any work that has investigated effort estimation in a large software project, involving multiple agile teams at different levels of maturity, over a long time period. Previous empirical studies on estimation and planning in agile and global software development discussed above, such as [17, 43, 44], did not use quantitative data to triangulate the results based on interviews and observations.

This study fills the aforementioned gaps through a longitudinal exploratory case study, covering almost three years of project development, to study effort estimation from multiple perspectives, such as the impact of a multi-staged estimation process and various other factors (e.g., maturity of the agile teams involved, multi-site development,
impact of size) on effort estimates. This study used a combination of quantitative and qualitative data to perform this investigation.

6.3 Research methodology

To address the research questions, we have conducted a longitudinal exploratory industrial case study [10]. In this section, we describe the case, the unit of analysis and the data collection, preparation, and analysis processes.

6.3.1 Case description

The case investigated in the study reported in this chapter is a large-scale project with an increasing degree of global distribution during the studied period. The actual project was selected through convenience sampling in consultation with company representatives.

The project is concerned with the development and maintenance of a large telecommunication software product at Ericsson. The product originated in Sweden and has evolved for over 20 years and comprises a considerable amount of legacy code. Many technical and methodological changes were introduced during this time, such as changing the programming language from C++ to Java and changing the software development methodology from plan-driven to agile. The product is part of a large business solution that comprises other products.

During the period covered in this study, the case involved 188 employees (15 product-level architects, 134 developers working in 24 teams and 15 other supporting roles) distributed across Sweden (eight software development teams), India (eleven software development teams), Italy (one software development team), USA (one software development team), Poland (two software development teams) and Turkey (one software development team).

The offshore locations were added in response to the growing demands for resources and to implement market-specific customizations. The last expansion was to Poland, where two teams were on-boarded in early 2016. An expansion to India took place between late 2014 and late 2015, where 10 teams were on-boarded in total. The site located in Turkey was decommissioned in 2014.

The developments teams are cross-functional and use agile practices in their daily work. Each team has from four to seven developers and a design lead, who is a senior developer. Due to the scale and level of distribution of the case, project managers use a mix of agile and plan-driven practices to manage and coordinate the work of the software development teams.
All software development teams receive an end-to-end responsibility for designing and implementing a task, which can be product customizations, trouble report fixing, product improvements, standardizations of market features and business use cases. These types of tasks resemble independent projects; they have specific start and end dates, responsible teams (one or more) and expected results. They can also be substantial in extent; the duration of a product customization, for example, varies from one to six months.

Software architects located in Sweden support all teams by responding to questions related to the product software architecture and by providing feedback on the teams’ work through code reviews. In urgent or particularly complex situations, the software architects also participate in actual coding.

6.3.2 Unit of analysis

The unit of analysis is the effort estimation process employed in the case to estimate the effort required to develop and test the Product Customization (PC) tasks. The effort estimation is performed at two stages in the case: at the quotation stage, and later at the analysis stage. The effort estimation process is described in detail in Section 6.4. In both stages, an expert judgment based approach is employed to estimate the effort required to develop and test the PC tasks.

6.3.3 Data collection

The data collected and analyzed in this paper is related to PC tasks, which are driven by Ericsson customers. PCs are negotiated as fixed price contracts; failing to accurately estimate their effort may lead to budget overruns for the company. Accurate effort estimates for PC projects are therefore particularly important. The data collected and analyzed in this paper includes PC tasks carried out between 2013 and 2016 (approximately three years). This data constitutes a subset of the data related to the case, which was made available for us by the company.

To support methodological and data triangulation [10], we used the following data collection methods:

- **Archival research** – We analyzed managerial documents (plans, progress reports, time reports, effort estimation spreadsheets, process descriptions and team setup reports) of 60 PCs involving 18 different teams in total (9 in India, 7 in Sweden, 1 in Italy and 1 in the USA). Time report of a PC contains the self-reported hours spent by each team member. We extracted the following data from these documents: effort estimation process description, effort estimates,
actual effort (in hours) of all involved in the development of a PC from the relevant time report, teams involved, team maturity, size of teams and customer type.

- **Unstructured individual interviews** - We held several unstructured interviews with a project manager to verify the consistence of the collected data.

- **Semi-structured individual interviews** – We interviewed a system managers, a project manager, a software architect, and a design lead (senior developer of the team having having architectural knowledge) to collect details about the effort estimation process employed in the case. We also talked to them to clarify the results of our data analysis, such as the main reasons that lead to the identified effort overruns. The interviewee sample was selected by the unit manager, who was our contact person at Ericsson for this investigation. Unit manager ensured to include in the interviewee sample a representative from all roles involved in the estimation processes. More details about the interviews are presented in Table 6.1.

Table 6.1: Description of semi-structured interviewee sample

<table>
<thead>
<tr>
<th>Role</th>
<th>Approach</th>
<th>Duration</th>
<th>Years in Ericsson</th>
</tr>
</thead>
<tbody>
<tr>
<td>System manager</td>
<td>face-to-face interview</td>
<td>70 mins</td>
<td>9 years</td>
</tr>
<tr>
<td>Architect</td>
<td>face-to-face interview</td>
<td>45 mins</td>
<td>7 years</td>
</tr>
<tr>
<td>Project manager</td>
<td>face-to-face interview</td>
<td>1 hour</td>
<td>15 years</td>
</tr>
<tr>
<td>Design lead</td>
<td>interview via Skype</td>
<td>45 mins</td>
<td>10 years</td>
</tr>
</tbody>
</table>

Table 6.2 maps the research methods to the research questions. Note that we used both data collection methods for each of the research questions.

### 6.3.4 Data preparation

We used a significant amount of data from different sources that was extracted using different data collection methods. To increase data reliability, we asked people involved with the data point (e.g., software architects or project managers) for clarification whenever we identified an issue or inconsistency with a data observation and corrected the data observation, if necessary.
Chapter 6. Effort Estimation in Large Scale Software Development: An Industrial Case Study

Table 6.2: Mapping RQs with the used data collection methods

<table>
<thead>
<tr>
<th>RQ</th>
<th>Main method</th>
<th>Secondary method</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Interviews</td>
<td>Archival research</td>
</tr>
<tr>
<td>2</td>
<td>Archival research</td>
<td>Interviews</td>
</tr>
<tr>
<td>3</td>
<td>Archival research</td>
<td>Interviews</td>
</tr>
<tr>
<td>4</td>
<td>Archival research</td>
<td>Interviews</td>
</tr>
</tbody>
</table>

Furthermore, before running the quantitative part of the data analysis, it was necessary to aggregate the collected data. To do so, we created R scripts\(^2\), which were used to create a unified data set.

### 6.3.5 Data analysis

To analyze the quantitative data (archival research), we calculated descriptive statistics, plotted charts to graphically identify trends in the data and employed inferential statistics (hypothesis testing and regression analysis) to answer RQ2–RQ4. To answer RQ1 and RQ3, we analyzed the qualitative data using lightweight qualitative analysis (coding) [11].

Before analyzing the accuracy of the effort estimates, we analyzed existing accuracy metrics. Traditionally, the Magnitude of Relative Error (MRE) is used frequently to calculate estimation accuracy. However, during the last decade, MRE has been criticized [22–24] for its uneven treatment of under- and over-estimation. Balanced Relative Error (BRE) and Balanced Relative Error Bias (BREbias) are more balanced metrics, i.e. it evenly balances the over and underestimation. Due to this reason, BRE and BREbias have been used in many recent studies on software effort estimation [25, 26]. BRE just calculates the magnitude of the estimation error, while BREbias measures both the size and the direction (over- or under-estimation) of the estimation error. Therefore, we used BRE (Equation 6.1) and BREbias (Equation 6.2) in our analysis of the effort estimates’ accuracy and bias.

\[
BRE = \frac{|\text{actual effort} - \text{estimated effort}|}{\min(\text{actual effort}, \text{estimated effort})} \quad (6.1)
\]

\(^2\)cran.r-project.org
\[ \text{BREbias} = \frac{(\text{actual effort} - \text{estimated effort})}{\min(\text{actual effort}, \text{estimated effort})} \] (6.2)

Before selecting an appropriate statistical test to analyze our results, we applied a Shapiro-Wilk normality test on BRE and BREbias values, which indicated that the tested data are not from a normally distributed population. Therefore, we applied non-parametric tests Wilcoxon signed-rank test [20] for RQ2 and Kruskal-Wallis [19] and Mann-Whitney [21] for RQ4. For the same reason, we used a non-parametric measure of Cliff’s delta [33] to measure effect size. Table 6.3 lists the tests applied, along with their purpose, in performing the data analysis.

Table 6.3: Statistical tests used in data analysis

<table>
<thead>
<tr>
<th>No.</th>
<th>Test</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Normality tests [29]</td>
<td>To check for the normality of the data</td>
</tr>
<tr>
<td>2</td>
<td>Wilcoxon signed-rank test [20]</td>
<td>To test if the difference between the two estimates is statistically significant</td>
</tr>
<tr>
<td>3</td>
<td>Kruskal-Wallis test [19]</td>
<td>To test if the difference in estimation accuracy of four PC sizes is statistically significant</td>
</tr>
</tbody>
</table>
| 4   | Mann-Whitney test [21]             | 1) To test if the difference in estimation accuracy of PCs of high and normal priority is statistically significant  
                           | 2) To test if the difference in estimation accuracy of PCs developed by more and less mature teams is statistically significant |
| 5   | Cliff’s delta [33]                 | To measure effect size                                 |
| 6   | Durbin-Watson test [31]            | To check for the independence of residuals (regression assumption) |
| 7   | Breusch-Pagan and Koenker tests [30]| To check for the constant variance in residuals (regression assumption) |
| 8   | Cook’s distance [32]               | To perform outliers diagnostics in regression analysis |

6.4 Results

6.4.1 RQ1: How is effort estimation carried out?

The aim of RQ1 is to explain how effort estimation processes are carried out in this very large distributed project. As shown in Figure 6.1 and 6.2 several people in different roles work collaboratively to perform effort estimation. First, we briefly describe these roles and the tasks that they perform in the effort estimation processes.

- Customers: Ericsson customers demand new features in the product and have to
Figure 6.1: QE process for PC (Product Customization) tasks
Figure 6.2: AE process for PC (Product Customization) tasks
approve the quoted price before development work on a product customization (PC) can begin.

- Customer Unit: Customer unit is part of Ericsson and is responsible for negotiations with the customers. It initiates the request for a new PC by specifying the requirements received from customers.

- System Managers: As stated previously, the product is part of a larger system comprising several products. System managers work at the overall system level. They receive request for new PCs from the customer unit, propose high level solutions and coordinate with the project managers and the architects of each concerned product in the system.

- Project Manager: Each product has a project manager, who is responsible for managing development teams across different sites and for planning, scheduling and coordinating the work on the PCs.

- Software Architecture Team (SAT): Each product has a team of architects, which is responsible for managing the evolution and integrity of the architecture. Besides proposing an initial solution and estimates for the requested PCs, they also provide design support, when required, to the development teams.

- Development Team: The development team is responsible for actually implementing and verifying the PCs. Mature development team have senior developers, who act as design leads. Design leads collaborate with the members of the SAT team during the estimation process.

The system managers, project managers, SAT and the development teams are part of the development unit. To fully understand the estimation processes in the case, we interviewed a representative from each of these roles. The interviewee sample (see Table 6.1) was selected by the unit manager, who was the contact person at Ericsson for this study.

Effort estimation is carried out at two levels in the case project, both using an expert judgment based estimation approach. First, a high level *quotation estimate* is prepared when a request for a new PC is initially analyzed. At this stage, it is not known which development team will actually develop the PC. Next, in a more detailed analysis phase, a more refined *analysis estimate* and a solution are proposed. At this stage, the development team that is going to develop the PC is known in most cases. We describe these estimation processes in the following.
**Quotation Estimate (QE)**  
QE process consists (depicted in Figure 6.1) of the following steps:

1. The process starts when a new PC request is initiated by the customer unit. The customer unit interacts with customers, and specifies the customer requirements.

2. The system manager scans the new PC requests for an initial analysis. The PCs are assigned to relevant system managers depending upon their availability. In this initial analysis step, the system manager performs the following tasks:
   
   - The system manager selects PCs with relatively clear requirements for further analysis, the remaining ones are sent back for further clarification.
   
   - The system manager compares the selected PCs with the previously developed PCs to identify any potential for reuse. In case a PC request is identical to a previously developed PC, the customer unit is asked to order a resell instead of a new implementation. This step ensures the better utilization of the already developed product capabilities.

3. The assigned system manager analyzes the requirements, and suggests a high level solution. The solution is shared with the SAT of the concerned product for further analysis.

4. The SAT is also required to provide the QE. Normally, One member of the SAT leads this analysis. The QE is an interval estimate wherein the interval is marked by three points: min, max and average most likely estimate in hours. These intervals represent the size and complexity of PCs and are referred as small, medium, large and extremely large. The min, max and average values for each interval are grounded in empirical evidence in the case, which has been going on for over 20 years. The SAT lead, instead of providing a single point estimate in hours, classifies the assigned PC into one of these interval estimates.

5. The QE along with the high level solution are shared with the customer unit.

6. The customer unit approves or disapproves the QE and the high level solution.

7. For the approved PC requests, system managers initiate the phase, i.e., preparation of the detailed solution proposal and the corresponding revised effort estimates.

This entire QE process, from the time customer unit requests a new PC (step 1) till its approval (step 6), take up to four to five days. It is important to note that at the
Chapter 6. Effort Estimation in Large Scale Software Development: An Industrial Case Study

quotation stage the goal is to support the customer unit in deciding whether there is a strong business case in moving to the next phase of the PC development. The focus of effort estimation at this stage is on providing a best cost indication without spending too much time. Therefore, small estimation errors are not a problem at this stage.

**Analysis Estimate (AE)** AE process (depicted in Figure 6.2) consists of the following steps:

1. This process is initiated when a system manager selects a PC, whose quotation estimates have been approved by the customer unit, for this analysis phase.

2. The system manager asks the SAT for the relevant products to prepare their respective designs, and to provide effort estimates in person hours.

3. One SAT member leads the work on each PC. At this point, it is usually known which development team is going to develop the PC. The SAT lead specifies the design in detail. The SAT lead also estimates the effort, in person hours, required to develop and verify the PC. The SAT lead collaborates with the design lead of the relevant development team if he/she has good product and architectural knowledge (maturity level C and D, see Subsection 6.4.3 for details). The team’s design lead provides his/her input on the solution proposal and the effort estimates.

4. The SAT lead presents the design and associated effort estimate to other SAT members for further discussion, and revise design and effort estimate if required. The revised estimate and design are shared with the project manager for his/her input.

5. The project manager reviews the estimate, and adjusts it, if required, based on the recent productivity and capacity of the concerned development team. Project manager has the knowledge of the teams’ available capacity and recent performance. Revised estimate is shared with the system manager for finalizing it with the customer unit.

6. The system manager forwards the design solution proposal and corresponding effort estimate to the customer unit for approval.

7. The customer unit evaluates the design and the corresponding estimates to determine if the proposal can be approved.

8. The system manager receives the approved proposals from the customer unit, and initiates the subsequent phases of the PC development.
6.4.2 RQ2: How accurate are the effort estimates?

The question aims to analyze the accuracy levels of the two estimates (i.e. QE and AE). This analysis is performed from the following two aspects:

- **Overall trend analysis**: To identify the dominant trends, we analyze how frequently the effort is over/under estimated corresponding to different estimation error levels (e.g., 25% to 50%). In this part, we do not use the mean or median of BRE and BREbias, as a single number (i.e. mean or median) does not fully convey the trend.

- **Estimation error analysis**: We analyze the accuracy of the effort estimates using BRE\(^3\) and BREbias, and compare the accuracy of the two estimates (QE and AE) to establish whether the re-estimation at the analysis stage (AE) improves the estimation accuracy over the original estimate (QE).

**Overall trend analysis**

As described in Subsection 6.4.1, the QE is an interval estimate. Our results show that actual effort falls within the estimated interval in 26.7% of the cases (16 PCs).

Table 6.4 summarizes for both estimates, the frequency of over/under/accurately estimated PCs corresponding to different levels of estimation error expressed here as percentages. For QEs the PCs are almost evenly distributed in the three categories (over- (30%), under (36.7%- and accurately (33.3%) estimated PCs), while in case of AEs underestimation is more common (50% PCs) as compared to overestimation (16.7%). The gross overestimation instances are more common for QEs, where relatively less details about PCs are available, and the team that is going to develop the PC is not known.

<table>
<thead>
<tr>
<th>Type</th>
<th>No. of overestimated PCs by</th>
<th>No. of accurately estimated PCs</th>
<th>No. of underestimated PCs by</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Total</td>
<td>Above 100%</td>
<td>51–100%</td>
</tr>
<tr>
<td>QE</td>
<td>18 (30%)</td>
<td>10</td>
<td>3</td>
</tr>
<tr>
<td>AE</td>
<td>10 (16.7%)</td>
<td>4</td>
<td>2</td>
</tr>
</tbody>
</table>

\(^3\)for QE we used mid value, which is the most likely estimate, to compute BRE.
Estimation error analysis

In this part we analyze and compare the accuracy of the two effort estimates using BRE and BREbias measures. The results are displayed in Table 6.5. The QEs’ magnitude of error (mean BRE 1.26, median BRE 0.69) is considerably higher than the AEs’ magnitude of error. The mean and median BREbias values for both estimates show that the estimates are biased towards underestimation, i.e. optimism. The underestimation bias, however, is more significant for AEs (mean BREbias 0.41, median BREbias 0.26).

Table 6.6 displays the results of a Wilcoxon signed-rank test to see if the differences between the two estimates, for our sample of 60 PCs, are statistically significant. The statistical tests are conducted in SPSS, and we report all relevant data from SPSS, and also the effect size measure (Cliff’s delta), when applicable. The results show a statistically significant difference (p = 0.04) between the BREbias of the two estimates with small to medium size effect (Cliff’s delta 0.14). However, the p-value of the test for the BRE of the two estimates is slightly over 0.05 (0.06) and therefore not statistically significant. The results in Table 6.5 and 6.6 show that the AEs are more accurate (median BRE 0.49) than the QEs, but contain significantly higher optimism (i.e., underestimation) bias.

<table>
<thead>
<tr>
<th>BRE</th>
<th>BREbias</th>
<th>BRE</th>
<th>BREbias</th>
</tr>
</thead>
<tbody>
<tr>
<td>QE</td>
<td>1.26</td>
<td>AE</td>
<td>0.81</td>
</tr>
<tr>
<td></td>
<td>0.69</td>
<td></td>
<td>0.49</td>
</tr>
<tr>
<td>Std. Deviation</td>
<td>1.48</td>
<td>0.97</td>
<td>1.93</td>
</tr>
</tbody>
</table>

Table 6.6: Testing the two estimates (QE and AE) for significant differences

<table>
<thead>
<tr>
<th>BRE AE – BRE QE</th>
<th>BREbias AE – BREbias QE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Negative Ranks</td>
<td>34 33.32 1133 0.06 -0.15</td>
</tr>
<tr>
<td>Positive Ranks</td>
<td>25 25.48 637 0.14</td>
</tr>
<tr>
<td>Ties</td>
<td>1 -1.87 .06 -0.15</td>
</tr>
<tr>
<td>Total</td>
<td>60</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>BRE AE – BRE QE</th>
<th>BREbias AE – BREbias QE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Negative Ranks</td>
<td>20 30.80 616 0.04 0.14</td>
</tr>
<tr>
<td>Positive Ranks</td>
<td>39 29.59 1154</td>
</tr>
<tr>
<td>Ties</td>
<td>1 -2.03 .04 0.14</td>
</tr>
<tr>
<td>Total</td>
<td>60</td>
</tr>
</tbody>
</table>
6.4.3 RQ3: Which factors impact the accuracy of the effort estimates?

Based on our initial scoping discussion with the unit manager, and later during interviews with the participants involved in the estimation processes, we identified the following factors that potentially impact the accuracy of the two effort estimates:

- **PC size** – Understandably, the estimation of larger PCs is perceived to be more challenging. The interviewees observed that PC size, measured in actual effort (work hours), beyond a certain level brings with it all the complexities of scale, and therefore impact the accuracy of the effort estimates. Using the actual effort spent, the PCs were classified based on the classification mechanism described in Section 6.4.1 into four types: small, medium, large and very large. Impact of project size on effort and schedule overruns has been investigated in other studies (cf., [27, 41]) as well. In these studies also, actual effort was used to classify projects into small and large sized projects.

In our regression analysis, the variable PC size is used as an ordinal variable with values 0 (small), 1 (medium), 2 (large) and 3 (very large).

- **Customer priority** – A set of PCs (26) in our sample belong to one large customer project. The case company ensures a dedicated team capacity for the implementation of these PCs, in return for the guarantee that the company will be paid for the work done on these PCs. The provision of dedicated capacity for these PCs is meant to prioritize them during planning phase. They are not required to wait for the availability of capacity to start the work. Therefore, the work is initiated relatively quickly on most of the high priority PCs after the quotation approval by the customer unit.

These 26 PCs are categorized as PCs having a customer with high priority, while the remaining 34 as PCs from customers with a normal priority.

- **Maturity of the development team** – The development teams, working across the globe, have varying levels of maturity. As teams become more mature, they work more independently and need less support from the product level architects.

The maturity of the teams investigated in this study was measured by the software architects involved in the case in one of our previous investigations [28]. They used the maturity matrix formalized in Britto et al. [28], which has the following levels:

  - *Maturity level A* – Teams at level A have very little understanding about the product’s code and architecture. Therefore, they need much support and
Chapter 6. Effort Estimation in Large Scale Software Development: An Industrial Case Study

guidance from the architects, even for less complex PC projects. Typically, newly on-boarded teams are at level A.

- **Maturity level B** – Teams at level B can implement non-complex tasks independently without much help from the product level architects. The architects still review most of the design and code.

- **Maturity level C** – Teams at level C have a good architectural understanding and are able to implement complex solutions. These teams perform code reviews and approvals independently. When critical components are affected, the product level architects are responsible for approval. The architects also support these teams with the design of the technical solution.

- **Maturity level D** – Teams at level D are very experienced, and are able to autonomously develop complex solutions that affect or include critical components. Their code does not need the approval from the product level architects.

In our sample of 60 PCs, 19 PCs were developed by level A teams, 12 by level B teams, 26 by level C teams, and 3 by level D teams. The variable team maturity is used as an ordinal variable in our regression analysis with the four values: 0 (maturity level A), 1 (maturity level B), 2 (maturity level C), 3 (maturity level D).

- **Multi-site development** – In our sample, six PCs were developed in a multi-site arrangement wherein development teams in different geographical locations collaboratively worked together to develop the PCs. Multi-site development is also perceived as a factor impacting the actual effort spent, and thus the accuracy of the effort estimates. However, we decided not to include this factor in our regression analysis for two reasons: 1) Very few PCs were in this category, and 2) the multi-site PCs were particularly large. It would therefore not be possible to differentiate the impact of PC size and multi-site setting. We, however, analyze the impact of multi-site development in more detail in Section 6.4.4.

We performed multiple linear regression to investigate whether the factors identified above significantly impact the accuracy of the effort estimates. Our goal is not to predict the accuracy and bias of the two effort estimates. We used simple linear models to understand the relationship between the effort estimates’ accuracy and bias with the factors described above. We applied stepwise regression in SPSS using backward elimination with an alpha value of 0.1 to remove variables.
Regression model of quotation estimates (QEs)

The results of the regression analysis showed that PC size (p < 0.005) and team maturity (p = 0.007) have a statistically significant relationship with BREbias of QEs (see Table 6A.1 and 6A.2 in Appendix A for details), while PC priority (p = 0.863) has no such relationship.

Given the details in Table 6A.2, the regression model is described as:

\[
\text{BREbias of QE} = -0.73 + 1.22 \times \text{PC Size} - 0.56 \times \text{Team Maturity} \quad (6.3)
\]

Model (6.3) suggests that the QEs are overestimated by 73% for the small size PCs (PC size = small = 0) and teams at lowest maturity level (Team Maturity = Level A = 0). However, increases in PC size increase the BREbias of QEs, i.e. underestimation bias, while increases in team maturity decrease the BREbias of QEs, i.e. overestimation bias. In other words mature teams are more likely to complete the work within the estimated time. These factors are investigated in detail in Section 6.4.4 and are further discussed in Section 6.5.

Model (6.3) is the one with highest value (0.40) of adjusted R\(^2\) (see Table 6A.3 in Appendix A). The value of adjusted R\(^2\) does not necessarily increase when more predictors are added in the model, and thus is a suitable measure in identifying appropriate predictors for a model. Including variable PC Priority (p = 0.863) slightly reduces the value of adjusted R\(^2\).

Regression model of analysis estimates (AE)

Our regression analysis showed that PC size (p = 0.001) and priority (p = 0.028) have a statistically significant relationship with BREbias of AEs, while team maturity (p = 0.929) has no such relationship. The regression details are shown in Tables 6B.1 and 6B.2 in Appendix B.

Given the details in Table 6B.2, the regression model is described as:

\[
\text{BREbias of AE} = -0.07 + 0.57 \times \text{PC Size} - 0.69 \times \text{Customer Priority} \quad (6.4)
\]

Model (6.4) shows that for small sized (PC size = small = 0) PCs with normal priority (customer priority = normal = 0), effort is slightly overestimated by 7%. An increase in PC size (PC size = medium = 1 or PC size = large = 2 or PC size = very large = 3) results in an increase in underestimation bias. However, the negative sign of priority in model (6.4) indicates that an increase in priority from normal to high (i.e. customer priority = normal = 0 to customer priority = high = 1) decreases the underestimation bias or leads to an overestimation bias. In other words, the high priority PCs
incur lesser magnitudes of effort overruns with respect to the analysis stage estimates. However, PC size is still a huge factor even with high priority PCs. When PC size becomes large or very large, the overall result is still a considerable underestimation bias. These factors are further analyzed in detail in Section 6.4.4, and further discussed in Section 6.5. Team maturity is not significantly impacting BREbias of AEs, since the estimators at the analysis stage are normally aware of which teams are going to develop a PC. Therefore, Team maturity already is accounted for in the analysis stage estimates.

Model (6.4) is the one with highest value (0.40) of adjusted $R^2$ (see Table 6B.3 in Appendix B). Including variable Team Maturity ($p = 0.929$) reduces the value of adjusted $R^2$.

**Regression assumptions**

The examination of the residuals of both models revealed that the regression assumptions are not violated, with one minor exception. The histograms and normal plots show some minor deviations from the normal distribution (see Figures 6A.1–6A.3 and 6B.1–6B.3 in the Appendix). Besides visual analysis using these plots, we also tested these assumptions using relevant tests.

- Independence of residuals was tested using a Durbin-Watson test [31]. The results for both models were in the required range of 1.5 and 2.5 (2.01 for model (6.3) and 2.09 for model (6.4)).

- Constant variance in residuals (Homoscedasticity) was tested using a Breusch-Pagan and Koenker tests [30]. For both models, the null hypothesis of homoscedasticity was not rejected due to $p$-values $> 0.05$: Breusch-Pagan (model (6.3), $p = 0.08$; model (6.4), $p = 0.41$), Koenker (model (6.3), $p = 0.28$; model (6.4), $p = 0.75$).

- Normal distribution of residuals was tested using a Kolmogorov-Smirnov test [29]. For both models the tests showed that the residuals are normally distributed, i.e. the null hypothesis of normality could not be rejected (model (6.3), $p = 0.20$; model (6.4), $p = 0.08$). According to a Shapiro-Wilk test, the residuals of model (6.3) are normally distributed ($p = 0.17$), but not the residuals of model (6.4) ($p = 0.003$).

Furthermore, tolerance values (see Table 6A.2 and 6B.2 in Appendix A and B respectively) show the absence of multicollinearity for both models. As for the outliers diagnostics, only one observation was outside the three sigma limit in case of model
The maximum cook’s distance [32] for any observation was 0.11 (much less than critical value of 1), indicating that there are no highly influential observations. Likewise for model (6.4), there is only one observation that is outside the three sigma limit, and the maximum cook’s distance was 0.31.

Besides BREbias, we also attempted to apply regression analysis using BREs of both QEs and AEs. In case of BRE of QEs, regression assumptions of homoscedasticity and normality were violated. For AEs, the analysis did not reveal any significant relationship between the outcome and independent variables.

### 6.4.4 RQ4: How do the identified factors impact the accuracy of the effort estimates?

This question aims to investigate in detail how the factors, identified as statistically significant above, impact the accuracy of the effort estimates.

**PC size**

The 60 PCs included in this study are of varying sizes in terms of actual effort spent. These PCs are divided into four categories (see Table 6.7) using the intervals of QEs (see Section 6.4.1). The aim of this question is to investigate in depth how PC size impacts the accuracy of the two effort estimates.

Table 6.7: Mean and median actual effort for all 60 PCs in work hours grouped by PC size

<table>
<thead>
<tr>
<th>PC size</th>
<th>No.</th>
<th>Mean (hours)</th>
<th>Median (hours)</th>
<th>Std. Deviation (hours)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Small</td>
<td>10</td>
<td>284.9</td>
<td>290</td>
<td>76.10</td>
</tr>
<tr>
<td>Medium</td>
<td>28</td>
<td>833.1</td>
<td>699</td>
<td>416.2</td>
</tr>
<tr>
<td>Large</td>
<td>12</td>
<td>1931.8</td>
<td>1561</td>
<td>661.9</td>
</tr>
<tr>
<td>Very Large</td>
<td>10</td>
<td>5735.5</td>
<td>4811.5</td>
<td>3362.3</td>
</tr>
</tbody>
</table>

**Quotation Estimate (QE)**  Figure 6.3 depicts the QEs’ BREBias (vertical axis) for all 60 PCs that are sorted from left to right in increasing order of actual effort spent. The bars below 0 represent overestimated PCs (i.e. estimate > actual), while above 0 are underestimated PCs (i.e. estimate < actual).

The results clearly show a contrasting pattern with respect to over- and underestimation; smaller PCs tend to be overestimated, while larger PCs tend to be underestimated.
Chapter 6. Effort Estimation in Large Scale Software Development: An Industrial Case Study

Figure 6.3: BREbias of QEs of PCs arranged on actual effort

Figure 6.4: BREBias boxplots by size for QE
Grouping BREBias by PC size shows that an increase in size seems to be related to an increase in estimation error and underestimation bias (see Figure 6.4). We therefore applied a Kruskal-Wallis test [19] to see whether there are statistically significant differences between the QE’s BREbias of the four PC sizes. The results show that the BREbias of QEs correlates significantly with PC size ($p < 0.0001$ at $\alpha = 0.05$).

![Figure 6.5: BREbias of AEs of PCs arranged on actual effort](image)

**Analysis estimate (AE)**  AEs show a similar trend, though the division of over- and underestimated PCs is not that clear (see Figure 6.5). Some smaller PCs are underestimated, and a few larger PC are slightly overestimated. However, underestimation becomes more frequent as we move to large and very large PCs.

Figure 6.6 shows AEs’ BREBias grouped by PC size. The results show that underestimation bias increases with the increase in PC size. A Kruskal-Wallis test showed that there are statistically significant differences between the BREbias of different PC sizes ($p < 0.0022$ at $\alpha = 0.05$).

Table 6.8 lists the mean and median BREbias of the two estimates for the four PC sizes. The results show two trends: 1) For both QEs and AEs, an increase in size is related to an increase in underestimation bias. The increase in bias is larger for
Figure 6.6: BREBias boxplots by size for AE

QEs, indicating that PC size impacts QEs more strongly; 2) For almost all PC sizes, the magnitude of estimation error decreases as we move from QEs to AEs, i.e. the re-estimation at the analysis stage (AE) improves the accuracy of the effort estimates.

For both estimates (QE and AE), there are statistically significant differences in BREBias values across four PC sizes i.e., increase in size from small and medium to large and very large PCs significantly increases the magnitude of effort overruns.

Customer priority

Our dataset comprises 26 PCs with high priority (one small, 10 medium, seven large, eight very large) and 34 PCs with normal priority (nine small, 15 medium, seven large, three very large). To make the samples and the results comparable, we excluded the small PCs, since there was only one small PC with high priority. The mean and median BREBias of the remaining 50 PCs (25 PCs each with high and normal priority) are summarized in Table 6.9. These results show the following interesting patterns:
Table 6.8: Mean and Median BREBias for both estimates (QE and AE) grouped by PC size

<table>
<thead>
<tr>
<th>PC size</th>
<th>Mean (QE)</th>
<th>Mean (AE)</th>
<th>Median (QE)</th>
<th>Median (AE)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Small</td>
<td>-1.18</td>
<td>-0.40</td>
<td>-0.99</td>
<td>-0.23</td>
</tr>
<tr>
<td>Medium</td>
<td>-0.34</td>
<td>0.35</td>
<td>-0.21</td>
<td>0.34</td>
</tr>
<tr>
<td>Large</td>
<td>1.31</td>
<td>0.76</td>
<td>0.40</td>
<td>0.29</td>
</tr>
<tr>
<td>Very Large</td>
<td>2.10</td>
<td>0.97</td>
<td>1.30</td>
<td>0.56</td>
</tr>
</tbody>
</table>

- The estimation accuracy improves considerably as we move from quotations (QE) to analysis (AE) stage estimation for high priority PCs. However, for normal priority PCs it is other way around.

- At the quotation stage, the estimates (QE) of normal priority PCs are more accurate as compared to the high priority PCs.

- At the analysis stage however, the estimates (AE) of the high priority PCs are more accurate.

These results indicate that there is a relation between customer priority and estimation accuracy. However, a Mann-Whitney test [21] did not detect a statistically significant difference in the distributions of BREbias of QEs and AEs across the two priorities ($p < 0.46$ for QEs and $p < 0.23$ for AEs; $\alpha = 0.05$).

Table 6.9: Mean and median BREBias for both estimates (QE and AE) grouped by PC priority

<table>
<thead>
<tr>
<th>Customer priority</th>
<th>Mean (QE)</th>
<th>Mean (AE)</th>
<th>Median (QE)</th>
<th>Median (AE)</th>
</tr>
</thead>
<tbody>
<tr>
<td>High</td>
<td>0.68</td>
<td>0.31</td>
<td>0.32</td>
<td>0.20</td>
</tr>
<tr>
<td>Normal</td>
<td>0.41</td>
<td>0.84</td>
<td>0.10</td>
<td>0.54</td>
</tr>
</tbody>
</table>

**Team maturity**

The PCs in our sample are developed by teams at varying levels of maturity (see Section 6.4.3 for description of the maturity levels). For the PCs developed at each maturity level, we computed the mean and median BREbias for both estimates (see Table 6.10). The results show a decrease in underestimation bias with the increase in the team maturity level from A to D. It indicates that the teams at low maturity level are more likely to
incure larger magnitude of effort overruns. However, a Kruskal-Wallis [19] test did not detect a statistically significant difference in the distribution of QEs and AEs’ BREbias across four team maturity levels.

Table 6.10: Mean and median BREBias for both estimates (QE and AE) across all maturity levels

<table>
<thead>
<tr>
<th>Team maturity</th>
</tr>
</thead>
<tbody>
<tr>
<td>No. and sizes of PCs</td>
</tr>
<tr>
<td>A</td>
</tr>
<tr>
<td>B</td>
</tr>
<tr>
<td>C</td>
</tr>
<tr>
<td>D</td>
</tr>
</tbody>
</table>

S = Small, M = Medium, L = Large, VL = Very Large

Team maturity varies from level A (lowest level) to level D (highest level), see Section 6.4.3. The data points, with respect to size and number of PCs, has an uneven distribution across the four maturity levels (see Column 2 of Table 6.10).

Thus, accounting for all maturity levels when conducting the hypothesis testing would lead to meaningless results. On the other hand, the maturity levels are clearly grouped in two categories: teams that require mentoring to do their tasks (A and B); teams that are independent (C and D). Thus, to enable analyzing the impact of maturity on the accuracy of effort estimates, we categorized teams at levels A and B as less mature teams and teams at levels C and D as more mature teams. PCs of all sizes are developed by these two types of teams, as follows:

- Number of PCs by less mature teams: small (six), medium (14), large (seven), very large (four).
- Number of PCs by more mature teams: small (four), medium (14), large (five), very large (six).

Table 6.11: Mean and median BREBias for both estimates (QE and AE) for two groups of teams

<table>
<thead>
<tr>
<th>Team Maturity</th>
<th>No. of PCs</th>
<th>Mean (QE)</th>
<th>Mean (AE)</th>
<th>Median (QE)</th>
<th>Median (AE)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Less mature teams</td>
<td>31</td>
<td>0.68</td>
<td>0.60</td>
<td>0.03</td>
<td>0.33</td>
</tr>
<tr>
<td>More mature teams</td>
<td>29</td>
<td>-0.20</td>
<td>0.26</td>
<td>0.08</td>
<td>0.25</td>
</tr>
</tbody>
</table>

Table 6.11 shows that the mean BREbias is better for mature teams, but quite similar for the medians. As for the estimation bias, the results show a considerable reduction in the underestimation bias for mature teams. Together, it means that less mature
teams are more likely to incur larger effort overruns as compared to the more mature teams. However, a Mann-Whitney test did not show any statistically significant differences in the distributions of BREbias for QEs and AEs across two groups of teams.

**Multi-site development**

Six PCs in our dataset were developed by geographically distributed teams (referred here as multi-site development), while the remaining 54 were developed by co-located teams with remote support provided by software architects located in Sweden. The development of these six PCs is led by a team at one site. The teams in other sites contribute by developing relatively smaller and specific parts due to special competencies at these sites. Multi-site development could also have an impact on the accuracy of the effort estimates.

The six multi-site development PCs have mainly large sizes (very large: three, large: two, medium: one), whereas the PCs developed in a co-located setting have all types of sizes. To make a fair comparison, we excluded all small PCs, leaving behind 44 PCs of medium, large and very large size for the co-located case. Table 6.12 summarizes effort estimation accuracies for co-located and multi-site PCs and shows that both mean and median BREbias are higher for multi-site PCs.

These results indicate that multi-site development in the case leads to relatively larger effort overruns. However, a Mann-Whitney test did not find the differences in the distribution of the BREbias for both QEs and AEs (44 co-located PCs and six multi-site PCs) to be statistically significant across two types of PCs, i.e. co-located and multi-site. This may be due to the very small number of the PCs in the multi-site category.

Moreover, in order to see whether multi-site development phenomena has a statistically significant impact on the results of the statistical tests reported in 6.4.2 (Table 6.6), Section 6.4.4, 6.4.4 and 6.4.4, we removed the multi-site PCs from our sample, and re-performed these tests. All tests returned similar results with minor changes in the resulting p-values. The changes were so insignificant that they did not lead to any reversal in the corresponding null hypotheses testing results.

Table 6.12: Mean and median BREBias for both estimates (QE and AE) for co-located and multi-site PCs

<table>
<thead>
<tr>
<th>PC type</th>
<th>Mean (QE)</th>
<th>Mean (AE)</th>
<th>Median (QE)</th>
<th>Median (AE)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Co-located</td>
<td>0.41</td>
<td>0.41</td>
<td>0.09</td>
<td>0.34</td>
</tr>
<tr>
<td>Multi-site</td>
<td>1.51</td>
<td>1.79</td>
<td>1.28</td>
<td>1.18</td>
</tr>
</tbody>
</table>
6.4.5 Results from the interviews

Besides understanding the effort estimation processes used in the case, the interviews were also used to discuss the results with the participants. Information about the four interviewees are provided in Table 6.1 in Section 6.3. The interviewees were asked about the following: i) their background and experience in the company and the case, ii) their role in the effort estimation processes, iii) how various factors impact the accuracy of the effort estimates, and iv) the main reasons for effort overruns. The information will be used in the discussion section to further explain the results presented above.

6.5 Discussion

In this section we further discuss the results presented in Section 6.4 in the light of our discussion with interviewees.

6.5.1 Effort estimation processes (RQ1)

Our interviews show that it is mainly the scale of a PC project that makes it challenging to estimate and plan. The scale includes many factors beyond PC size, such as the number of sites involved, the number of stakeholders involved in the process, the size and complexity of the legacy code, etc. The coordination between different types of stakeholders involved in the estimation processes also introduces challenges (see Section 6.4.1 for stakeholders details). Research shows that coordination challenges exacerbate issues in multi-team projects [36]. Despite these challenges, a reasonable proportion of PCs (1/3, see Table 6.4) are accurately estimated within a 25% error margin at both quotation and analysis stages.

The involvement of the mature teams in the analysis stage estimation process is an important practice. The interviewed architect and project manager both suggested that it helps in arriving at better estimates. Previous studies [38, 39] found that the involvement of the development team members in the estimation process improves estimation accuracy. Whenever possible, the teams should provide their input in the estimation process.

6.5.2 Accuracy of estimates (RQ2)

Our analysis showed that, overall, underestimation is the dominant trend at both quotation and analysis stages. This is in line with the results in other estimation studies [34]. The interviewees attributed the inaccuracies in the effort estimates mainly to the following challenges that they encounter in this very large-scale distributed agile project:
Table 6.13: Summary of results

<table>
<thead>
<tr>
<th>Research Question</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>RQ1: Effort estimation process</td>
<td>Factors related to scale (e.g., PC size, number of sites/teams involved, size and complexity of legacy code) makes estimation and planning more challenging. The involvement of the development teams in the estimation process helps in arriving at better estimates.</td>
</tr>
<tr>
<td>RQ2: Accuracy of estimates</td>
<td>Overall, underestimation is the dominant trend. Re-estimation at the analysis stage improves estimation accuracy. However, it increases the overoptimism bias. Requirements related issues (such as lack of details and changes) are the main reason for effort overruns.</td>
</tr>
<tr>
<td>RQ3 and RQ4: Factors affecting the estimates</td>
<td>It is much more challenging to estimate large PC tasks. PCs of large size were found to have larger magnitude of effort overruns. Less mature teams are more likely to incur larger magnitude of effort overruns. Late changes in requirements for high priority PCs, after quotation approval stage, reduces the accuracy of the QEs. On the other side early start and dedicated capacity of teams improves the accuracy of AEs of high priority PCs. PCs developed in multi-site development setting have larger magnitude of effort overruns than PCs developed in co-located setting. It should be noted that only six PCs in our sample of 60 PCs were developed in multi-site setting.</td>
</tr>
</tbody>
</table>

- Requirements related issues such as lack of details and changes in the requirements.
- Lack of expertise of newly on-boarded teams results in delays.
- Dependencies (such as for code reviews) on specific human resources (e.g., prod-
uct architects) introduce delays.

- Project scale and distribution across multiple sites.
- Underestimating the technical complexity of some large PCs.

All interviewees stressed the importance of better requirements to be able to estimate more effectively. In a survey of 300 software professional, Grimstad et al. [40] found that the frequently changing, new and lack of well-defined requirements commonly contribute to estimation inaccuracy.

Our results show that the AEs are more accurate than the QEs. However, underestimation bias increases at the analysis stage. The interviewees suggest that the availability of more detailed information and knowledge of the teams and their input at the analysis stage might have increased the estimators’ optimism, resulting in relatively higher underestimation. Underestimation bias is relatively less evident at quotation stage, where in 30% of the PCs the effort is overestimated (see Table 6.4). At quotation stage, the estimators are more conservative in their estimates mainly due to a high level of uncertainty, lack of detail in the requirements, and that the PCs have not yet been assigned to a development team (i.e., the development team’s maturity level is unknown).

### 6.5.3 Factors affecting the effort estimates (RQ3 and RQ4)

We identified four factors that impact the accuracy of the effort estimates (Section 6.4.3), and analyzed how these factors affect the effort estimates (Section 6.4.4). We now discuss these factors in the light of the interview results.

#### PC size

We found statistically significant differences in the distribution of BREbias, for both types of effort estimates (QE and AE), across PCs of different sizes. For smaller PCs, there is a tendency to overestimate, while for large and very large PCs our results show a large underestimation bias. The interviewees attribute this to the inherent difficulty in estimating large PCs, whereby estimators underestimate the likely complexities and technical challenges. Moreover, the development of large PCs involve more unanticipated challenges and risks related to project management, coordination, design and architecture, resulting in potentially large delays. It is more challenging to estimate large projects due to their inherent complexity [42].

Moløkken-Østvold and Furulund [27] also identified a tendency of larger effort overruns for projects of large size. They used actual effort spent to classify projects into
big and small ones. We also used actual effort spent to classify PCs into four categories to see if the PC size has any impact on the effort overruns. Ropponen and Lytinen [41] found that project size has an impact on the scheduling and timing risks. They noted that larger projects, in terms of man months and actual cost, perform significantly worse than smaller projects.

**Team maturity**

Team maturity also impacted the effort estimates. Our results show that PCs developed by mature teams have more accurate effort estimates. In the present case, 11 teams were on-boarded relatively recently. They are the least mature teams in our dataset, and are incurring higher effort overruns than the more mature teams. Tanveer et al. [43] also found that the developer’s knowledge and experience affect the accuracy of the effort estimates.

Team related factors, such as team’s skill level and prior experience, are considered as important drivers of the development effort [13]. To mature, newly on-boarded teams require a lot of time and mentoring support from product architects in Sweden [28]. The mentoring support provided by the product architects is critical to achieving technical consistency, which otherwise is hard to maintain in such projects that scale up involving several agile teams at different levels of maturity [36].

The interviewees suggested that mature teams perform better than quotation estimates due to a better understanding of the product architecture and the high technical competence. At the analysis stage estimation, team maturity is not a significant factor according to our regression analysis (see Section 6.4.3). This might be because the team’s maturity is already accounted for in the AEs, as the estimators are aware of the team which is going to work on the PC being estimated.

**Customer priority**

Customer priority was also identified as a factor impacting the effort estimates’ accuracy. The results show a contrasting pattern, i.e., effort estimates of PCs with high priority are more accurate at analysis stage (AE) and less accurate at quotation stage (QE), while it is the other way around in case of PCs with normal priority. The interviews revealed that for high priority PCs there is relatively more emphasis to shorten lead times by ensuring dedicated capacities for these high priority PCs. Moreover, as the work is initiated relatively quickly on most of the high priority PCs, more is known when the estimation at the analysis stage is performed. More information implies less uncertainty, which could be a reason for the relatively better analysis stage estimates (AE). The interviewees attributed the relatively higher inaccuracy of QEs for high pri-
Multi-site development

Six PCs in our dataset were developed in multi-site settings. It is a too small number to perform a generalizable quantitative analysis. Nevertheless, in our limited sample the multi-site PCs have a stronger underestimation bias than co-located PCs, i.e., they incur larger effort overruns (as compared to the estimates). The interviewees attribute this to the inherent difficulties related to communication and coordination present in multi-site arrangements.

Global barriers related to time, distance and culture in multi-site development have been identified as important cost contributors in other studies as well [15]. Herbsleb and Mockus [9] found that multi-site development introduces significant delays during the task development. Dikert et al. [36] identified that managers in several organizations find it challenging to create and estimate user stories in large-scale distributed agile projects. This impact of multi-site development on effort estimation needs to be further investigated, preferably with a larger dataset.

6.6 Threats to validity

For our discussion, we use the classification of threats to validity by Runeson and Höst [10]. Furthermore, we also discuss conclusion validity [4].

Reliability is related to the repeatability of a study and how data is collected and analyzed [10].

To minimize this threat, we designed and followed an explicit, detailed case study protocol, following the guidelines by Runeson and Höst [10]. Furthermore, several researchers were involved in the design and execution of our investigation to minimize dependencies based on particular individuals. In addition, our observations and findings were verified with the company representatives to avoid false interpretations.

Internal validity is related to factors that researchers are unaware of or cannot control regarding their effect on the variables under investigation [10].

The effort estimates’ accuracy, how the estimates are obtained (estimation process) and actual effort in projects are constructs influenced by different factors, such as team maturity. It is therefore difficult to identify causal relationships between them, since there are many confounding factors. The factors accounted for in our investigation do not constitute an exhaustive list of factors that impact effort estimation. They were
identified through our discussions with the interviewees and the unit manager. Further, reliable data was available about these factors only. Nevertheless, the investigated factors are reported by related literature and also by the interviewees. Regarding the qualitative part of our research (semi-structured interviews), the main internal validity threats are investigator bias and interviewee bias. To mitigate these threats, three researchers were involved with the design of the interview guides (investigator triangulation). We mitigated interviewee bias by interviewing people with different roles (data triangulation). However, our interview sample consists of only one person in each of the roles relevant to the effort estimation process. It is possible that their views and explanations are not shared by other people involved in the case project. The interviewees are working in the case company for long time (see Table 6.1) and are involved in the estimation process, and therefore have a good understanding of the estimation process and the factors investigated in this study.

**Construct validity** reflects how well the measures used actually represent the constructs the study intends to measure [10]. The main threat of this category related to our investigation is the mono-method bias (only one method to measure a construct). Although just one measurement method was used to measure each construct, we cross checked different documents to increase the reliability of the collected data, with special focus on the collected actual effort. For example, we compared the data on progress reports and time reports to see whether the hours reported were in line with the number of people working in a given week in a particular task. Finally, we also conducted unstructured interviews with a project manager to further verify the consistency of the collected data.

**External validity** is concerned with the generalizability of the findings [10]. Since we employed the case study method, our findings are strongly bounded by the context of our study. In addition, the investigated case involved only one project in one company. We investigated the impact of four factors on effort estimation. However, considering the above mentioned limitations, we believe that it is important to investigate their impact in other cases. To mitigate this threat, we made an attempt to detail the context of our study as much as possible, complying with corporate confidentiality concerns. To mitigate this threat, we made an attempt to describe the context of our study in as much detail as possible. This makes it easier to relate the present case to similar cases.

**Conclusion validity** is concerned with the correctness of conclusions regarding relationships in the analyzed data [4]. The main threats of this category related to our investigation are the low reliability of measures due to noise and low statistical power. To mitigate the first threat, we conducted an unstructured interview with a project manager to verify the consistency of the collected data. Regarding statistical power, we investigated the largest number of tasks possible. However, it is important
Chapter 6. Effort Estimation in Large Scale Software Development: An Industrial Case Study

to emphasize that it is just a sample and does not cover all product customization tasks carried in the investigated case.

6.7 Conclusion

This study reports the results of a case study conducted in Ericsson AB with the aim to investigate the effort estimation in a large-scale distributed project. The case is related to a large telecommunication product, which is being evolved for over 20 years by geographically distributed agile teams. We used archival research to collect data about 60 Product Customization (PC) tasks involving 18 different teams in Sweden, India, Italy and USA.

RQ1: effort estimation process Effort estimation process is carried out at two stages in the case: quotation and analysis. At the quotation stage, the system manager along with the SAT prepares the QE to support the customer unit in deciding whether there is a business case in approving the PC for development. At the analysis stage, a more detailed analysis is performed of those PCs whose QEs have been approved by the customer unit. The assigned SAT member leads this analysis to propose a solution and effort estimate of the PC.

RQ2: accuracy of the effort estimates The results showed that underestimation is the more frequent trend at both quotation and analysis stages. The estimates at the analysis stage (AEs) were found to be more accurate than the QEs. However, they have significantly higher optimism bias as compared to QEs.

RQ3 and RQ4: factors impacting effort estimates Based on our analysis of 60 PCs from one project within one organization, we identified PC size, team maturity, customer priority and multi-site development as factors that impact the accuracy of the effort estimates.

The results indicate that PC size has a strong influence on the effort estimates’ accuracy and bias, i.e., increase in PC sizes were found to be strongly related to the rise in the underestimation bias. We also identified that the mature teams, due to their experience and technical competence, are less likely to incur large delays. Furthermore, PCs with high priority in terms of having dedicated capacity have better analysis stage estimates mainly due to the early start of work on them. Furthermore, the results also indicated that the multi-site PCs have a larger magnitude of effort overruns as compared to the co-located PCs.
The identified correlations between effort estimates and these factors (e.g. PC size, team maturity, customer priority and multi-site development) needs to be further investigated before drawing any conclusions.

These results have important implications for practice. Software practitioners, working in similar contexts, should consider the following aspects when estimating:

- As underestimation is the more dominant trend, it is important to consider factors (e.g., PC size, team maturity, etc.) that could potentially add delays in the project. The identification and consideration of these factors would be helpful in reducing over-optimism bias.

- It is important to involve all concerned stakeholders (e.g., system managers, project manager, architects, design lead) in the estimation process. Specifically, software architects have a pivotal role in the estimation process due to their understanding of the product’s architecture, which allows them to effectively perform an impact analysis of new requirements/tasks at the system level.

- Re-estimation at the analysis stage improves the effort estimates. At this stage estimators know more about the requirements and the assigned team(s). These refined estimates at the analysis stage support project managers in monitoring the progress on the task development.

- PC size was found to be strongly related to the effort estimates’ accuracy and bias. Large sized requirements/tasks are more likely to incur large effort overruns.

- Mature teams should be involved in the effort estimation process as they have architectural knowledge and expertise.

- Less mature teams need more time to complete the tasks, and hence are more likely to incur effort overruns.

- Multi-site development, wherein geographically distributed teams collaboratively work on a task, exacerbates the underestimation bias. A better task allocation strategy that minimizes the inter-site coordination and communication needs could help in reducing the overheads associated with the multi-site development.

As for research, the identified factors need to be further investigated in other contexts. The practitioners in the investigated case use expert judgment to estimate the software development effort. Expert judgment is the most widely used estimation technique [37]. We plan to investigate how the explicit consideration of relevant factors...
Chapter 6. Effort Estimation in Large Scale Software Development: An Industrial Case Study

with the help of a checklist supports expert judgment in arriving at better and informed effort estimates. In the next chapter we propose a process to develop and evolve estimation checklists to improve expert judgment based effort estimation process.
6.8 References


Chapter 6. Effort Estimation in Large Scale Software Development: An Industrial Case Study


Appendix 6A: BREbias QE regression model details

Table 6A.1: BREbias QE regression analysis: ANOVA

<table>
<thead>
<tr>
<th></th>
<th>Sum of Squares</th>
<th>df</th>
<th>Mean Square</th>
<th>F</th>
<th>Sig</th>
</tr>
</thead>
<tbody>
<tr>
<td>Regression</td>
<td>93.21</td>
<td>2</td>
<td>46.60</td>
<td>20.98</td>
<td>0.000</td>
</tr>
<tr>
<td>Residual</td>
<td>126.61</td>
<td>57</td>
<td>2.22</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>219.82</td>
<td>59</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 6A.2: BREbias QE regression coefficients’ details

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Constant</td>
<td>-0.73</td>
<td>0.41</td>
<td>-1.78</td>
<td>0.08</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Team Maturity</td>
<td>-0.56</td>
<td>0.20</td>
<td>-0.28</td>
<td>-2.78</td>
<td>0.007</td>
<td>0.996</td>
</tr>
<tr>
<td>PC Size</td>
<td>1.22</td>
<td>0.20</td>
<td>0.61</td>
<td>6.01</td>
<td>0.000</td>
<td>0.996</td>
</tr>
</tbody>
</table>

Table 6A.3: BREbias QE regression models’ summary

<table>
<thead>
<tr>
<th>Variables</th>
<th>R²</th>
<th>Adjusted R²</th>
<th>PC Size</th>
<th>PC Priority</th>
<th>Team Maturity</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.42</td>
<td>0.39</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>2</td>
<td>0.42</td>
<td>0.40</td>
<td>✓</td>
<td></td>
<td>✓</td>
</tr>
</tbody>
</table>
Figure 6A.1: Histogram of residuals

Figure 6A.2: Normal PP Plot of regression standardized residuals.
Figure 6A.3: Scatter plot standardized residuals and predicted values
Appendix 6B: BREbias AE regression model details

Table 6B.1: BREbias AE regression: ANOVA.

<table>
<thead>
<tr>
<th></th>
<th>Sum of Squares</th>
<th>df</th>
<th>Mean Square</th>
<th>F</th>
<th>Sig</th>
</tr>
</thead>
<tbody>
<tr>
<td>Regression</td>
<td>16.36</td>
<td>2</td>
<td>8.18</td>
<td>6.79</td>
<td>0.002</td>
</tr>
<tr>
<td>Residual</td>
<td>68.69</td>
<td>57</td>
<td>1.21</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>85.05</td>
<td>59</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 6B.2: BREbias AE regression coefficients’ details.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Constant</td>
<td>-0.07</td>
<td>0.25</td>
<td>-0.27</td>
<td>-0.79</td>
<td>0.44</td>
<td></td>
</tr>
<tr>
<td>PC Priority</td>
<td>-0.69</td>
<td>0.31</td>
<td>-0.29</td>
<td>-2.251</td>
<td>0.028</td>
<td>0.862</td>
</tr>
<tr>
<td>PC Size</td>
<td>0.57</td>
<td>0.16</td>
<td>0.45</td>
<td>3.54</td>
<td>0.001</td>
<td>0.862</td>
</tr>
</tbody>
</table>

Table 6B.3: BREbias AE regression models’ summary.

<table>
<thead>
<tr>
<th>Variables</th>
<th>R²</th>
<th>Adjusted R²</th>
<th>PC Size</th>
<th>PC Priority</th>
<th>Team Maturity</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>0.19</td>
<td>0.15</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>2</td>
<td>0.19</td>
<td>0.16</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>
Chapter 6. Effort Estimation in Large Scale Software Development: An Industrial Case Study

Figure 6B.1: Histogram of residuals

Figure 6B.2: Normal PP Plot of regression standardized residuals
Figure 6B.3: Scatter plot standardized residuals and predicted values
Chapter 7

Developing and Using Checklists to Improve Expert Effort Estimation: A Multi-Case Study

7.1 Introduction

Effort estimation techniques based on expert judgment are the most commonly used techniques for estimating software development effort [3, 10, 11]. In expert estimation, the involved experts rely on their knowledge and previous experience to subjectively analyze a number of factors to estimate the development effort. Experts perform this subjective assessment using their “intuition”, which is described as a “non-explicit and non-recoverable reasoning process” by Jørgensen [11]. In some cases experts may rely on their intuition only, while in others experts may use historical data and/or checklists, subject to their availability, to make more informed estimates.

Expert estimation involves little documentation and a light-weight process. Compared to algorithmic estimation models, such as COCOMO I and II [12, 13], the focus is on the experts performing the estimation. These attributes of expert estimation synchronize well with agile processes, where individuals and their interactions are valued more than processes and tools.

The lack of formalism in expert estimation may have some negative consequences;
in the absence of any explicit structure such as checklist, expert estimation can be-
come an ad-hoc activity. Furthermore, experts may miss important tasks (e.g., testing
effort [3] or non-functional requirements [3, 20]) and as a result underestimate devel-
opment effort. In early project stages uncertainty is also high, e.g., due to incomplete
and changing requirements, which may lead to inconsistent expert estimates [21]. Ex-
perts are normally aware of the factors that should be considered. However, they might
not always recall and consider these factors consistently during estimation. In the agile
context, where team-based expert estimation is the norm [2, 3] and there is lack of doc-
umentation [46], junior or new team members might not be aware of all activities and
factors that should be accounted for during effort estimation. Furthermore, relatively
less emphasis on documentation in the agile context is also likely to result in the loss
of the useful estimation experience generated during the effort estimation process.

Checklists have the potential to support estimation in addressing the issues de-
scribed above [11]. Checklists document the factors and tasks that should be consid-
ered during effort estimation, which ensures that they are taken into account. Checklists
are also useful in supporting new team members [11], who would be less aware of the
activities and factors that should be considered during estimation. Checklists can also
prove helpful in documenting and reusing estimation data and experience to improve
estimation accuracy over time [22].

There is evidence that the use of checklists helps to improve estimation accuracy
(e.g., [16, 17]). However, we have not seen any empirical study in software engineering
that describes and demonstrates how to develop and evolve such checklists for agile
teams. The present study addresses these gaps by making the following contributions:

• Propose a process to develop and evolve a customized checklist to improve ex-
  pert estimation.

• Apply the process with agile teams in three different companies to develop and
  evolve checklists for their specific contexts.

• Evaluate the usefulness of the proposed checklists using both qualitative and
  quantitative data.

The remainder of the chapter is organized as follows: Section 7.2 details the back-
ground of this work and presents related work on the use of checklists in software
effort estimation; the research methodology is described in Section 7.3; Sections 7.4
and 7.5 present the results and analysis; Section 7.6 provides further discussions on
these results, while Section 7.8 concludes the chapter.
7.2 Background and related work

In this section first we discuss the use of checklists in other disciplines such as aviation and health care. Next, we briefly discuss use of checklists in software engineering in general, also in effort estimation in particular. Lastly, we describe our previous works that provided important inputs and motivation for the work presented in this study.

7.2.1 Checklists in other disciplines

Checklists are widely used to avoid human error, improve safety and performance in variety of disciplines, such as health care, aviation and product manufacturing [26]. In complex environments, experts are likely to overlook routine matters due to fallible memory and distraction under pressing situations [27]. Furthermore, experts can knowingly skip certain steps with the assumption that certain tasks are not always required. Checklists have the potential to help avoiding such lapses [27]. They are widely used in fields, such as aviation and health care, where safety and accuracy are critical to reduce human error [26].

Checklists became a standard operating procedure in the aviation industry after the crash of the Boeing\(^1\) Model 299 during the trial demonstration in 1935 [25]. From mid 1990s, electronic checklists were introduced in aircrafts to prevent crew errors associated with paper checklists [32, 33].

Pronovost proposed and used a simple five step checklist to avoid human mistakes, which were responsible for causing deadly infections to patients in surgical Intensive Care Units (ICUs) [28]. The intervention resulted in a large reduction (up to 66\%) in infection rates [29]. Later, Dr. Pronovost’s idea was adopted by all other states of the USA [30], and was also replicated at other places (cf. [31]).

7.2.2 Checklists in software engineering

Checklist have long been used as an aid during inspections of different software artifacts such as requirements and design documents, and code [38]. Checklists support inspectors or reviewers in systematically evaluating or reading the software artifact being inspected (cf. [34–37]). Checklists are also used in empirical software engineering research (cf. [1, 39]) to assess the quality of the empirical studies. These inspection and review checklists can be classified as “evaluative checklists” [40], as the objective is to evaluate or review an artifact or document. Evaluation checklists improve the

\(^1\)http://www.boeing.com/
consistency of the evaluations by providing reviewers a set of practices or guidelines or criteria to be used during evaluation [40, 41].

Besides evaluation, checklists are also employed as mnemonic instruments that serve as a “reminder system” to help experts in consistently applying procedures and processes [40]. Mnemonic checklists facilitate experts in remembering the relevant information and steps, which they can otherwise forget due to lapse in attention or memory [27]. In software effort or cost estimation, such checklists have been used to support estimators during the estimation activity [11]. Jørgensen [11] noted that the checklists are beneficial as they support experts in not forgetting relevant activities and factors during estimation, increase their confidence in estimates and encourage them to use more accurate strategies during estimation. Similar results were found by Passing and Shepperd [17], who performed an experiment with student subjects to investigate how the use of checklists and group discussions support expert estimation. They noted that the checklists improved the consistency and transparency of size and effort estimates. Further, they also found that the use of checklist increased the subjects’ confidence in their estimates. The checklist helped student estimators in recalling the relevant factors that should be considered during estimation.

Furulund and Moløkken-Østvold [16] found that projects where checklists were used during estimation have relatively accurate estimates as compared to the projects where checklists were not used.

Jørgensen [15] proposed a software cost management process framework and a preliminary checklist that follows the structure of the framework. The framework consists of four phases and several activities. The four phases are: preparation, estimation, application and learning phases. The framework and accompanying checklist have a broad scope, as they go beyond the typical estimation activity. Our goal is to propose and demonstrate a process to develop and evolve customized checklists specifically for the estimation activity to support practitioners in arriving at better estimates in an agile context.

7.2.3 Process support for software estimation checklists

There is a lack of process support for developing and implementing software estimation checklists. In the study by Furulund and Moløkken-Østvold [16], the focus is on demonstrating the usefulness of checklists and experience data in increasing the estimation accuracy. How the checklist was developed, evolved and used during effort estimation was not discussed. Likewise, Passing and Shepperd [17] investigated the use of checklist and group discussions in supporting expert estimation. In this study as well, the focus was not on process support for developing and implementing the checklist. The checklist contents are listed and consist of the typical activities (e.g.,...
manage requirements, class design, prototype development, defect fixing, unit testing, documentation etc.) in a development project.

As mentioned previously, preliminary checklist proposed by Jørgensen [15] is structured on a project management framework consisting of four phases and twelve activities. The checklist is based on a number of sources, including estimation checklists from the six Norwegian software companies, and the literature on cost management, forecasting and estimation. The scope of the checklist is much wider than the typical estimation activity, and also includes phases, such as preparation phase, beyond the typical estimation activity. In the context of the agile teams, wherein estimation is performed repeatedly with each release and iteration, it may not be feasible to use a long checklist covering activities beyond the typical estimation activity. Furthermore, checklist factors should be empirically grounded in the context in which they are intended to be used. Jørgensen [15] suggested that the proposed preliminary checklist should be customized to include only the relevant issues.

Hales et al. [40] recommended that the contents of the checklist should be identified systematically through a review of the peer-reviewed literature, and should also take into account the contextual factors of the local hospital. Our aim is also to develop estimation checklists, and accompanying process support, that are based both on published literature and also on the systematically elicited input from the involved agile teams. Hales et al. [40] also suggested that the checklist should not be onerous, and should be pilot tested and validated first in a controlled environment. We also aim to develop a customized and light-weight checklist. Furthermore, before transferring the checklist to the involved teams, we aim to present, validate and revise it first in a controlled setting, and then later evolve it further based on feedback from the actual use.

7.2.4 Agile estimation taxonomy

We organized the state of the art and practice on effort estimation in agile context as a facet based taxonomy [14], referred here as AET (Agile Estimation Taxonomy). At the top level, AET has four dimensions: context, estimation technique, effort predictors and effort estimate. Software effort estimation is carried out in a certain context (e.g., company/team size, domain, agile practices followed etc.) in which the estimator(s) use some estimation technique(s) (e.g., expert judgment, case base reasoning etc.) and effort predictors (e.g., size, teams expertise etc.) to arrive at an effort estimate (e.g., point estimate in hour or three point estimate etc.). These dimensions in turn have several facets, which were identified through a systematic review [2] and survey [3] on effort estimation in agile context. We used AET to characterize the effort estimation process of the three cases included in this study in Section 7.3 (see Table 7.4).
We used AET to characterize effort estimation processes of three companies. These companies found AET useful in documenting their effort estimation sessions [14]. The AET, however, was not used during the effort estimation process. To do so, we decided to develop a checklist instrument to support and structure the otherwise largely ad-hoc expert judgment based effort estimation process, which is the most frequently practiced estimation technique in software industry [9, 11].

The factors identified in AET served as important input in this study. In particular, AET’s effort predictor dimension encompasses factors (see Table 7.1) that estimators consider during the effort estimation process. Further input to this study came from an industrial case study on effort estimation in large scale development [24]. We investigated and identified factors (Table 7.1) that impact the accuracy of the effort estimates. Based on the data and participants’ opinion, the study recommended that it is important to consider the identified factors during effort estimation, as they can potentially lead to effort overruns in the project. Table 7.1 lists factors identified in our previous works, which acted as a starting point in this study.

<table>
<thead>
<tr>
<th>Factor</th>
<th>AET</th>
<th>Case Study</th>
</tr>
</thead>
<tbody>
<tr>
<td>Product size</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Team’s skill level</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Multi-site development issues</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Team’s prior experience with similar tasks</td>
<td>✓</td>
<td>-</td>
</tr>
<tr>
<td>Non-functional requirements</td>
<td>✓</td>
<td>-</td>
</tr>
<tr>
<td>Customer communication</td>
<td>✓</td>
<td>-</td>
</tr>
<tr>
<td>Customer priority</td>
<td>-</td>
<td>✓</td>
</tr>
</tbody>
</table>

### 7.3 Research methodology

We selected case study methodology to investigate effort estimation, which is the phenomenon under study. Based on Runeson et al.’s classification [1], it can be classified as “improving” case study as our goal was to improve the expert effort estimation process.

#### 7.3.1 Research context

The study is performed in three different case companies. The details about these companies, their products and processes are described in the following.
Case company 1

Infoway\(^2\) is a medium-sized Brazilian company developing software applications for the health sector. It has over 40 employees in different development roles. We worked with two teams working on two different products at Infoway: Product A and B.

Product A is a management information system for health insurance providers. The product supports insurance providers in marketing and sales management, order management, customer records and payment management. It also provides an interface to hospitals through which they can register patients and interact with their corresponding insurance providers for authorization of the treatments. The development team consists of three developers, one team lead and a project manager. The product has been evolved for over 7 years, and is being used by many insurance companies in Brazil. The product is being developed in Java\(^3\) and consists of approximately 0.41 million lines of code.

Product B is a business intelligence application to support health insurance providers to optimize their insurance plans. It helps the insurance providers to setup key indicators to monitor the pattern of physicians recommending different examinations to their customers. The product notes and reports the unusual behavior/pattern as an alert to the managers of the insurance provider. The managers can set up new indicators, and also define/change the thresholds for alerts associated with different indicators. The development team consists of five developers and a team lead (also performs the project management tasks). The product has been evolved for over 5 years, and is being used by many insurance companies in Brazil. The product is being developed in JAVA and frameworks associated with Java (e.g., Spring framework\(^4\)). The code size is approximately 0.49 million lines of code.

Case company 2

Diyatech\(^5\) is a medium-sized software vendor based in Pakistan. It has over 50 employees based in Pakistan. They develop performance enhancement applications for distributed environments. We studied one product of Diyatech: Product C. Product C is designed to enhance performance and support extreme transaction processing for applications running in distributed environments. It is being developed by a 6-8 member development team. It is a mature product having customers in the USA and EU in finance, health and airline industries. It is being evolved for over 10 years. Product C

\(^2\)http://infoway-br.com/
\(^3\)https://go.java/index.html
\(^4\)https://projects.spring.io/spring-framework
\(^5\)http://www.diyatech.com
is developed in .Net platform\(^6\). Some code is also written in Java, C++ and PHP. The code size is approximately 1.5 lines of codes.

**Case company 3**

Case company 3, referred here as TSoft due to confidentiality, is an offshore development center of a transport company based in Norway. It is a new setup to develop an application for the parent company in Norway. One of the senior executives in Norway acts as the product owner, and interacts with the development center. The parent company in Norway is the main customer. However, the long term aim is to develop and market the product to outside world as well. The development team consists of a project manager, four developers, one QA engineer and a UI designer. The first release of the product was round the corner when this study was performed. The product is being developed in .Net platform. For mobile front ends, Android\(^7\) and iOS\(^8\) are used.

**Unit of analysis**

The unit of analysis is the effort estimation process used in the case organizations. Expert judgment based effort estimation is practiced in all three case organizations.

At infoway both teams use planning poker to estimate selected tasks during a monthly sprint plan meeting. This sprint plan meeting normally takes place during the first week of each month. Teams use the last week of each month for retrospectives, and for converting the backlog items to tasks that can be estimated and assigned to individual team members in the next meeting. The central two weeks are used by team members to develop and test the tasks assigned to them. The customer department interacts with the clients to collect the new requirements and feedback. The project manager is responsible for interacting with the clients, mainly through customer department, and if required directly as well.

In parallel with the planned tasks for a sprint, Infoway teams have to work on the immediate unplanned customer requirements as well. In order to account for this continuous flow of unplanned requirements, managers set the focus factor for a team. A focus factor of 60% for a team means that 60% of the time in the sprint is allocated for the planned tasks, while the remaining 40% is reserved for the immediate customer requests. The focus factor for Infoway teams varies from 50% to 80%.

\(^6\)https://www.microsoft.com/net/
\(^7\)https://www.android.com
\(^8\)https://developer.apple.com/ios/
At Diyatech, the development unit consists of one development manager, architects, development and QA (Quality Assurance) teams and their respective leads. The marketing team identifies the features for the next release, and translate them to specific requirements. The development unit also gives their input in this feature discovery phase. The development manager is responsible for estimating the requirements for the next release. He consults architects and leads of the development and QA teams during the estimation process. One release consists of three to four sprints, wherein one sprint is three to four weeks long. In the last sprint of every release emphasis is on validation of the implemented functionality.

At TSoft the project manager maintains the product backlog with the help of the product owner in Norway. The team uses two weeks time boxed sprints to manage the development. At the end of each sprint, the developers select relevant items from the backlog for the next sprint, and individually prepare an effort estimate. The selected items and their estimates are discussed in a sprint plan meeting at the start of each sprint.

**Commonalities in three cases**

The following three aspects are common in the three case organizations:

- Some variant of expert judgment based approach is used to estimate effort.

- The organizations follow agile practices such as sprint and release planning, time boxed sprints, stand ups and retrospectives, unit testing, backlog etc.

- Effort is mostly underestimated, i.e. actual effort exceeds the corresponding estimate. This was highlighted upfront at the start of our collaboration in all three cases. Later, every interviewee in all three cases confirmed it. This is in line with the evidence in the effort estimation literature, which shows that underestimation occurs more frequently as compared to other scenarios (e.g., [3, 9]).

**7.3.2 Research questions**

The overarching research goal is to improve the expert judgment based effort estimation process in an agile context. We focused on expert judgment for being the frequently used estimation approach in software industry in general [11], and for the agile context in particular [2, 3]. The context is agile software development as agile practices are widely used in software industry these days.
After understanding the research context described above, the following specific research questions were framed to guide this study:

- **RQ1:** How to develop and evolve a customized checklist to support expert judgment based effort estimation?
- **RQ2:** How useful is the checklist in improving expert judgment based effort estimation?

The aim in RQ1 is to propose and validate a process for developing a customized effort estimation checklist to improve expert judgment based effort estimation process. In RQ2, we plan to see how useful are checklists during effort estimation.

### 7.3.3 Data collection methods

We used multiple data collection methods in all cases to ensure method and source triangulation [1]. The employed methods are listed below.

1. **Interviews:** We used semi-structured interviews at multiple stages in all cases: First, with a senior manager to understand the organization, the case context and an overview of the estimation process, and then later with the team members for more specific elicitation, feedback and validation purposes.

2. **Workshop:** We used the workshop to present and explain the estimation checklist to the team members, and also to take their initial feedback.

3. **Questionnaire:** We also used questionnaire to take team members’ feedback on estimation checklist. The questionnaire contained both open and closed ended questions.

4. **Metrics:** We obtained the effort related data from the archival records of the companies.

5. **Checklist data:** We operationalized the estimation checklist as a spreadsheet so that the usage data is stored.

The data collection methods listed above were applied differently in three case organizations due to varying contexts, availability and confidentiality. The details are described in Table 7.2.

Table 7.3 presents the demographic information of the study participants for all three cases.
Table 7.2: Description of data collection mechanism in the case organizations

<table>
<thead>
<tr>
<th>Organization</th>
<th>Data collection</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Infoway</strong></td>
<td>In the beginning two interviews over Skype were conducted with the company director to understand the company profile and portfolio, and planning and estimation processes. The study plan was also agreed upon in these discussions over Skype. Later, the first author visited the company in Brazil to conduct the face to face interviews and workshops. For product A, we interviewed two developers, team lead and the project manager. For Product B, we interviewed the team lead and three developers. The workshop was conducted jointly with representatives from both teams. The feedback on the checklist was subsequently obtained through an iterative process using questionnaire (filled by the teams jointly) and interviews with the team leads. Team leads selected the developers for interviews. The effort metrics (actual and estimated effort in hours) were collected by the respective team leads from the project management tool used in the organization.</td>
</tr>
<tr>
<td><strong>Diyatech</strong></td>
<td>To start with two interviews over Skype with the development manager were conducted focusing on the company profile and portfolio, and planning and estimation processes. Later, the first author visited Pakistan to conduct interviews and workshops with the development manager. The interviews with the architect and the developers were conducted in the written form, i.e., open ended questions were handed over to them through development manager (see interview questions in Appendix 7B: Data Collection Instruments). The development manager selected the developers for written interviews. The feedback on the checklist was subsequently obtained through an iterative process using questionnaire and interviews with the development manager. We did not interact directly with the development team in this case, as they were not involved in the estimation process. The effort metrics (actual and estimated effort in hours) were collected by the development manager in a spreadsheet from the project management tool used in the organization.</td>
</tr>
<tr>
<td><strong>TSoft</strong></td>
<td>One Skype interview with the project manager was performed to understand the case context, and their planning and estimation processes. Later, the first author visited the company to conduct interviews and workshops with the development team. We interviewed project manager, three developers and one tester. The workshop was run jointly with these team members. The feedback on the checklist was subsequently obtained through an iterative process using questionnaire and interviews with all interviewees. Being a new offshore setup, TSoft did not maintain the effort metrics of their previously completed sprints. They, however, started doing that during this study.</td>
</tr>
</tbody>
</table>

7.3.4 Data analysis

We analyzed the qualitative data gathered through interviews and questionnaire by applying coding [45]. We used the existing knowledge on effort estimation (e.g., known reasons for effort overruns) to manually assign interpretive and pattern codes [45] to the interview text. The data obtained in the interviews was used to describe the effort estimation process and the factors that should be considered during effort estimation process. The estimation process, and the factors organized as a checklist were validated with the teams in workshops to ensure the correctness. The qualitative data in the questionnaire was validated by conducting follow up interviews in all cases. The quantitative data collected through questionnaire about the relevance of each factor in checklist was analyzed by using simple frequency statistics.

We used BRE (Balanced Relative Error) to calculate the accuracy of the effort esti-
Chapter 7. Developing and Using Checklists to Improve Expert Effort Estimation: A Multi-Case Study

Table 7.3: Demographic information of the study participants

<table>
<thead>
<tr>
<th>No.</th>
<th>Role</th>
<th>Exp. in current company (Years)</th>
<th>Total exp. in industry (Years)</th>
<th>No. and type of interviews</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Project manager</td>
<td>1.5</td>
<td>5</td>
<td>one face to face only (1 hr)</td>
</tr>
<tr>
<td>2</td>
<td>Team lead</td>
<td>3.5</td>
<td>3.5</td>
<td>several (both face to face and Skype)</td>
</tr>
<tr>
<td>3</td>
<td>Developer 1</td>
<td>2</td>
<td>2</td>
<td>one face to face only (45 mins)</td>
</tr>
<tr>
<td>4</td>
<td>Developer 2</td>
<td>4</td>
<td>5</td>
<td>one face to face only (30 mins)</td>
</tr>
</tbody>
</table>

**Infoway - Product B**

<table>
<thead>
<tr>
<th>No.</th>
<th>Role</th>
<th>Exp. in current company (Years)</th>
<th>Total exp. in industry (Years)</th>
<th>No. and type of interviews</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Team lead</td>
<td>1</td>
<td>4</td>
<td>several (both face to face and Skype)</td>
</tr>
<tr>
<td>2</td>
<td>Developer 1</td>
<td>1</td>
<td>4</td>
<td>one face to face only (45 mins)</td>
</tr>
<tr>
<td>3</td>
<td>Developer 2</td>
<td>1</td>
<td>5</td>
<td>one face to face only (25 mins)</td>
</tr>
<tr>
<td>4</td>
<td>Developer 3</td>
<td>0.5</td>
<td>3.5</td>
<td>one face to face only (25 mins)</td>
</tr>
</tbody>
</table>

**Diyatech**

<table>
<thead>
<tr>
<th>No.</th>
<th>Role</th>
<th>Exp. in current company (Years)</th>
<th>Total exp. in industry (Years)</th>
<th>No. and type of interviews</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Dev. manager</td>
<td>13</td>
<td>13</td>
<td>several (both face to face and phone)</td>
</tr>
<tr>
<td>2</td>
<td>Architect</td>
<td>10</td>
<td>10</td>
<td>written open ended</td>
</tr>
<tr>
<td>3</td>
<td>Developer 1</td>
<td>1</td>
<td>4</td>
<td>written open ended</td>
</tr>
<tr>
<td>4</td>
<td>Developer 2</td>
<td>3.5</td>
<td>3.5</td>
<td>written open ended</td>
</tr>
<tr>
<td>5</td>
<td>Developer 3</td>
<td>5</td>
<td>5</td>
<td>written open ended</td>
</tr>
<tr>
<td>6</td>
<td>Developer 4</td>
<td>3</td>
<td>6</td>
<td>written open ended</td>
</tr>
</tbody>
</table>

**TSoft**

<table>
<thead>
<tr>
<th>No.</th>
<th>Role</th>
<th>Exp. in current company (Years)</th>
<th>Total exp. in industry (Years)</th>
<th>No. and type of interviews</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Project manager</td>
<td>1</td>
<td>3</td>
<td>several (both face to face and Skype)</td>
</tr>
<tr>
<td>2</td>
<td>Developer 1</td>
<td>0.5</td>
<td>2.5</td>
<td>one face to face (1 hr) and one Skype (20 mins)</td>
</tr>
<tr>
<td>3</td>
<td>Developer 2</td>
<td>1</td>
<td>3.5</td>
<td>one face to face (45 mins) and one Skype (15 mins)</td>
</tr>
<tr>
<td>4</td>
<td>Developer 3</td>
<td>0.5</td>
<td>2</td>
<td>one face to face (30 mins) and one Skype (15 mins)</td>
</tr>
<tr>
<td>5</td>
<td>Quality engineer</td>
<td>0.5</td>
<td>0.5</td>
<td>one face to face only (30 mins)</td>
</tr>
</tbody>
</table>

mates. Historically, the Magnitude of Relative Error (MRE) has been frequently used for estimation accuracy. However, it has recently been criticized for its biased treatment of over and under estimation [4–6]. BRE balances the over and underestimation, and have therefore been used in many recent studies on effort estimation [7, 8]. BRE (Equation 7.1) calculates the magnitude of the estimation error, while BREbias (Equation 7.2), in addition to the magnitude of error, also shows the direction (over or under) of estimation error.

\[
BRE = \frac{|\text{actual effort} - \text{estimated effort}|}{\min (\text{actual effort}, \text{estimated effort})} \tag{7.1}
\]

\[
BRE_{\text{bias}} = \frac{(\text{actual effort} - \text{estimated effort})}{\min (\text{actual effort}, \text{estimated effort})} \tag{7.2}
\]

We used BREbias in this study, as we are interested in both magnitude and direction of the estimation error.

230
7.3.5 Characterizing the three cases with the Agile Estimation Taxonomy (AET) [14]

In Table 7.4 we applied the AET (briefly described earlier in Section 7.2.4) to characterize the effort estimation process of the three case companies included in this study. The characterization includes the details about the four taxonomy dimensions, i.e., context, effort predictors, estimation technique and effort estimate. In case of Diyatech estimation is performed at the release planning level, while in other two cases it is performed at the sprint planning level.

7.4 Results and Analysis - RQ1

This question aims to describe the process that we used to develop and evolve estimation checklists in the three case organizations. The process consists of five activities (see Figure 7.1): understand estimation context, develop and present checklist, validate checklist statically, validate checklist dynamically, transfer and follow up. We describe these activities, how they were applied in the three case companies, their inputs and outputs in the following sections. The concepts of static and dynamic validation are adopted from Gorshek et al.’s [23] technology transfer model.

7.4.1 Understand estimation context

The purpose of this activity is two-fold: 1) to understand the current effort estimation process and the context in which it is carried out, 2) to elicit the factors that should be included in the checklist.

Once the checklist is developed, it needs to be implemented and integrated with the estimation process of the case company. To do that effectively, it is important to understanding the current estimation process and the context (e.g., at which stage, release or iteration, estimation is performed) in which it is carried out.

Checklist factors can be identified in different ways, such as:
Table 7.4: Using AET to characterize estimation process of the three cases

<table>
<thead>
<tr>
<th>Dimension</th>
<th>Facet</th>
<th>Infoway</th>
<th>Diyatech</th>
<th>TSoft</th>
</tr>
</thead>
<tbody>
<tr>
<td>Context</td>
<td>Planning level</td>
<td>Sprint</td>
<td>Release</td>
<td>Sprint</td>
</tr>
<tr>
<td></td>
<td>Estimated activity</td>
<td>Implementation and Testing (I&amp;T)</td>
<td>All</td>
<td>I&amp;T</td>
</tr>
<tr>
<td>Agile Method(s)</td>
<td>Customized Scrum</td>
<td>Customized Scrum</td>
<td>Scrum practices</td>
<td></td>
</tr>
<tr>
<td>Product domain</td>
<td>eHealth</td>
<td></td>
<td></td>
<td>Transportation</td>
</tr>
<tr>
<td>Project setting</td>
<td>Co-located</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Estimation unit</td>
<td>Task</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Number of tasks/reqs.</td>
<td>Refer to Table 7.11</td>
<td>Refer to Table 7.12</td>
<td>21 tasks</td>
<td></td>
</tr>
<tr>
<td>Team size</td>
<td>5-6</td>
<td></td>
<td></td>
<td>6-8</td>
</tr>
</tbody>
</table>

Effort predictors: Refer to their final checklists in Appendix Appendix 7A: Checklist Versions to see the effort predictors used during effort estimation.

<table>
<thead>
<tr>
<th>Estimation technique</th>
<th>Technique name</th>
<th>Type</th>
<th>Effort estimate</th>
<th>Effort data (actual, estimate and accuracy level achieved)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Group</td>
<td>Planning poker</td>
<td>Refer to Table 7.11 and Table 7.12 to see the details</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Individual</td>
<td>Expert judgment</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Group</td>
<td>Expert judgment</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Group</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Hrs</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Hrs</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Hrs</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Hrs</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Point</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Point</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Point</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>BREbias</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>BREbias</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>BREbias</td>
<td></td>
</tr>
</tbody>
</table>
1. Understand estimation context
- Identify CL (CheckList) factors
- Understand current estimation process

2. Develop and present initial CL
- Develop initial CL using identified factors
- Present CL to the involved development team(s)
- Discuss how CL will be included in the estimation process
- Obtain team's initial feedback
- Revise CL based on team feedback, and present again in case of significant changes
- Discuss next step, and request the involved team(s) to prepare for the static validation

3. Validate CL statically
- The involved team(s) use CL on a sample of old requirements/tasks
- Obtain team's feedback after trial
- Revise CL based on feedback and send revised CL to the team
- Discuss next step, and request teams to prepare for the dynamic validation

4. Validate CL dynamically
- The involved team(s) use CL in real context with their estimation process
- Obtain team's feedback after real use
- Discuss next step, and request teams to keep recording the effort data
- Request effort data after the end of the planned release/iteration for accuracy analysis

5. Transfer and followup
- Revise CL based on team's feedback
- Transfer the revised CL to the teams for the future use
- Plan and perform followup, if possible

Figure 7.1: Checklist development process.

- Top down: In top down approach, effort and cost drivers from the effort estimation literature (e.g., [12, 13]) can be used as factors to include in the checklist. Traditional cost estimation models use these drivers in arriving at the cost and
effort estimates. However, in expert judgment based approaches experts use the contextual information to determine the effort estimates. The general purpose cost drivers are not relevant for all contexts.

- **Bottom down**: In bottom up approach, the factors are identified directly from the software experts involved in the estimation process. Understandably, factors identified in this way are likely to be different for each case. As the factors are elicited from the experts involved in the estimation process, they would be relevant for the given context.

- **Hybrid**: In this approach a combination of the above two approaches can be used. The factors from the literature can be used as starting point for discussion. Based on the discussion with the experts, additional factors can be identified.

Hales et al. [40], for the development of medical checklists, suggested to base the checklist contents both on the published literature and the local hospital policies and procedures. We also used a hybrid approach to elicit relevant factors for estimation checklist. The factors identified in our previous works (see Section 7.2 and Table 7.1) were used to guide the discussion in the semi-structured interviews with the study participants to identify additional factors.

**Infoway**

Infoway teams use planning poker, an expert judgment based group estimation technique, to estimate effort. We elicited details of the estimation process initially during Skype interviews with company director, and later during face to face interviews with team leads and members (see Section 7.3.1 for details).

In order to elicit the factors that should be considered during estimation, we asked study participants to discuss and describe the reasons for the most frequent estimation scenario (i.e., effort overruns) in their teams. The identified factors from the two Infoway teams are listed in Table 7.5. Four factors (No. 3, 4, 5 and 7 in Table 7.5) overlap with the previous works. The remaining factors, such as No. 1 and 2, have been described as the leading reasons for effort overruns in other studies as well (e.g., [3]).

**Diyatech**

At Diyatech the development manager uses expert judgment to estimate requirements for the next release. He takes input from the architect and the team leads. The developers are not involved in the estimation process (see Section 7.3.1 for more details). The development manager and the architect have been working in the same
Table 7.5: checklist factors identified through interviews

<table>
<thead>
<tr>
<th>No.</th>
<th>Factor</th>
<th>InfoWay (n = 4)</th>
<th>Product B (n = 4)</th>
<th>Diyatech (n = 6)</th>
<th>TSoft (n = 5)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Requirements issues (changes, missing details etc.)</td>
<td>4</td>
<td>4</td>
<td>6</td>
<td>5</td>
</tr>
<tr>
<td>2</td>
<td>New type of tasks involve more uncertainty</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>Overestimating the available skills/knowledge in team</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>Missing tasks while decomposing requirements</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>Ignoring non-functional requirements</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>6</td>
<td>Architecturally complex tasks introduce more delays</td>
<td>3</td>
<td>3</td>
<td>6</td>
<td>4</td>
</tr>
<tr>
<td>7</td>
<td>Large sized tasks are more prone to effort overruns</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>8</td>
<td>Team members working on other projects/tasks in parallel</td>
<td>2</td>
<td>1</td>
<td>-</td>
<td>2</td>
</tr>
<tr>
<td>9</td>
<td>New team members</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>-</td>
</tr>
<tr>
<td>10</td>
<td>Low team productivity during recent sprints</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>11</td>
<td>Involvement of external interfaces</td>
<td>-</td>
<td>-</td>
<td>2</td>
<td>5</td>
</tr>
</tbody>
</table>

company for over ten years. The factors that contribute to the inaccurate effort estimation are described in Table 7.5. These were used as input to the first draft of the estimation checklist for Diyatech.

**TSoft**

At Tsoft, team members prepare estimates for the selected backlog items for the next release. The estimates are discussed and revised, if required, in a sprint meeting. In the opinion of the development team, the factors that contribute to the effort overruns are listed in Table 7.5. These factors overlap with the factors elicited from the other two cases. The one additional factor is related to the external interfaces in case of TSoft. The system being developed had to interface with several third party systems, and the tasks that involve these interfaces were more complex. We used these factors as input to develop the first version of the estimation checklist for TSoft.

### 7.4.2 Develop and present initial checklist

The purpose of this activity is to use the output of the previous elicitation step to formulate and present a customized checklist for the involved teams. The checklist would include factors that should be considered by the team when estimating requirements. It is an iterative step wherein the proposed checklist should be revised based on teams’ feedback. In case there are no more change suggestions, the involved teams are requested to prepare for the next step, i.e., static validation, wherein they will have to apply the checklist to estimate a sample of old tasks or requirements.

While presenting the checklist to the involved teams, it should also be discussed how exactly the checklist will be used by the teams during effort estimation.
Infoway

The first version of the Infoway estimation checklist is described in Table 7.6. We conducted a joint workshop with both teams to present the checklist and take participants’ initial feedback. We presented each checklist item by focusing on three questions: What (is the factor), Why (is it there in the checklist), and How (is it to be answered during estimation). During the discussion on the “how” part, we discussed suitable scale values that teams would be required to select for each factor while filling in the checklist to estimate a task or requirement. It was agreed that the checklist would be operationalized as a spreadsheet. The team leads would use its factors to guide estimation discussions, and would also answer each factor with a consensus value.

The participants agreed on the following changes (see rows in gray background in Table 7.6) during the workshop:

• Remove Factor 10, and add it as last scale value in place of “Lacking” with Factor 8. We therefore removed Factor 10 and redefined the scale of Factor 8 as: Very good, good, average, below average and need to talk to customer before moving on.

• Add a new factor to identify the type of the task, i.e., implementation or testing.

We incorporated these recommended changes in the checklist, and handed it over to the teams for static validation (see next Section).

In the light of the discussions during workshop, we also added detailed notes in the spreadsheet with each checklist factor explaining the factor and corresponding scale values and when a value should be selected.

An example note with the checklist factor about domain knowledge:

It is the information/knowledge about the domain of the product that you are developing. For example in case of ehealth products, information on the health system, insurance policies, rules etc. represent domain knowledge

Select "Very good" if you think team has all the domain knowledge required to implement this task
Select "Good" if you think team has most of the domain knowledge required to implement/test this task
Select "Average" if you think team has to study/research for some time to obtain the required domain knowledge
Select "Below Average" If you think team has to study/research more to obtain the required domain knowledge
Select "Lacking" if you think team needs to study/research a lot or need help from seniors or other sources to be able to get the required domain knowledge.
Figure 7.2: Checklist versions.
Chapter 7. Developing and Using Checklists to Improve Expert Effort Estimation: A Multi-Case Study

Table 7.6: Infoway checklist V1

<table>
<thead>
<tr>
<th>No.</th>
<th>Checklist factor</th>
<th>Scale</th>
<th>Linked no. in Table 7.5</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>How similar is the task to the previously developed tasks?</td>
<td>To a great extent, somewhat, very little,</td>
<td>No. 2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>not at all</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Rank team’s technical competence/skills required to implement and test this task</td>
<td>Very good, good, average, below average,</td>
<td>No. 3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>lacking</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Rank team’s domain knowledge required to implement this task</td>
<td>Very good, good, average, below average,</td>
<td>No. 3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>lacking</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Does the task require an architectural change?</td>
<td>Yes, not sure, no</td>
<td>No. 6</td>
</tr>
<tr>
<td>5</td>
<td>Does the task involve communication between multiple (sub) systems?</td>
<td>Yes, not sure, no</td>
<td>No. 6</td>
</tr>
<tr>
<td>6</td>
<td>Does the task involve accessing and/or modifying several different elements/tables in the persistence/DB layer?</td>
<td>Yes, not sure, no</td>
<td>No. 6</td>
</tr>
<tr>
<td>7</td>
<td>Are there stringent non-functional requirements (performance, security etc.) associated with this task?</td>
<td>Yes, not sure, no</td>
<td>No. 5</td>
</tr>
<tr>
<td>8</td>
<td>How clear is your understanding of the task?</td>
<td>Very good, good, average, below average,</td>
<td>No. 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>lacking</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>Is the task size suitable to fit in the planned sprint?</td>
<td>Yes, not sure, no</td>
<td>No. 7</td>
</tr>
<tr>
<td>10</td>
<td>Is there a need to communicate with customer to further clarify this task?</td>
<td>Yes, not sure, no</td>
<td>No. 1</td>
</tr>
<tr>
<td>11</td>
<td>Will the team or some member(s) be working on other products/projects/tasks in parallel?</td>
<td>Yes, not sure, no</td>
<td>No. 8</td>
</tr>
<tr>
<td>12</td>
<td>Does the team has new member(s)?</td>
<td>Yes, no</td>
<td>No. 9</td>
</tr>
<tr>
<td>13</td>
<td>Rank team’s recent productivity</td>
<td>Very good, good, average, below average,</td>
<td>No. 10</td>
</tr>
<tr>
<td></td>
<td></td>
<td>lacking</td>
<td></td>
</tr>
</tbody>
</table>

Based on the above characterizations, most likely effort estimate for this task is: ———– hours

Diyatech

The first version of the checklist is described in Table 7.7. It was shared with the development manager for initial feedback. Later, it was also discussed in one workshop session with the development manager wherein the first author presented each checklist item and its corresponding scale values. Besides describing each checklist factor and the reason behind its inclusion in the checklist, we also explained how can it be answered. Using the knowledge and experience from the Infoway case, we also described and presented the explanations for how to answer each checklist factor by selecting the appropriate scale values. Checklist was implemented as a spreadsheet in this case as well. It was discussed that the development manager would first answer all checklist factors with suitable values before proposing the most likely effort estimate for each requirement.

The following changes were suggested during this workshop:

- Remove Factor 10 as it is not relevant to the type of system being developed in
Table 7.7: Diyatech checklist V1

<table>
<thead>
<tr>
<th>No.</th>
<th>Checklist factor</th>
<th>Scale</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>The requirement is classified as:</td>
<td>x, y, z (company specific categories)</td>
</tr>
<tr>
<td>2</td>
<td>How similar is the requirement to the previously developed requirements?</td>
<td>To a great extent, somewhat, very little, not at all</td>
</tr>
<tr>
<td>3</td>
<td>Rank team’s technical competence required to implement this requirement</td>
<td>Very good, good, average, below average, lacking</td>
</tr>
<tr>
<td>4</td>
<td>Rank team’s technical competence required to test this requirement</td>
<td>Very good, good, average, below average, lacking</td>
</tr>
<tr>
<td>5</td>
<td>Rank team’s domain knowledge required to implement this requirement</td>
<td>Very good, good, average, below average, lacking</td>
</tr>
<tr>
<td>6</td>
<td>Rank team’s recent productivity</td>
<td>Very good, good, average, below average, lacking</td>
</tr>
<tr>
<td>7</td>
<td>Does the team has new member(s)?</td>
<td>Yes, no</td>
</tr>
<tr>
<td>8</td>
<td>Will this requirement’s implementation involve an architectural change?</td>
<td>Yes, not sure, no</td>
</tr>
<tr>
<td>9</td>
<td>Will this requirement’s implementation involve communication between multiple (sub) systems?</td>
<td>Yes, not sure, no</td>
</tr>
<tr>
<td>10</td>
<td>Does the task involve accessing and/or modifying several different elements/tables in the persistence/DB layer?</td>
<td>Yes, not sure, no</td>
</tr>
<tr>
<td>11</td>
<td>The requirement is classified as Functional Requirement (FR) or Non-Functional Requirement (NFR) or both?</td>
<td>Only an NFR, FR with major NFR constraints, FR with minor NFR constraints, only FR</td>
</tr>
<tr>
<td>12</td>
<td>How clear is your understanding of the requirement?</td>
<td>Very good, good, average, below average, need to talk to customer before moving on</td>
</tr>
<tr>
<td>13</td>
<td>Will the team or some member(s) be working on other products/projects/tasks in parallel?</td>
<td>Yes, not sure, no</td>
</tr>
<tr>
<td>14</td>
<td>Is the requirement size suitable to fit in the planned release?</td>
<td>Yes, not sure, no</td>
</tr>
</tbody>
</table>

Based on the above characterizations, specify your most likely effort estimate for the following activities:

- Requirements analysis ——— hrs
- Documentation ——— hrs
- Design ——— hrs
- Coding ——— hrs
- Testing ——— hrs
- Non-development work ——— hrs
- Total ——— hrs

the case under study.

- Remove Factor 13 as the team members do not work on other projects in their context.

We incorporated these changes in the checklist and handed over the revised version (Table 7A.3 in Appendix 7A: Checklist Versions) to the development manager for static validation.

TSoft
Chapter 7. Developing and Using Checklists to Improve Expert Effort Estimation: A Multi-Case Study

The first version of TSoft checklist is described in Table 7.8. It was presented to the development team in a workshop held in the company. We used the same approach as discussed above in the Infoway case to present checklist to the TSoft team. TSoft team members individually prepare initial estimates of their tasks before discussing them in joint meeting. It was agreed that the team members would first use the checklist while preparing their individual estimates, and later to present and discuss their estimates in the joint meeting.

The following changes were proposed in the workshop:

• Revise Factor 1 on task classification to include more options in the available scale points. Previously, we had just frontend, backend and business logic as task types. The team members suggested to include a combination of these types as additional option.

• Remove Factor 13 about team recent productivity. The team members opined that the recent productivity could be bad due to issues outside their control, such as ambiguity in requirements, and therefore should not be a factor in determining the estimates.

We incorporated these recommendations in the checklist, and handed over the revised checklist (Table 7A.5 in Appendix 7A: Checklist Versions) to them for static validation.

7.4.3 Validate checklist statically

In static validation the involved team(s) is asked to use the checklist to estimate a sample of tasks or requirements from previous sprints. The idea is to allow the teams to use the checklist in a close to real situation to develop a good understanding of how to use the checklist to estimate tasks. After the validation, team’s feedback is elicited focusing on suggestions for further improvement. In case of major changes, the revised checklist should be tried again by the involved teams. At the end of this step, the teams should be requested to prepare for using the revised checklist in real context.

Infoway

At the end of the workshop described above in Section 7.4.2, both teams were requested to select a sample of at least three tasks from previous sprints, and estimate them with the checklist. It was decided that the two team leads would perform this task, and would act as the contact persons for this study. They would themselves interact with the teams, if and when required.
Table 7.8: Tsoft checklist V1

<table>
<thead>
<tr>
<th>No.</th>
<th>Checklist factor</th>
<th>Scale</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>What is the type of the task?</td>
<td>UI, Business logic, DB</td>
</tr>
<tr>
<td>2</td>
<td>How similar is the task to the previously developed tasks?</td>
<td>To a great extent, somewhat, very little, not at all</td>
</tr>
<tr>
<td>3</td>
<td>Rank team’s technical competence/skills required to implement and test this task</td>
<td>Very good, good, average, below average, lacking</td>
</tr>
<tr>
<td>4</td>
<td>Rank team’s domain knowledge required to implement this task</td>
<td>Very good, good, average, below average, lacking</td>
</tr>
<tr>
<td>5</td>
<td>Will the task implementation likely to involve interfacing with external services/systems?</td>
<td>Yes, not sure, no</td>
</tr>
<tr>
<td>6</td>
<td>Does the task require an architectural change?</td>
<td>Yes, not sure, no</td>
</tr>
<tr>
<td>7</td>
<td>Does the task involve communication between multiple (sub) systems?</td>
<td>Yes, not sure, no</td>
</tr>
<tr>
<td>8</td>
<td>Does the task involve accessing and/or modifying several different elements/tables in the persistence/DB layer?</td>
<td>Yes, not sure, no</td>
</tr>
<tr>
<td>9</td>
<td>Are there stringent non-functional requirements (performance, security etc.) associated with this task?</td>
<td>Yes, not sure, no</td>
</tr>
<tr>
<td>10</td>
<td>How clear is your understanding of the task?</td>
<td>Very good, good, average, below average, lacking</td>
</tr>
<tr>
<td>11</td>
<td>Is the task size suitable to fit in the planned sprint?</td>
<td>Yes, not sure, no</td>
</tr>
<tr>
<td>12</td>
<td>Will the team or some member(s) be working on other products/projects/tasks in parallel?</td>
<td>Yes, not sure, no</td>
</tr>
<tr>
<td>13</td>
<td>Rank team’s recent productivity</td>
<td>Very good, good, average, below average, lacking</td>
</tr>
</tbody>
</table>

Based on the above characterizations, most likely effort estimate for this task is: ———- hours

We held a discussion with both team leads where they were requested to explain how they used the checklist, and provide their feedback. The team leads expressed positive opinions on the checklist, and both shared their desire to use the checklist dynamically during the next sprint plan meeting. Furthermore, changes in the following two factors were suggested and agreed:

- The first change is about Factor 7, which deals with non-functional requirements. Team leads suggested that it is possible that a task is entirely about a non-functional requirement or functional requirement or both. Further, this factor should also be moved to the top of the list to identify the task type upfront.

- The second change is about the newly added factor during the workshop, which is about type of the task. The team leads explained that sometimes a task is assigned to a team member to do some research on it. To incorporate that, we added a new scale value “Research” besides implementation and testing in this factor.

The revised checklist is included in Appendix 7A: Checklist Versions, Table 7A.1,
with revised factors marked with the gray background. The team leads expressed their teams’ readiness in using the revised checklist during the next sprint plan meeting.

**Diyatech**

After using the checklist to estimate a sample of requirements from a previous release, one more factor was suggested to be included in the checklist. The factor is about priority of the requirements with respect to the importance of the customers to the product and company. The requirements from the old and large customers are treated as high priority items. The development manager opined that this should be reflected in the checklist. We added this factor in the revised checklist (Table 7A.3 in Appendix 7A: Checklist Versions). The development manager expressed his interest to continue using the checklist in actual planning.

**TSoft**

All involved members at TSoft used the checklist to estimate a sample of three previous tasks. After this exercise, we held a face to face session with them. The only change requested by two members of the team was to re-arrange the checklist factors by moving up the factor about task understandability. We re-organized the checklist according to the sequence suggested by them. The team members expressed confidence and interest in using the checklist during the next iteration planning instance.

### 7.4.4 Validate checklist dynamically

The purpose of dynamic validation is to start using the estimation checklist in real context, i.e., during estimation and planning meetings to estimate tasks or requirements. After teams have practically used the checklist with their effort estimation process for some time, one can analyze its usefulness and obtain teams’ feedback on their interest using and evolving it in future. It is an iterative step wherein teams can continue to evolve the checklist in light of their experiences to add or remove or modify checklist factors. Checklist should not be perceived as a final or static document. It is important to request the involved teams to keep record of the effort data, so that after the release or iteration is complete, we are able to analyze the estimation accuracy.

**Infoway**

The revised checklist, described in Appendix 7A: Checklist Versions (Table 7A.1), was handed over to the team leads as two separate spreadsheets. It was discussed and agreed that the team leads would ensure that each checklist factor is characterized
before estimating a task during estimation meetings. The first author stayed with the teams in Brazil until this stage.

After both teams had used the checklist with two sprints, we contacted team leads to take their feedback. The checklist’ perceived benefits and its impact on accuracy are discussed in Section 7.5. Both teams expressed keen interest in continuing to use the checklist in future. The following changes were suggested by the two teams.

- The teams suggested to add another factor to deal with the issue of legacy code. Since both teams are working on relatively old products, a considerable amount of legacy code has been developed. The tasks that involve changing or understanding legacy code are more effort intensive. We therefore added a new factor in the checklist about legacy code.

- The teams opined in cases where there will be relatively less time available for estimation meetings, the use of checklist adds some time overhead. To deal with such scenario it was discussed to classify the checklist factors into two parts: mandatory and optional. Mandatory part consists of those factors that will be characterized in all situations, while optional part includes factors that will be filled subject to the availability of the time. Furthermore, the team leads opted to do this classifications by themselves after consultations with their respective teams. The two leads finalized a common checklist consisting of the two parts, mandatory and optional.

The checklist was further revised (see Table 7A.2 in Appendix 7A: Checklist Versions) in the light of the changes mentioned above, and handed over to the Infoway teams for future use.

**Diyatech**

The revised checklist was used in estimating requirements of one release. The development manager and the architect suggested the following two changes in the checklist:

- Some requirements involve designing and implementing complex distributed algorithms. These requirements are found to be more challenging. They suggested to add a new factor to cover such cases.

- Another factor, related to the interruptions caused by meetings and other company events, was highlighted. These interruptions affect the momentum of the team members due to the switching of the context from the work.
Chapter 7. Developing and Using Checklists to Improve Expert Effort Estimation: A Multi-Case Study

We added the two suggested factors in the checklist, and handed over the revised version of the checklist (Table 7A.4 in Appendix 7A: Checklist Versions) to the development manager. The encouraging sign for us was their interest in continuing to use the checklist in future as well.

**TSoft**

TSoft team used the estimation checklist in two sprints to estimate tasks. The team did not suggest any change in the checklist. They expressed their interest in using the checklist with future sprints as well.

### 7.4.5 Transfer and follow up

After the estimation checklist has been dynamically validated, the next step is to revise and transfer the finalized checklist to the teams. Furthermore, it is important to submit a study report to the company management describing the whole exercise and its outputs in the form of final checklist and its version history. The report should also explain the benefits of using estimation checklists based on the analysis of the data collected during the whole process. It is up to the company management to see if they are interested in officially adding the checklist instrument as part of their estimation process.

A further follow up step can be performed after some time in future. The aim is to determine if the checklist is still in use or being evolved, and how useful it has been in improving the effort estimation process.

**Infoway**

Given that the teams have expressed their interest in continuing the use of estimation checklist after dynamic validation step, the finalized checklist is transferred to the teams for their use. Furthermore, study report was submitted to the company management that includes the finalized checklist, version history, its benefits based on the analysis of the qualitative and quantitative data (see Section 7.5) collected at different stages of this study.

We did a follow up after a gap of six months. The team leads informed us that the checklist has become a part of their estimation and planning process. It is, however, being used in a different form. Both teams have printed the checklist on a single page that each team member uses during the estimation. Further, the checklist had also been translated to Portuguese language to facilitate some team members. During the follow
up interaction, the Product A team lead shared his experience that checklist helped him to get back into the estimation process when he came back from vacations.

“T’ve spent sometime not participating in estimation meetings and then I went back for it. At that moment, the checklist really helped me, because I had spent some time far from the estimation meeting and then I had to do it again, it was much easier to adapt myself back again.”

Product A team lead

Diyatech

After the dynamic validation step, we handed over the revised checklist to the development manager, who shared with us that the checklist would remain part of their estimation process. We highlighted the importance of maintaining the estimation data over the next few releases for a potential follow up investigation. The dynamic validation step has been completed recently. We plan to conduct a follow up in future when data from more releases will be available for analysis. Furthermore, checklist users would have gained more experience in the application of checklist, which would allow us to further improve the checklist.

TSoft

The revised checklist was handed over to the project manager and the development team. We also provided the project manager with a report of the study findings using estimation data after the use of checklist and developers’ subject opinion. The report also included suggestions relating to the planning process, code review process, and the need to report estimates and actual effort for monitoring performance and using the data to improve planning and estimation in future. The TSoft team was not using any project management tool to record estimates and actual effort spent, we requested them to enter the estimates, and subsequently the actual effort in the spreadsheet, which included the checklist as well.

After a gap of six months, we contacted the project manager of the TSoft team through email for a follow up. In a half hour Skype interview subsequently, we were informed that the development team is not using the checklist to estimate the tasks during sprint planning. According to the project manager the developers perceived it as an overhead to: 1) use the checklist to estimate, 2) and more importantly, to log the actual hours spent on each task. Keeping track of the actual effort spent was a requirement from our side, as we wanted to determine the accuracy of the effort estimates.

Nonetheless, the project manager shared with us that he himself is using the checklist factors in prioritizing the product backlog. The project manager uses the checklist factors to perform an initial impact analysis of the backlog items to determine their priority. Some checklist factors such as architectural change, involvement of NFRs and
external services, support in establishing an initial understanding of the complexity of the involved tasks. Other factors, such as team’s skills and domain knowledge, helps in ranking the team’s readiness and capacity to take on the particular backlog item.

We reformulated the checklist factors by referring them to backlog items (see Table 7A.6 in Appendix 7A: Checklist Versions), rather than sprint tasks, so that they are aligned with the revised usage of the checklist.

7.5 Results and Analysis - RQ2

This section presents and analyzes the results for RQ2, which is about usefulness of the checklist in improving the expert judgment based effort estimation in the three case organizations. In all three cases the study participants showed keen interest to continue the use of estimation checklist in real situations (dynamic validation) after the workshops and the static validation. The checklist was then used by the study participants in real situations. Based on this real use, we collected more data to see how useful has been the checklist in improving the effort estimation.

In all three cases we first used a questionnaire, which included questions about relevance of each checklist factor to the effort estimation process, and also the feedback and reflections on the checklist as a whole (see feedback instrument in Appendix 7B: Data Collection Instruments). Further, we also collected effort data to see if the use of the checklist has made any impact on the accuracy of the effort estimates. Finally, we performed Skype interviews to further understand the results and also the observed benefits and drawbacks, if any, of the checklists. We summarized the benefits noted by the case participants in Table 7.9.

The inclusion of the recommended changes in the different versions of the checklists has been discussed earlier in Section 7.4.

7.5.1 Infoway

After the revised estimation checklist (Table 7A.2 in Appendix 7A: Checklist Versions) was used by both Infoway teams in three sprints, we collected data to see how useful was the checklist.

Checklist usefulness based on subjective feedback:

A common concern of both teams on the initial two versions of the checklist was time overhead due to the use of checklist during estimation. This was addressed by revising the checklist to include mandatory and optional parts.
The teams, after using the revised checklist for two sprints, found that the checklist provided many benefits. They noted that the checklist, besides improving the estimation process and their confidence in estimates, has also helped them in improving the understanding of the tasks being estimated. The team leads associated it with the fact that checklist induces discussions on various topics (e.g., architecture, legacy code) covered in various factors.

The team leads first rated each checklist factor after consulting their respective teams. Both leads discussed their ratings with each other and shared with us their common rating. They rated factors 1 to 8 in Table 7.10 as extremely relevant, while factors 9 to 12 were rated as relevant to the effort estimation. They were undecided about factor 13 and 14. The remaining factors in Table 7.10 (15 to 17), marked as “N/A” are not rated as they were not part of the checklist for the Infoway case.

Checklist usefulness based on effort data:

Table 7.11 presents the mean and median BREbias values for sprints before and after the checklist was introduced in the two Infoway teams. The effort estimates vary from 1 hour to 8 hours normally. The company provided us with the estimation metrics for five sprints before the introduction of the estimation checklist. Apart from few exceptions, the data shows that the underestimation was the dominant trend for both products. It is much more significant in case of Product B. We have the data for only three sprints when the checklist was subsequently used to estimate tasks. We can see reduction in the underestimation bias in both products. In case of product A the data shows a shift towards the overestimation. The accuracy of the estimates in the three sprints where checklist was used, is better than the previous five sprints.
Chapter 7. Developing and Using Checklists to Improve Expert Effort Estimation: A Multi-Case Study

Table 7.9: Benefits of using estimation checklist

<table>
<thead>
<tr>
<th>No.</th>
<th>Benefit</th>
<th>Infoway</th>
<th>Diyatech</th>
<th>TSoft</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>The checklist improved our confidence in estimates</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>2</td>
<td>The checklist reduced the chances of missing important factors during estimation</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>3</td>
<td>The checklist improved our estimation process</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>4</td>
<td>The checklist reduced the chances of missing tasks</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>5</td>
<td>The checklist improved our understanding of the tasks being estimated</td>
<td>✓</td>
<td>-</td>
<td>✓</td>
</tr>
<tr>
<td>6</td>
<td>Expert estimation became relatively objective due to the checklist</td>
<td>-</td>
<td>✓</td>
<td>-</td>
</tr>
<tr>
<td>7</td>
<td>The checklist helped in understanding the implementation complexity of tasks</td>
<td>-</td>
<td>-</td>
<td>✓</td>
</tr>
<tr>
<td>8</td>
<td>The checklist is a good support for new team members</td>
<td>✓</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 7.10: Teams’ feedback on the relevance of the checklist factors to the effort estimation

<table>
<thead>
<tr>
<th>No.</th>
<th>Checklist factor</th>
<th>Infoway</th>
<th>Diyatech</th>
<th>TSoft</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>What is the type of the task*?</td>
<td>++</td>
<td>++</td>
<td>++</td>
</tr>
<tr>
<td>2</td>
<td>The task is implementing a Functional Requirement (FR) or Non-Functional Requirement (NFR) or both?</td>
<td>++</td>
<td>++</td>
<td>+</td>
</tr>
<tr>
<td>3</td>
<td>Rank team’s technical competence/skills required to implement and test this task</td>
<td>++</td>
<td>++</td>
<td>++</td>
</tr>
<tr>
<td>4</td>
<td>How similar is the task to the previously developed tasks?</td>
<td>++</td>
<td>+</td>
<td>++</td>
</tr>
<tr>
<td>5</td>
<td>Rank team’s domain knowledge required to implement this task</td>
<td>++</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>6</td>
<td>How clear is your understanding of the task?</td>
<td>++</td>
<td>+</td>
<td>++</td>
</tr>
<tr>
<td>7</td>
<td>The implementation of the task requires understanding and/or changing legacy code</td>
<td>++</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>8</td>
<td>Does the task involve communication between multiple (sub) systems?</td>
<td>++</td>
<td>++</td>
<td>++</td>
</tr>
<tr>
<td>9</td>
<td>Does the task require an architectural change?</td>
<td>+</td>
<td>++</td>
<td>+</td>
</tr>
<tr>
<td>10</td>
<td>Does the team has new member(s)?</td>
<td>+</td>
<td>++</td>
<td>N/A</td>
</tr>
<tr>
<td>11</td>
<td>Will the team or some member(s) be working on other products/projects/tasks in parallel?</td>
<td>+</td>
<td>N/A</td>
<td>++</td>
</tr>
<tr>
<td>12</td>
<td>Does the task involve accessing and/or modifying several different elements/tables in the persistence/DB layer?</td>
<td>+</td>
<td>N/A</td>
<td>+</td>
</tr>
<tr>
<td>13</td>
<td>Is the task size suitable to fit in the planned sprint?</td>
<td>-</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>14</td>
<td>Rank team’s recent productivity</td>
<td>-</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>15</td>
<td>The task involves designing and/or revising and/or implementing distributed algorithm(s)</td>
<td>N/A</td>
<td>++</td>
<td>N/A</td>
</tr>
<tr>
<td>16</td>
<td>The may be interruptions during the planned release/sprint affecting team member’s momentum due to context switch</td>
<td>N/A</td>
<td>++</td>
<td>N/A</td>
</tr>
<tr>
<td>17</td>
<td>What is priority of the requirement being implement</td>
<td>N/A</td>
<td>++</td>
<td>N/A</td>
</tr>
<tr>
<td>18</td>
<td>The task involves interfacing with external services/systems?</td>
<td>N/A</td>
<td>N/A</td>
<td>++</td>
</tr>
</tbody>
</table>

* In case of Diyatech the word requirement was used instead of task, as they estimate requirements not tasks.
++ = strongly agree, + = agree, - = undecided, - = disagree, - - = strong disagree, N/A = Not Applicable

7.5.2 Diyatech

The revised checklist was used in one release to estimate requirements. The release consisted of 8 requirements, wherein the estimates varied from 200 hours to 800 hours.
Table 7.11: Accuracy of estimates before and after checklist introduction at Infoway

<table>
<thead>
<tr>
<th>Sprint</th>
<th>No. of tasks</th>
<th>Mean BREbias</th>
<th>Median BREbias</th>
<th>No. of tasks</th>
<th>Mean BREbias</th>
<th>Median BREbias</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sprint 1</td>
<td>25</td>
<td>0.80</td>
<td>0.81</td>
<td>28</td>
<td>1.13</td>
<td>-0.06</td>
</tr>
<tr>
<td>Sprint 2</td>
<td>21</td>
<td>0.62</td>
<td>0.57</td>
<td>25</td>
<td>-0.36</td>
<td>-0.09</td>
</tr>
<tr>
<td>Sprint 3</td>
<td>33</td>
<td>0.25</td>
<td>0.09</td>
<td>32</td>
<td>2.11</td>
<td>1.06</td>
</tr>
<tr>
<td>Sprint 4</td>
<td>26</td>
<td>-0.04</td>
<td>0.11</td>
<td>63</td>
<td>1.30</td>
<td>0.15</td>
</tr>
<tr>
<td>Sprint 5</td>
<td>48</td>
<td>0.25</td>
<td>0.14</td>
<td>34</td>
<td>3.21</td>
<td>1.20</td>
</tr>
</tbody>
</table>

Sprints where checklist was used in estimation

<table>
<thead>
<tr>
<th>Sprint</th>
<th>No. of tasks</th>
<th>Mean BREbias</th>
<th>Median BREbias</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sprint 1</td>
<td>40</td>
<td>-0.11</td>
<td>0.05</td>
</tr>
<tr>
<td>Sprint 2</td>
<td>26</td>
<td>-0.26</td>
<td>-0.13</td>
</tr>
<tr>
<td>Sprint 3</td>
<td>28</td>
<td>-0.02</td>
<td>-0.14</td>
</tr>
</tbody>
</table>

Checklist usefulness based on subjective feedback:

After the estimation was performed, we collected feedback from the development manager about the usefulness of the estimation checklist. Development manager shared several benefits of using checklist during estimation (see Table 7.9). In addition to the benefits highlighted by Infoway and TSoft cases, development manager opined that the expert estimation became more objective due to the use of checklist, without introducing much overhead.

"The best part is that by following this checklist, the estimation becomes more factual and objective. One might skip important factors if checklist is not followed and hence becomes underestimation."

"Although, some of the checklist items were always considered but there were a few important items that we mostly overlooked...it has helped us in structuring estimation process and has moved us a step closer to be more accurate in time estimation."

Development Manager

The development manager rated the relevance of the 13 factors in the final checklist to the estimation. He rated 9 factors as extremely relevant, while the remaining 4 as relevant. These ratings (see Table 7.10) were obtained on the revised checklist, and therefore understandably no factors were rated as irrelevant.
Chapter 7. Developing and Using Checklists to Improve Expert Effort Estimation: A Multi-Case Study

Checklist usefulness based on effort data:

We managed to get the effort estimation data for one previous release when the checklist was not introduced. The mean and median BREbias values for the two releases (see Table 7.12) show that the checklist estimates are more accurate and have considerably low underestimation bias. The data clearly is not enough to draw any conclusions. However, it is aligned with the opinion of the manager and team that the underestimation had been the dominant trend, and checklist has helped in improving the estimates.

Table 7.12: Accuracy of estimates before and after checklist introduction at Diyatech

<table>
<thead>
<tr>
<th>Release</th>
<th>No. of requirements</th>
<th>Mean BREbias</th>
<th>Median BREbias</th>
</tr>
</thead>
<tbody>
<tr>
<td>One release where checklist was not used</td>
<td>8</td>
<td>0.20</td>
<td>0.27</td>
</tr>
<tr>
<td>One release where checklist was used</td>
<td>7</td>
<td>0.02</td>
<td>0.04</td>
</tr>
</tbody>
</table>

7.5.3 TSoft

The TSoft team used the checklist for two sprints to estimate 21 tasks in total. The task estimates vary from 4 hours (half day work) to 40 hours (one week work). At the time of study, TSoft did not have a process in place to log the actual effort spent on each task. It was agreed during this study that the team members will log their time in the same spreadsheet that included the estimation checklist. Therefore, we can not compare the accuracy of checklist estimates with the accuracy of the previous estimates.

Checklist usefulness based on subjective feedback:

TSoft team members shared several benefits of using checklist to estimate tasks (see Table 7.9). Besides the common benefits that were also shared by the Infoway and Diyatech cases, TSoft team opined that the use of checklist resulted in useful discussions on the technical and architectural complexity of the tasks. The development team rated the 13 factors (see Table 7.10) in the TSoft revised checklist for their relevance to the estimation. They rated seven factors as extremely relevant, while the remaining six as relevant.

Checklist usefulness based on effort data:

The mean and median BREbias for the 21 tasks estimated with the checklist is -0.04 and -0.05. The accuracy values indicate the presence of minor overoptimism bias. The
developers, during their interviews earlier, opined that previously the most common scenario had been the presence of underestimation bias. The use of checklist may have lead to the decrease in the over optimism in their estimates. However, due to the unavailability of the previous data and also limited new data (only 21 tasks) we can not conclude anything in this case.

“We get more understanding of the task type by going through these factors. The task effort is estimated more accurately by considering these factors.”

Developer 1

“I think checklist help to understand the task basically wherever similar task show on sprint I get an idea how much time I needed to complete this task.”

“...this checklist improve my confidence while estimating the task.”

Developer 2

“Improvement in estimation and also helpful to understand the complexity of task and complete a task on time.”

Developer 3

7.6 Discussion

In this section first, we further discuss the results of the two research questions. Later, we describe the implications of this work for both SE researchers and practitioners.

7.6.1 RQ1: Checklist development and evolution

We followed a multi-step process to develop and evolve estimation checklists for the three cases included in this work. The process starts with the understand estimation context activity, wherein we elicited the details of the estimation process followed, and the factors that are considered during estimation through semi-structured interviews with case participants. The results (see Table 7.5) show that there are a number of reasons for effort overruns that are common across three cases. These common reasons include lack of details in requirements, changing requirements, overestimating team’s technical competence and domain knowledge, missing tasks, ignoring non-functional requirements, architectural complexity of tasks and task size. These reasons for effort overruns have been identified in other studies as well (cf. [3, 42, 43]). There were, however, some reasons which were not common in all cases, such as involvement of external interfaces.

The next activity is the develop and present checklist, wherein the first draft of the estimation checklist was presented in each case to the involved teams in a workshop. The aim is to present the checklist, and describe how it can be used and more
importantly take initial feedback to revise it. In this phase, participants in each case suggested changes (see Figure 7.2) to customize checklists to their respective contexts. These changes should be embraced and welcomed, as it indicates the study participants’ interest in the proposed checklist. At this stage, the participants have not yet used the checklist during effort estimation. It is likely that when they actually use checklist, they want to further customize the checklist. The next activity, validate checklist statistically, provide this opportunity to the case participants to use the estimation checklist on a sample of previous tasks and provide feedback for further changes. In all three cases, the participants used the checklist on a small sample of tasks and recommended further changes. For example, in Diyatech case the development manager suggested to include a factor about requirement’s priority in the checklist, and in case of TSoft the team suggested to remove a factor about team’s recent productivity. This type of validation in a controlled setting has been suggested by Hales et al. [40] as well in the development of medical checklists.

During the next activity, validate checklist dynamically, the teams use the checklist in real context to estimate tasks/requirements for the next release/sprint. When the teams use the checklist in real context, they may feel the need to further fine tune the checklists. After the dynamic validation phase, the teams were requested to provide their feedback on the estimation checklists through a feedback instrument (see Checklist feedback instrument in Appendix 7B: Data Collection Instruments) consisting of both open and close ended questions. The participants from two cases (Infoway and Diyatech) suggested further changes (see Figure 7.2), while TSoft did not ask for more changes. These changes further customized the checklist to the context of the respective teams and companies.

One of the concern of the Infoway teams was about time overhead due to the use of checklist during estimation sessions. In some cases, the teams have to estimate quickly due to the limited time available for estimation. Hales et al. [40] suggested to avoid having long checklists, while Jørgensen [15] recommended that the checklist should only consist of relevant content. In a follow up discussion with the Infoway team leads, we came up with the idea to classify the checklist factors into two parts: mandatory and optional. The mandatory part consist of that minimum set of factors that are to be considered in all situations, while the optional part consist of factors that will be considered subject to the availability of the time. Infoway products are relatively old products, and considerable amount of legacy code has been developed. To account for that, the Infoway teams suggested to include an additional factor about legacy code in the mandatory part of the checklist.

Diyatech development manager suggested to add two more factors in their checklist after using it to estimate requirements of the next release. First one is about the involvement of complex distributed algorithms, which is a common scenario in the
The last phase of the process is **Transfer and follow up**, wherein the revised checklist is handed over to the involved team for future use. It is up to the company management to decide if they want to include the checklist in their estimation process. In each case after dynamic validation, we handed over the revised checklists to our contacts in the three companies. At that stage, all had expressed their interest to keep using the checklist in future as well. As part of the follow up, we contacted the Infoway team leads and TSoft project manager. Infoway team leads informed us that the checklist has become a part of their estimation process. It is being used by the two teams during estimation sessions. The checklist has also been translated to Porterhouse to support those team members who feel more comfortable with their native language. TSoft project manager informed us that the team does not use the checklist anymore during estimation. However, he himself has been using checklist factors in prioritizing product backlog and sprint planning. The checklist factors support him in performing impact analysis of the backlog items. Dynamic validation phase of the Diyatech case was concluded relatively recently. Therefore, no follow up has yet been performed in this case.

## 7.6.2 RQ2: Usefulness of the checklist in improving effort estimation

In all three cases, the participants provided a highly positive feedback after using the checklist in real context. They noted several benefits (see Table 7.9) of using checklist during effort estimation. Checklist factors were found to be useful in reminding teams of the important factors that should be considered during effort estimation. The checklist introduced some structure in the otherwise largely ad-hoc expert estimation process that was being practiced in all three cases. One of the reasons for effort overruns, shared by participants from all three cases, was forgetting tasks during the task definition/decomposition process. The participants observed that the checklist factors such as those related to non-functional requirements, testing and tasks classification into company specific types, has reduced the chances of missing tasks. These benefits of estimation checklists have been reported earlier as well [11]. Checklists have been found to be beneficial in avoiding human error in omitting relevant activities and adherence to best practices in other disciplines as well, such as aviation, product manufacturing and health care [26].

The study participants noted that the explicit consideration of checklist factors during estimation has increased their confidence in effort estimates. Passing and Shep-
Chapter 7. Developing and Using Checklists to Improve Expert Effort Estimation: A Multi-Case Study

perd [17] also noted that the use of checklist improves estimators’ confidence in effort estimates.

Another benefit of using checklist noted by participants is about improved understandability of the task/requirement being estimated. The participants believe that the checklist factors induced discussion on different topics, such as impact on architecture, involvement of legacy code or non-functional requirements and required skills, which resulted in much better understanding of the tasks/requirements. Not that these factors were novel or were never considered before during effort estimation, making them part of estimation process through checklist made their use more consistent. Previously, they were discussed occasionally when someone from the team is able to recall particular factors during estimation.

The use of checklist has reduced the underestimation bias in two cases (Infoway and Diyatech), and resulted in more accurate estimates. The manager and team leaders from these two cases attributed this improvement to the use of the checklist, which forces them to consider these factors during estimation. Previously, these factors were not considered consistently. Furulund and Moløkken-Østvold [16] also observed that the checklist improved the estimation accuracy. In case of TSoft, we do not have the previous data to perform this comparison. However, the available effort data for just 21 tasks, that were estimated using checklist, show that the estimates are quite accurate and have a minor overestimation bias.

7.6.3 Implications for research

During the course of these three case studies, we had the opportunity to closely work with the involved teams to develop and evolve estimation checklists to improve their expert estimation processes. We learned the following lessons during this multi-case study, which can be used by other researchers investigating similar topics:

- Provide multiple opportunities for practitioners’ feedback: We used a process that allowed case participants to provide their feedback on checklist at multiple stages (e.g., during first workshop, interviews after static and dynamic validation). Garousi et al. [44] also recommended to have effective communication with the industrial partner, and to run regular workshops and seminars. The initial feedback is not enough, as it just consist of participants’ first impressions on the relevance of checklist factors. When participants have used the checklist during effort estimation for some time, they can provide more useful suggestions for improvement. Therefore, it is important to try to have them use the checklist, preferably in real context, in order to get effective feedback. The on-site presence of the first author in two cases proved very helpful in establishing trust and
efficiently working together to develop and try initial checklist versions.

- Incorporate suggested changes: One checklist cannot fit all contexts. The context of the company and the teams differ with respect to the domain, product complexity and history, team member’s experience and skill etc. Therefore, it is important that the checklist is customized for the team context, and this customization should be based on the team’s feedback. At each stage, we made an effort to act on the participants’ feedback (e.g., add/remove a factor from checklist, improve scales or description) and reported back with the revised checklist.

- Agile context: Most development teams use a combination of agile practices to develop software. In agile software development context the focus is on using light-weight processes and tools. Therefore if the checklist has to be used by an agile team as support to expert estimation, it should not consist of long list of factors. Agile teams would have to use estimation checklist repeatedly on the same project with each iteration and/or release. We tried to include only those factors in each checklist, which were considered relevant by the team members. Even then, in Infoway case, we had to divide the checklist factors into mandatory and optional parts.

- Share results of analysis: It is also important to share the results of the data analysis with case participants. For example, if there is any improvement in the accuracy of the effort estimates. And, how the whole team thinks about the usefulness of the estimation checklist. Showing benefits of the proposed solutions to the industrial partner is a good practice to improve industry-academia collaboration [44].

7.6.4 Implications for practice

Expert estimation is most widely used estimation technique in software industry. It involves little documentation and formalism. And, instead of using some algorithm or model, the involved experts rely on their intuition to estimate software development effort. Despite of all these “popular” attributes of expert estimation, it could still be inconsistent and ad-hoc. The experts may consider one set of factors during estimation one day, and a different set of factors another day. Furthermore, which factors were actually considered in a specific case may also gets lost. Checklist can be used as a tool to overcome these problems associated with expert estimation. On one hand, checklist would ensure the consistent consideration of the set of factors during effort estimation. One the other hand, it also documents, at low cost, what was considered during estimation of different stories and requirements. In three cases that we investigated in this
study, the participants found the idea of using checklist during estimation beneficial. The involved companies are from different domains, and their size varies from small to medium.

As noted before as well, in order for checklist to be useful, it has to be customized for the involved team and company context. This can only be realized if the involved team actively works with the researchers to provide them their feedback, and use the checklist according to the provided guidelines. Apart from the qualitative information through interviews and discussions, effort estimation data (i.e., actual effort, effort estimates) is required to be able to perform quantitative analysis. For example, in order to see if the checklist has improved accuracy of the effort estimates, effort data for the past and current sprints/release is required.

The proposed process (see Figure 7.1) can be used by the managers to prepare customized checklists for their teams. They can start with arranging workshops or discussions with the practitioners involved in the estimation process to elicit the reasons for the lack of accuracy and consistency in their effort estimates. Through these discussions, an initial checklist consisting of the factors that should be considered during estimation can be prepared. It is crucial that the team develops a shared understanding of the checklist factors, and how to use them during effort estimation. Next, the involved team should test the checklist on trial basis in few sprints. This pilot testing is vital in understanding how to use and integrate the checklist in the estimation process, and also in revising them using team’s feedback. We observed during these studies that the suggestion (cf. [15, 40]) to keep the checklists brief and relevant is crucial to ensure that the team members remain motivated in using the checklist. Lastly, it is also essential for the managers and teams to remember that the checklist should not be treated as a static document. In the light of their experiences and analysis, the checklist should continue to evolve.

7.7 Validity threats

We discussed the validity aspects of our study using the classification adopted by Runeson and Höst [1] in their guidelines from Yin [18] and Wohlin et al [19].

7.7.1 Construct validity

Construct validity is about ensuring if the measures used actually represent the concepts that researchers want to investigate and what is described in the research questions. In our case, for example, we discussed two effort estimation scenarios (under and overestimation) and their possible reasons with interviewees. In order to make sure that the
interviewees do not interpret them in a different way, we provided them with examples to explain what we mean by under and overestimation (cf, scenarios used in written interview questions in Appendix 7B: Data Collection Instruments).

The proposed estimation checklist contain number of factors and corresponding scale values that are to be used by the study participants during estimation. We took number of step to ensure that each factor is understood and interpreted correctly and consistently. These steps include presenting each factor and associated scale in a workshop, written description notes for each factor to explain what is meant by it and how to use the associated scale, static validation and the follow up discussion to ensure shared understanding.

### 7.7.2 Internal validity

Internal validity relates to the possibility of confounding factors impacting the investigated factor, wherein the researcher is either unaware of the existence of these factors or their impact on the investigated factor. When we elicited the reasons for effort overruns, in each case we involved multiple participants (source triangulation) to ensure that we have not missed out anything. Furthermore, after dynamic validation step, the metrics data showed improvement in the accuracy of the effort estimates in two cases (Infoway and Diyatech cases). We conducted follow up discussions (method triangulation) to request the case participants to explain the results, and to see if the improvements could actually be attributed to some other factor. In both cases, the participants attributed the improvements to the use of checklists.

### 7.7.3 External validity

External validity is about the extent to which the study findings are generalizeable and applicable to cases and people outside the investigated case. We investigated three cases in this study. Our findings are limited to the type of cases we studied and their context. Two companies (Infoway and Diyatech) are medium sized relatively old companies, while the third one (Tsoft) is a new and small offshore development centre of a parent company in a European country. The companies are developing software in different domains (eHealth, performance enhancement and transportation). Two common aspects in these three studied cases are: expert estimation and agile practices. We can therefore say that our findings may be of interest to the small to medium software companies that use expert estimation to estimate development effort in an agile software development context.
Chapter 7. Developing and Using Checklists to Improve Expert Effort Estimation: A Multi-Case Study

7.7.4 Reliability

Reliability is concerned with the degree to which the data and the analysis are dependent on the researcher. We involved multiple researchers during the study planning, data collection and analysis phases. We also involved the case participants (company director for Infoway, development manager for Diyatech, project manager for TSoft) in reviewing the study protocol before the start of the data collection phase. However, since the work is lead by the first author (PhD student), we can not be certain that the data and analysis are completely independent of the first author. The involvement of other researchers in all study phases would have reduced this dependency, if not completely eliminated it.

We were able to obtain limited effort data both before and after the use of checklist. In case of TSoft, previous effort data was not available. Therefore, we could not compare in quantitative terms the impact of checklist on the accuracy of the effort estimates. Furthermore, in case of Diyatech we have data about two releases only, one without the checklist use and one where checklist was used. This is not enough to be able to draw any conclusions. In case of Infoway we have relatively more data, but that too is not enough to conclude anything. However, it was good to observe during the follow up interaction with these companies, that they are still using the checklists. We hope we will be able to have more data, both qualitative and quantitative, in future to be able to perform more analysis.

7.8 Conclusions and future work

We reported a multi-case study to develop and evolve estimation checklists to improve their expert estimation processes. The three companies involved in the reported multi-case study are: Infoway (a medium-sized company in Brazil developing products in eHealth domain), Diyatech (a medium-sized company in Pakistan developing performance enhancement applications for distributed environments) and TSoft (small and a new offshore development center of a parent company in Norway, developing application in the transportation domain).

In RQ1, we proposed a multi-step process to develop and evolve checklist to improve expert estimation of software development effort. The application of the process was demonstrated in three companies. The checklists were developed and evolved based on the feedback from the participants in all cases. After dynamically validating the checklists in the real context, we transferred the revised checklists to the involved teams. We performed a follow-up step after a gap of six months in two cases. In case of Infoway teams, we got to know that the checklist has become a part of their estimation
process. In case of TSoft, the development team was not using the checklist anymore. However, the project manager was using checklist factors to prioritize product backlog. We did not perform any follow up with Diyatech, as the checklist was transferred to them only recently after dynamic validation.

In **RQ2**, based on a combination of qualitative and quantitative data, we showed the usefulness of checklists in improving the expert effort estimation. After using the checklist in the real context, participants from all cases shared that they found the checklist extremely useful in improving the estimation process. The commonly observed benefits of estimation checklists were: improved task understandability, increased confidence in effort estimates, fewer chances of missing tasks and factors to be considered, and more objectivity in the expert estimation process. The analysis of the effort data from two cases showed that the use of checklist had improved the accuracy of the effort estimates, and the underestimation bias was also reduced, which was one of the common challenges for all three companies before the introduction of checklists.

As part of our future work, we plan to collect more data from these companies to see how useful have been the checklists in the long run. We will also try to identify possibilities for further improvements related to the checklists. We plan to apply proposed checklist development process in a large Swedish telecommunication software vendor, which has recently shown interest in using our work to improve their expert estimation process.
Chapter 7. Developing and Using Checklists to Improve Expert Effort Estimation: A Multi-Case Study

7.9 References


261
Chapter 7. Developing and Using Checklists to Improve Expert Effort Estimation: A Multi-Case Study


Chapter 7. Developing and Using Checklists to Improve Expert Effort Estimation: A Multi-Case Study


### Table 7A.1: Infoway checklist V2 (after initial feedback and static validation)

<table>
<thead>
<tr>
<th>No.</th>
<th>Checklist factor</th>
<th>Scale</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>What is the type of the task?</td>
<td>Implementation, testing, research</td>
</tr>
<tr>
<td>2</td>
<td>The task is implementing a Functional Requirement (FR) or Non-Functional Requirement (NFR) or both?</td>
<td>Only an NFR, FR with major NFR constraints, FR with minor NFR constraints, only FR</td>
</tr>
<tr>
<td>3</td>
<td>How similar is the task to the previously developed tasks</td>
<td>To a great extent, somewhat, very little, not at all</td>
</tr>
<tr>
<td>4</td>
<td>Rank team’s technical competence/skills required to implement and test this task</td>
<td>Very good, good, average, below average, lacking</td>
</tr>
<tr>
<td>5</td>
<td>Rank team’s domain knowledge required to implement this task</td>
<td>Very good, good, average, below average, lacking</td>
</tr>
<tr>
<td>6</td>
<td>Does the task require an architectural change?</td>
<td>Yes, not sure, no</td>
</tr>
<tr>
<td>7</td>
<td>Does the task involve communication between multiple (sub) systems?</td>
<td>Yes, not sure, no</td>
</tr>
<tr>
<td>8</td>
<td>Does the task involve accessing and/or modifying several different elements/tables in the persistence/DB layer?</td>
<td>Yes, not sure, no</td>
</tr>
<tr>
<td>9</td>
<td>How clear is your understanding of the task?</td>
<td>Very good, good, average, below average, need to talk to customer before moving on</td>
</tr>
<tr>
<td>10</td>
<td>Is the task size suitable to fit in the planned sprint</td>
<td>Yes, not sure, no</td>
</tr>
<tr>
<td>11</td>
<td>Will the team or some member(s) be working on other products/projects/tasks in parallel?</td>
<td>Yes, not sure, no</td>
</tr>
<tr>
<td>12</td>
<td>Does the team has new member(s)?</td>
<td>Yes, no</td>
</tr>
<tr>
<td>13</td>
<td>Rank team’s recent productivity</td>
<td>Very good, good, average, below average, lacking</td>
</tr>
</tbody>
</table>

Based on the above characterizations, most likely effort estimate for this task is: ———– hours
Table 7A.2: Infoway checklist V3 (prioritized and classified after dynamic validation)

<table>
<thead>
<tr>
<th>No.</th>
<th>Checklist factor</th>
<th>Scale</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>Mandatory Part</strong></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>What is the type of the task?</td>
<td>Implementation, testing, research</td>
</tr>
<tr>
<td>2</td>
<td>The task is implementing a Functional Requirement (FR) or Non-Functional Requirement (NFR) or both?</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Only an NFR, FR with major NFR constraints, FR with minor NFR constraints, only FR</td>
</tr>
<tr>
<td>3</td>
<td>Rank team’s technical competence/skills required to implement and test this task</td>
<td>Very good, good, average, below average, lacking</td>
</tr>
<tr>
<td>4</td>
<td>How similar is the task to the previously developed tasks?</td>
<td>To a great extent, somewhat, very little, not at all</td>
</tr>
<tr>
<td>5</td>
<td>Rank team’s domain knowledge required to implement this task</td>
<td>Very good, good, average, below average, lacking</td>
</tr>
<tr>
<td>6</td>
<td>How clear is your understanding of the task?</td>
<td>Very good, good, average, below average, need to talk to customer before moving on</td>
</tr>
<tr>
<td>7</td>
<td>The implementation of the task requires understanding and/or changing legacy code</td>
<td>Yes, not sure, no</td>
</tr>
<tr>
<td>8</td>
<td>Does the task involve communication between multiple (sub) systems?</td>
<td>Yes, not sure, no</td>
</tr>
<tr>
<td></td>
<td><strong>Optional Part</strong></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>Does the task require an architectural change?</td>
<td>Yes, not sure, no</td>
</tr>
<tr>
<td>10</td>
<td>Does the team has new member(s)?</td>
<td>Yes, no</td>
</tr>
<tr>
<td>11</td>
<td>Will the team or some member(s) be working on other products/projects/tasks in parallel?</td>
<td>Yes, not sure, no</td>
</tr>
<tr>
<td>12</td>
<td>Does the task involve accessing and/or modifying several different elements/tables in the persistence/DB layer?</td>
<td>Yes, not sure, no</td>
</tr>
<tr>
<td>13</td>
<td>Is the task size suitable to fit in the planned sprint?</td>
<td>Yes, not sure, no</td>
</tr>
<tr>
<td>14</td>
<td>Rank team’s recent productivity</td>
<td>Very good, good, average, below average, lacking</td>
</tr>
</tbody>
</table>

Based on the above characterizations, most likely effort estimate for this task is: ———– hours
<table>
<thead>
<tr>
<th>No.</th>
<th>Checklist factor</th>
<th>Scale</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>The requirement is classified as:</td>
<td>x, y, z (company specific categories)</td>
</tr>
<tr>
<td>2</td>
<td>How similar is the requirement to the previously developed requirements?</td>
<td>To a great extent, somewhat, very little, not at all</td>
</tr>
<tr>
<td>3</td>
<td>Rank team’s technical competence required to implement this requirement</td>
<td>Very good, good, average, below average, lacking</td>
</tr>
<tr>
<td>4</td>
<td>Rank team’s technical competence required to test this requirement</td>
<td>Very good, good, average, below average, lacking</td>
</tr>
<tr>
<td>5</td>
<td>Rank team’s domain knowledge required to implement this requirement</td>
<td>Very good, good, average, below average, lacking</td>
</tr>
<tr>
<td>6</td>
<td>Rank team’s recent productivity</td>
<td>Very good, good, average, below average, lacking</td>
</tr>
<tr>
<td>7</td>
<td>Does the team has new member(s)?</td>
<td>Yes, no</td>
</tr>
<tr>
<td>8</td>
<td>Will this requirement’s implementation involve an architectural change?</td>
<td>Yes, not sure, no</td>
</tr>
<tr>
<td>9</td>
<td>Will this requirement’s implementation involve communication between multiple (sub) systems?</td>
<td>Yes, not sure, no</td>
</tr>
<tr>
<td>10</td>
<td>The requirement is classified as Functional Requirement (FR) or Non-Functional Requirement (NFR) or both?</td>
<td>Only an NFR, FR with major NFR constraints, FR with minor NFR constraints, only FR</td>
</tr>
<tr>
<td>11</td>
<td>How clear is your understanding of the requirement?</td>
<td>Very good, good, average, below average, need to talk to customer before moving on</td>
</tr>
<tr>
<td>12</td>
<td>Is the requirement size suitable to fit in the planned release?</td>
<td>Yes, not sure, no</td>
</tr>
<tr>
<td>13</td>
<td>What is the priority of the requirement?</td>
<td>Company specific priority categories</td>
</tr>
</tbody>
</table>

Based on the above characterizations, specify your most likely effort estimate for the following activities:

- Requirements analysis ——— hrs
- Documentation ——— hrs
- Design ——— hrs
- Coding ——— hrs
- Testing ——— hrs
- Non-development work ——— hrs

Total ——— hrs
Table 7A.4: Diyatech checklist V3 (after dynamic validation)

<table>
<thead>
<tr>
<th>No.</th>
<th>Checklist factor</th>
<th>Scale</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>The requirement is classified as:</td>
<td>x, y, z (company specific categories)</td>
</tr>
<tr>
<td>2</td>
<td>How similar is the requirement to the previously developed requirements?</td>
<td>To a great extent, somewhat, very little, not at all</td>
</tr>
<tr>
<td>3</td>
<td>Rank team’s technical competence required to implement this requirement</td>
<td>Very good, good, average, below average, lacking</td>
</tr>
<tr>
<td>4</td>
<td>Rank team’s technical competence required to test this requirement</td>
<td>Very good, good, average, below average, lacking</td>
</tr>
<tr>
<td>5</td>
<td>Rank team’s domain knowledge required to implement this requirement</td>
<td>Very good, good, average, below average, lacking</td>
</tr>
<tr>
<td>6</td>
<td>Rank team’s recent productivity</td>
<td>Very good, good, average, below average, lacking</td>
</tr>
<tr>
<td>7</td>
<td>Does the team has new member(s)?</td>
<td>Yes, no</td>
</tr>
<tr>
<td>8</td>
<td>Will this requirement’s implementation involve an architectural change?</td>
<td>Yes, no</td>
</tr>
<tr>
<td>9</td>
<td>Will this requirement’s implementation involve communication between multiple (sub) systems?</td>
<td>Yes, not sure, no</td>
</tr>
<tr>
<td>10</td>
<td>The requirement is classified as Functional Requirement (FR) or Non-Functional Requirement (NFR) or both?</td>
<td>Only an NFR, FR with major NFR constraints, FR with minor NFR constraints, only FR</td>
</tr>
<tr>
<td>11</td>
<td>How clear is your understanding of the requirement?</td>
<td>Very good, good, average, below average, need to talk to customer before moving on</td>
</tr>
<tr>
<td>12</td>
<td>Is the requirement size suitable to fit in the planned release?</td>
<td>Yes, not sure, no</td>
</tr>
<tr>
<td>13</td>
<td>What is the priority of the requirement?</td>
<td>Company specific priority categories</td>
</tr>
<tr>
<td>14</td>
<td>Will this requirement's implementation involve designing and/or revising and/or implementing distributed algorithm(s)?</td>
<td>Yes, not sure, no</td>
</tr>
<tr>
<td>15</td>
<td>Do you think there may be interruptions during the planned release/sprint affecting team member's momentum due to context switch?</td>
<td>Yes, not sure, no</td>
</tr>
</tbody>
</table>

Based on the above characterizations, specify your most likely effort estimate for the following activities:

<table>
<thead>
<tr>
<th>Requirements analysis</th>
<th>——— hrs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Documentation</td>
<td>——— hrs</td>
</tr>
<tr>
<td>Design</td>
<td>——— hrs</td>
</tr>
<tr>
<td>Coding</td>
<td>——— hrs</td>
</tr>
<tr>
<td>Testing</td>
<td>——— hrs</td>
</tr>
<tr>
<td>Non-development work</td>
<td>——— hrs</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td>——— hrs</td>
</tr>
</tbody>
</table>
### Table 7A.5: Tsoft checklist V2 (after initial feedback and static validation)

<table>
<thead>
<tr>
<th>No.</th>
<th>Checklist factor</th>
<th>Scale</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>What is the type of the task?</td>
<td>UI, Business logic, DB, UI and Business logic, Business logic and DB, All</td>
</tr>
<tr>
<td>2</td>
<td>How clear is your understanding of the task?</td>
<td>Very good, good, average, below average, lacking, need to talk to product owner before moving on</td>
</tr>
<tr>
<td>3</td>
<td>How similar is the task to the previously developed tasks</td>
<td>To a great extent, somewhat, very little, not at all</td>
</tr>
<tr>
<td>4</td>
<td>Rank team’s technical competence/skills required to implement and test this task</td>
<td>Very good, good, average, below average, lacking</td>
</tr>
<tr>
<td>5</td>
<td>Rank team’s domain knowledge required to implement this task</td>
<td>Very good, good, average, below average, lacking</td>
</tr>
<tr>
<td>6</td>
<td>Will the task implementation likely to involve interfacing with external services/systems?</td>
<td>Yes, not sure, no</td>
</tr>
<tr>
<td>7</td>
<td>Does the task require an architectural change?</td>
<td>Yes, not sure, no</td>
</tr>
<tr>
<td>8</td>
<td>Does the task involve communication between multiple (sub) systems?</td>
<td>Yes, not sure, no</td>
</tr>
<tr>
<td>9</td>
<td>Does the task involve accessing and/or modifying several different elements/tables in the persistence/DB layer?</td>
<td>Yes, not sure, no</td>
</tr>
<tr>
<td>10</td>
<td>Are there stringent non-functional requirements (performance, security etc.) associated with this task?</td>
<td>Yes, not sure, no</td>
</tr>
<tr>
<td>11</td>
<td>Will the team or some member(s) be working on other products/projects/tasks in parallel?</td>
<td>Yes, not sure, no</td>
</tr>
<tr>
<td>12</td>
<td>Is the task size suitable to fit in the planned sprint</td>
<td>Yes, not sure, no</td>
</tr>
</tbody>
</table>

Based on the above characterizations, most likely effort estimate for this task is: ——— hours
Table 7A.6: Tsoft checklist V3 (after follow up discussion with project manager)

<table>
<thead>
<tr>
<th>No.</th>
<th>Checklist factor</th>
<th>Scale</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>What types of tasks are likely to be involved for this backlog item?</td>
<td>UI, Business logic, DB, UI and Business logic, Business logic and DB, All</td>
</tr>
<tr>
<td>2</td>
<td>How clear is your understanding of the backlog item?</td>
<td>Very good, good, average, below average, lacking, need to talk to product owner before moving on</td>
</tr>
<tr>
<td>3</td>
<td>How similar is the item to the previously developed items</td>
<td>To a great extent, somewhat, very little, not at all</td>
</tr>
<tr>
<td>4</td>
<td>Rank team’s technical competence/skills required to implement and test this backlog item</td>
<td>Very good, good, average, below average, lacking</td>
</tr>
<tr>
<td>5</td>
<td>Rank team’s domain knowledge required to implement this backlog item</td>
<td>Very good, good, average, below average, lacking</td>
</tr>
<tr>
<td>6</td>
<td>Does the implementation of the backlog item likely to involve interfacing with external services/systems</td>
<td>Yes, not sure, no</td>
</tr>
<tr>
<td>7</td>
<td>Does the implementation of the backlog item likely to require an architectural change?</td>
<td>Yes, not sure, no</td>
</tr>
<tr>
<td>8</td>
<td>Does the implementation of the backlog item likely to involve communication between multiple (sub) systems?</td>
<td>Yes, not sure, no</td>
</tr>
<tr>
<td>9</td>
<td>Does the implementation of the backlog item likely to involve accessing and/or modifying several different elements/tables in the persistence/DB layer?</td>
<td>Yes, not sure, no</td>
</tr>
<tr>
<td>10</td>
<td>Are there stringent non-functional requirements (performance, security etc.) associated with this backlog item?</td>
<td>Yes, not sure, no</td>
</tr>
<tr>
<td>11</td>
<td>Will all or some member(s) of team be working on other products/projects/tasks in parallel?</td>
<td>Yes, not sure, no</td>
</tr>
<tr>
<td>12</td>
<td>Does the size of the backlog item suitable to fit in one sprint</td>
<td>Yes, not sure, no</td>
</tr>
</tbody>
</table>

Based on the above characterizations, most likely effort estimate for this task is: ———– hours
Appendix 7B: Data Collection Instruments

Interview questions about effort estimation at Diyatech

Please provide information about your role and experience in Table 1

<table>
<thead>
<tr>
<th>Your current role</th>
<th>Experience in current role (in Years)</th>
<th>Previous roles in this company, if any</th>
<th>Total experience in this company</th>
<th>Experience before this company, if any</th>
</tr>
</thead>
</table>

There are two common scenarios (see Table 2) with respect to the accuracy of the effort estimates: underestimation (Scenario 1) and overestimation (Scenario 2). Underestimation is also referred as effort overrun or delay.

<table>
<thead>
<tr>
<th>Scenario 1 (underestimation)</th>
<th>Scenario 2 (overestimation)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Effort estimate: 25 Hours</td>
<td>Effort estimate: 25 Hours</td>
</tr>
<tr>
<td>Actual effort: 40 Hours</td>
<td>Actual effort: 15 Hours</td>
</tr>
</tbody>
</table>

1. How frequently these two scenarios occur in your product? Which one is more frequent?
2. In your opinion, what are the main reasons for software effort overruns, i.e. scenario 1, in your context?
3. In your opinion, what are the main reasons for overestimation scenario, i.e. scenario 2, in your context?
4. What are the main challenges/difficulties in estimating the development effort?
5. How effort is estimated in your product/company? Briefly describe main steps.
6. Which factors do you consider in estimating the effort of a requirement or story?
7. What role, if any, do you play in the effort estimation process?
8. How to improve the current effort estimation process? What changes should be made, in your opinion, in the current effort estimation process?

Developers are only asked Q1 to Q3, as they are not directly involved in the estimation process. The architect and the development manager were requested to answer all questions, as they participate in the estimation process.
Chapter 7. Developing and Using Checklists to Improve Expert Effort Estimation: A Multi-Case Study

**Effort Estimation Checklist Feedback Instrument**

First of all, we reproduced the relevant checklist here to facilitate the participants in answering feedback questions.

**Evaluation of the proposed effort estimation checklist by practitioners**

1. **Questions about each checklist item**

   Provide your opinion on each checklist factor with respect to their relevance to the effort estimation. Mark your choice by writing “Yes” in the cell of your choice.

   ![Checklist Table]

2. **Suggestions for improvement** (This part is really important, as your suggestions will guide us in improving the proposed checklist)

   2.1 Are there any other factors that should be added to the checklist? If yes, then kindly list those factors here, also state why do you recommend adding them.
   
   *Space for the answer*

   2.2 Are there checklist items that should be removed from the checklist? If yes, then kindly list those items here and also state why do you recommend removing them?
   
   *Space for the answer*

   2.3 What are the benefits of using this estimation checklist?
   
   *Space for the answer*

   2.4 What are the drawbacks/limitations of this checklist in your opinion? How can we improve/revise this checklist? Please feel free to provide your frank opinion, as it will help us in improving our work.
   
   *Space for the answer*

   2.5 Are you interested in using the checklist in future sprints/releases?
   
   *Space for the answer*