

Alhamed, Mohammed (2022) *On the application of artificial intelligence and human computation to the automation of agile software task effort estimation*. PhD thesis.

<https://theses.gla.ac.uk/83231/>

Copyright and moral rights for this work are retained by the author

A copy can be downloaded for personal non-commercial research or study, without prior permission or charge

This work cannot be reproduced or quoted extensively from without first obtaining permission in writing from the author

The content must not be changed in any way or sold commercially in any format or medium without the formal permission of the author

When referring to this work, full bibliographic details including the author, title, awarding institution and date of the thesis must be given



ON THE APPLICATION OF ARTIFICIAL  
INTELLIGENCE AND HUMAN  
COMPUTATION TO THE AUTOMATION OF  
AGILE SOFTWARE TASK EFFORT  
ESTIMATION.

MOHAMMED ALHAMED

SUBMITTED IN FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE OF  
*Doctor of Philosophy*

SCHOOL OF COMPUTING SCIENCE  
COLLEGE OF SCIENCE AND ENGINEERING  
UNIVERSITY OF GLASGOW

OCTOBER, 2022

© MOHAMMED ALHAMED

## **Abstract**

Software effort estimation (SEE), as part of the wider project planning and product road mapping process, occurs throughout a software development life cycle. A variety of effort estimation methods have been proposed in the literature, including algorithmic methods, expert based methods, and more recently, methods based on techniques drawn from machine learning and natural language processing. In general, the consensus in the literature is that expert-based methods such as Planning Poker are more reliable than automated effort estimation. However, these methods are labour intensive and difficult to scale to large-scale projects.

To address this limitation, this thesis investigates the feasibility of using human computation techniques to coordinate crowds of inexpert workers to predict expert-comparable effort estimates for a given software development task. The research followed an empirical methodology and used four different methods: literature review, replication, a series of laboratory experiments, and ethnography.

The literature uncovered the lack of suitable datasets that include the attributes of descriptive text (corpus), actual cost, and expert estimates for a given software development task. Thus, a new dataset was developed to meet the necessary requirements.

Next, effort estimation based on recent natural language processing advancements was evaluated and compared with expert estimates. The results suggest that there was no significant improvement, and the automated approach was still outperformed by expert estimates. Therefore, the feasibility of scaling the Planning Poker effort estimation method by using human computation in a micro-task crowdsourcing environment was explored. A series of pilot experiments were conducted to find the proper design for adapting Planning Poker to a crowd environment.

This resulted in designing a new estimation method called Crowd Planning Poker (CPP). The pilot experiments revealed that a significant proportion of the crowd submitted poor quality assignments. Therefore, an approach to actively managing the quality of SEE work was proposed and evaluated before being integrated into the CPP method. A substantial

overall evaluation was then conducted. The results demonstrated that crowd workers were able to discriminate between tasks of varying complexity and produce estimates that were comparable with those of experts and at substantially reduced cost compared with small teams of domain experts.

It was further noted in the experiments that crowd workers provide useful insights as to the resolution of the task. Therefore, as a final step, fine-grained details about crowd workers' behaviour, including actions taken and artifacts reviewed, were used in an ethnographic study to understand how crowd effort estimation takes place in a crowd. Four persona archetypes were developed to describe the crowd behaviours, and the results of the behaviour analysis were confirmed by surveying the crowd workers.

## **Acknowledgements**

In honour of those people without whom it would not have been possible for me to complete this research and write this thesis.

My sincere gratitude goes out to my parents, Abdullah Alhamed and Norah Albahly for their endless support, encouragement, and patience throughout my life and especially during my PhD. Mom, I will not forget your constant calls and prayers to ensure a comfortable life for me. You always have a place for me. Remember, Dad, you are the one who ignited this ambition in me, and kept reminding me of the end goal. Without your encouragement and support, this work wouldn't have seen the light. I also would like to thank my wife, Samiah Aljadhah, for not only standing by me, but also for taking great care of me and our children, Husam, Norah, Sarah, and Lara.

Thank you to my supervisor, Dr Tim Storer, for his patience, guidance, and support. You have provided me with a wealth of knowledge, a profound way of thinking and questioning, and meticulous examining and editing that I would never imagined before. You have literally transformed my skills in scientific research. You are always there for endless support and guidance, be it editing my writing or recommending me in your letters. Throughout the years, you kept the research trails fascinating for me as I passed through this journey's trials. I have been extremely grateful that you have taken me on as a student and continued to believe in me. I also thank my second supervisor, Dr Inah Omoronyia, for his great feedback, excellent encouragement, and guidance.

Thank you to Caroline Orr<sup>1</sup> for proofreading the thesis and helping with the language, the University of Jeddah for sponsoring this research, and the School of Computing Science at the University of Glasgow for their support.

---

<sup>1</sup>[www.orreditorial.com](http://www.orreditorial.com)

### **Declaration**

I declare that this thesis has been composed by myself, that the research presented embodies the results of my own work and that it does not include work forming part of a thesis presented for a degree in this or any other University.

The author's original work presented in this thesis has contributed to a number of publications that have been co-authored with Dr Timothy Storer:

- M. Alhamed and T. Storer, "Estimating Software Task Effort in Crowds," 2019 IEEE International Conference on Software Maintenance and Evolution (ICSME), 2019, pp. 281-285, doi: 10.1109/ICSME.2019.00042.
- M. Alhamed and T. Storer, "Playing Planning Poker in Crowds: Human Computation of Software Effort Estimates," 2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE), 2021, pp. 1-12, doi: 10.1109/ICSE43902.2021.00014.

# Table of Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Background . . . . .	1
1.1.1	Software Effort Estimation . . . . .	1
1.1.2	Planning Poker . . . . .	3
1.1.3	Human Computation and Crowdsourcing . . . . .	4
1.2	Motivation . . . . .	5
1.3	Thesis Statement . . . . .	7
1.4	Contribution . . . . .	9
1.5	Thesis Outline . . . . .	11
<b>2</b>	<b>Research Methodology</b>	<b>13</b>
2.1	Software Engineering Empirical Research . . . . .	13
2.2	Literature review . . . . .	15
2.3	Replication . . . . .	16
2.4	Empirical Experiments . . . . .	17
2.4.1	Pilot Experimentation . . . . .	18
2.4.2	Laboratory Experiments . . . . .	19
2.5	Ethnography . . . . .	19
2.6	Summary . . . . .	21
<b>3</b>	<b>Literature Review</b>	<b>23</b>
3.1	Characterising Software Effort Estimation . . . . .	23
3.1.1	Definitions . . . . .	24
3.1.2	Different Perspectives of Software Effort Estimation . . . . .	25

3.1.3	SEE Methods Classification . . . . .	26
3.2	Expert-based Software Effort Estimation Methods . . . . .	28
3.2.1	Guesstimation . . . . .	29
3.2.2	Wideband Delphi . . . . .	29
3.2.3	Estimeetings . . . . .	30
3.2.4	Stochastic Budget Simulation (SBS) . . . . .	31
3.2.5	Sparse Data Method (SDM) using Analytic Hierarchy Process . . .	32
3.2.6	Planning Poker . . . . .	33
3.3	Machine Learning Software Effort Estimation . . . . .	35
3.3.1	Artificial Neural Network (ANN) . . . . .	35
3.3.2	Case-Based Reasoning (CBR) . . . . .	35
3.3.3	Decision Tree (DT) . . . . .	36
3.3.4	Support Vector (SV) . . . . .	36
3.3.5	Performance of ML-Based . . . . .	37
3.3.6	Contexts Where ML-Based is Recommended . . . . .	37
3.4	Review of Comparative Research of Expert-Based and ML-Based SEE . . .	38
3.5	Human Computation . . . . .	40
3.5.1	Crowdsourcing . . . . .	41
3.5.2	Collective Intelligence . . . . .	42
3.5.3	Social Computing . . . . .	42
3.5.4	Quality Assignment in Human Computation . . . . .	43
3.6	Software Engineering Applications in Human Computation . . . . .	45
3.7	Summary . . . . .	47
<b>4</b>	<b>JIRA Open Source Software Effort Dataset</b>	<b>49</b>
4.1	Publicly Available SEE Datasets . . . . .	50
4.2	Software Effort Estimation Datasets Research Studies . . . . .	51
4.3	Collection of the JOSSE Dataset . . . . .	53
4.4	JOSSE Dataset Refinement Options . . . . .	57
4.4.1	Quantity of Data Points Per Project . . . . .	59
4.4.2	Dataset Outliers . . . . .	59



4.4.3	Dataset Dissension . . . . .	60
4.4.4	Dataset Readability . . . . .	62
4.4.5	Discretising Software Effort Estimates . . . . .	64
4.4.6	Dataset Domain and Origin . . . . .	65
4.4.7	The Quality Taxonomy Assessment . . . . .	65
4.5	Summary . . . . .	67
<b>5</b>	<b>Evaluation of Language-Based Transfer Model for Software Effort</b>	<b>68</b>
5.1	Background on ML and Software Effort Estimation . . . . .	69
5.1.1	Converting Text Corpus for Use in ML Models . . . . .	70
5.1.2	BERT and RF as Machine Learning Models . . . . .	72
5.2	Experiment Design . . . . .	75
5.2.1	Experiment Datasets . . . . .	76
5.2.2	Estimation Method . . . . .	79
5.2.3	Evaluation Metrics . . . . .	81
5.3	Results . . . . .	81
5.3.1	RQM1: Accuracy of ML models . . . . .	82
5.3.2	RQM2: Evaluation of Feature Extraction Methods . . . . .	83
5.3.3	RQM3: ML models compared with expert-based estimates . . . . .	84
5.4	Discussion . . . . .	85
5.5	Summary . . . . .	88
<b>6</b>	<b>Crowd Planning Poker: A Preliminary Study</b>	<b>89</b>
6.1	General Considerations of Planning Poker . . . . .	90
6.2	Crowd Planning Poker (CPP) General Model . . . . .	93
6.3	Experimental Design . . . . .	99
6.3.1	Dataset . . . . .	100
6.3.2	Measures . . . . .	101
6.3.3	Experiment Trials and Variables . . . . .	102
6.3.4	Evaluation and Result Test . . . . .	104
6.4	Results . . . . .	104

6.4.1	Information Experiment . . . . .	104
6.4.2	Crowd Size Experiment . . . . .	105
6.4.3	Process Design Experiment . . . . .	106
6.5	Discussion . . . . .	106
6.5.1	PRQ1: Proper working settings for CPP . . . . .	106
6.5.2	PRQ2: Proper process design for CPP . . . . .	108
6.5.3	PRQ3: Feasibility of producing expert-comparable estimates . . . . .	108
6.5.4	Beyond Estimates – Crowd Insights . . . . .	109
6.5.5	Evaluation of Costs in Pilot Studies . . . . .	110
6.5.6	Threat to Validity – Issue Availability . . . . .	111
6.6	Summary . . . . .	112
<b>7</b>	<b>Quality Assessment and Enhancement of Crowd Planning Poker</b>	<b>113</b>
7.1	Exploring the Quality of Crowd Assignments in the CPP Context . . . . .	114
7.2	Measuring Quality of Crowd Assignments . . . . .	118
7.2.1	User Behaviour Quality . . . . .	119
7.2.2	Manual Assessment of Issue Quality . . . . .	120
7.2.3	Machine Learning . . . . .	121
7.3	Improving Crowd Quality . . . . .	124
7.3.1	Crowd Feedback Loop . . . . .	124
7.3.2	Encouraging Improvement Using Loss Attention . . . . .	127
7.3.3	Handling Rejected Cases Using Soft-Reject . . . . .	128
7.3.4	Quality Improvement Experiment Design . . . . .	129
7.3.5	Experiment Results . . . . .	130
7.4	Discussion . . . . .	132
7.5	Summary . . . . .	135
<b>8</b>	<b>Human Computation of Software Effort Estimates</b>	<b>137</b>
8.1	Full Design of Crowd Planning Poker Using Human Computation . . . . .	138
8.2	Experimental Design . . . . .	147
8.3	Result and Evaluation . . . . .	150

8.3.1	Crowd Performance Compared with Experts . . . . .	151
8.3.2	CPP Scalability . . . . .	152
8.4	Discussion . . . . .	154
8.4.1	Threats to Validity – Issue Availability . . . . .	155
8.5	Summary . . . . .	156
<b>9</b>	<b>Crowd Estimator Personas: an Ethnographic Study of Crowd Behaviour</b>	<b>158</b>
9.1	Related Work . . . . .	159
9.2	Design of Systematic Behaviour Scanning Study . . . . .	162
9.2.1	UI Interaction Log Systematic Scanning . . . . .	163
9.2.2	Behaviour Descriptors for Crowd Personas . . . . .	168
9.3	Results of Behaviour Analysis and Crowd Personas . . . . .	171
9.3.1	Combined Class A and B . . . . .	171
9.3.2	Class C . . . . .	174
9.3.3	Class D . . . . .	177
9.3.4	Identified Personas . . . . .	178
9.4	Post CPP Survey Study . . . . .	180
9.4.1	Survey Design . . . . .	180
9.4.2	Survey Results . . . . .	185
9.5	Discussion . . . . .	189
9.6	Summary . . . . .	191
<b>10</b>	<b>Conclusions</b>	<b>193</b>
10.1	Summary of Research Activity . . . . .	193
10.2	Questions and Findings . . . . .	194
10.2.1	ML Algorithms . . . . .	195
10.2.2	CPP Design . . . . .	195
10.2.3	CPP Quality . . . . .	196
10.2.4	CPP Automation . . . . .	197
10.2.5	Crowd Behaviour . . . . .	198
10.3	Contributions and Learned Lessons . . . . .	198

10.4	Thesis Scope and Validity . . . . .	203
10.5	Future CPP Research Work . . . . .	205
10.5.1	Applying Crowd Planning Poker in an Industrial Case Study . . . .	205
10.5.2	Investigating the Effects of Obfuscating on Estimate Reliability . .	206
10.5.3	Extending Crowd Planning Poker Applications . . . . .	207
10.5.4	Extending Ethnographic Effort Estimation Study . . . . .	207
10.6	A Final Thought... . . . .	208
<b>A</b>	<b>Details of JOSSE Open Source Software Project</b>	<b>209</b>
<b>B</b>	<b>Detailed ML Results For Each Dataset</b>	<b>214</b>
<b>C</b>	<b>POST Crowd Planning Poker Survey</b>	<b>219</b>
	<b>Bibliography</b>	<b>226</b>

## List of Tables

4.1	Effort distribution among datasets. . . . .	51
4.2	Datasets summary. . . . .	51
4.3	JOSSE project statistics. . . . .	55
4.4	JOSSE summary. . . . .	56
4.5	JOSSE projects Quantification. . . . .	59
4.6	Statistics of Corpus Grammar Error Percentages. . . . .	64
4.7	JOSSE Evaluation Against Quality Taxonomy. . . . .	66
5.1	Effort Distribution in Experiment Datasets . . . . .	77
5.2	Summary details of datasets. . . . .	77
5.3	Time-Based Categories . . . . .	78
5.4	Results Summary . . . . .	82
5.5	Expert Performance . . . . .	85
6.1	Properties of software development tasks . . . . .	101
6.2	Summary of trials settings . . . . .	103
6.3	Summary of the five trials results . . . . .	105
6.4	Summary of the four crowd-size trials results . . . . .	105
6.5	Summary of the nine process-design trials results . . . . .	107
6.6	Breakdown of trial costs of the third experiment (process design). . . . .	110
7.1	Experiment Issues' Characteristics . . . . .	115
7.2	Crowd Behaviour Actions . . . . .	119
7.3	Hyperparameters optimal values . . . . .	123
7.4	Experiment Issues' Characteristics . . . . .	123

7.5	Experiment Issues' Characteristics . . . . .	129
7.6	Crowd Workers Submissions . . . . .	130
7.7	Crowd Workers Improvement Responses . . . . .	131
7.8	Assignment Quality VS Workers' Behaviour . . . . .	134
8.1	Comparison of estimation error metrics . . . . .	150
8.2	Summary of trial results . . . . .	151
8.3	Breakdown of trial costs. . . . .	153
9.1	Association of crowd action and UI events . . . . .	166
9.2	Number of assignments (cases) that are considered from each class . . . . .	167
9.3	statistical summary of crowd descriptors . . . . .	169
9.4	Questions of the crowd survey . . . . .	182
9.5	Results of crowd worker familiarity with CPP . . . . .	186
9.6	Results of unfamiliar crowd worker reaction . . . . .	187
9.7	Results of useful CPP components . . . . .	187
9.8	Results of crowd workers time allocation . . . . .	188
9.9	Results of good estimation rational . . . . .	188
9.10	Results of quality feedback reaction . . . . .	188
9.11	Results of information for revision . . . . .	189
A.1	An overview brief of each project included in the JOSSE dataset. . . . .	213
B.1	Performance of ML models and feature extraction methods . . . . .	218

# List of Figures

1.1	Effort Estimation Activities . . . . .	3
1.2	Scrum Flow Chart . . . . .	4
1.3	Thesis chapters and questions map . . . . .	10
2.1	Logical architecture of empirical methods . . . . .	14
3.1	Classification of Software Effort Estimation Methods . . . . .	27
3.2	Example of a Planning Poker Deck . . . . .	34
4.1	JIRA time logging screenshot. . . . .	54
4.2	JOSSE ER diagram. . . . .	54
4.3	Two box plots representing the dataset before and after the outlier removal.	61
4.4	Data dissension heat maps of JOSSE. . . . .	63
5.1	Simple Random Forest for Estimating Effort. . . . .	73
5.2	Fine-tuning BERT for SEE problem . . . . .	74
5.3	Dataset Refinement Process . . . . .	78
5.4	Feature Extraction Methods . . . . .	80
5.5	Box Plot of Performance Metrics . . . . .	83
6.1	General model of the crowdsourcing Planning Poker task. . . . .	93
6.2	Screenshot of AMT showing CPP HIT specifications . . . . .	95
6.3	Effort estimates available for crowd workers to select. . . . .	96
6.4	Asynchronous communication flow chart of two messages. . . . .	97
7.1	Proposed CPP Quality Model . . . . .	125
7.2	Low Quality Warning Message . . . . .	127

8.1	The Business Process Model and Notation (BPMN) of CPP . . . . .	140
8.2	Screenshot of AMT showing CPP HIT specifications. . . . .	141
8.3	HIT instructions as they are listed in the CPP application. . . . .	142
8.4	A screenshot show extra issue information and estimate options . . . . .	144
8.5	A screenshot shows low quality warning message as feedback . . . . .	145
8.6	A screenshot shows a list of CPP rounds . . . . .	146
8.7	List of software issues that are estimated by crowd . . . . .	148
9.1	State machines of crowd assignments . . . . .	164
9.2	State machines of a crowd worker . . . . .	165
9.3	Activity stream maps of accurate and inaccurate accepted assignment . . .	172
9.4	Activity stream maps of accurate and inaccurate Class C assignments . . .	176
9.5	Activity stream map for poor assignments (Class D). . . . .	179
9.6	Histogram of crowd familiarity with Planning Poker . . . . .	186
10.1	Activity stream map for poor assignments (Class D). . . . .	202



# Chapter 1

## Introduction

In software engineering, effort estimation is part of the planning phase in the system development life cycle (SDLC). The goal is to predict a reliable cost of achieving a given software development task such as implementing a new feature or fixing a bug. Effort estimates can be predicted using formal models, experts, or a combination of the two. This thesis investigates the feasibility of using human computation to predict expert-comparable effort estimates for a given software development task. This chapter introduces the thesis by giving a brief background and explaining the thesis motivation. It also lists the thesis contribution and describes the structure of the rest of the thesis.

### 1.1 Background

The thesis topic is an application of an application of Human Computation and Artificial Intelligence (AI) in Software Engineering. This section gives some background information to introduce these disciplines. It explains effort estimation in Agile software development and human computation using crowds. It also highlights the research methods used during the research.

#### 1.1.1 Software Effort Estimation

In software engineering, different development methods, such as Waterfall, Spiral, and Agile, can implement the SDLC differently. There are some software development phases common to those models, including planning, design, implementation, and testing. However, each model has its own way of arranging these phases and their activities. Software effort estimation processes vary based on the approach, the input and output, and the involved subjects. However, the goal in all cases is to predict a reliable effort estimate for a given task. Effort

can be presented in different units, such as time, cost, or even a points system, to determine project schedules and budgets.

Effort estimation approaches can be based on a formal model, an expert judgement, a machine learning model, or a mixture of these. Formal model approaches do not involve any human judgement in their processes, and they rely solely on mathematical equations. One popular formal method is the Constructive Cost Model (COCOMO) [1]. It has several versions: Basic, Intermediate, and Detailed. COCOMO depends on lines of code plus other constants that can be determined based on the project type: Organic, Semi-Detached, or Embedded.

Formal model estimation approaches are efficient and fast. Therefore, it is easy to use such approaches on a large scale. However, several studies have cast doubt on the reliability of formal model approaches [2]. Such models do not consider relevant contextual data that might provide insights into the estimation process. For instance, Putnam's model [3] does not differentiate between non-critical and critical systems that require additional testing and certification. In addition, such models do not permit critical analysis or negotiation of resulting estimates and thus lack justifications to accompany and support the outcome.

Expert-based methods rely on expert judgement using different estimation strategies such as Guesstimation [4], Wideband Delphi [1], and Planning Poker [5]. The methods share common activities, including topic understanding, searching for related information, and evaluating and discussing suggested estimates. According to the literature [2, 6, 7], a general model of the expert estimation process consists of the following activities and follows the flow chart illustrated in Figure 1.1.

- Understanding the topic: where estimators comprehend available materials about the software takes.
- Searching for related information: including similar software tasks from history to compare and understand the size of the current task.
- Quantifying calculation parameters: when estimators have understood the task, several features of the given task, such as the number of software modules that need to be considered, will be identified.
- Calculating an estimate: based on the identified calculation parameters, the estimate will be produced.
- Evaluating qualities of the estimate: including estimate accuracy and quality of the calculation parameters.
- Discussing estimates: to provide reasoning, critique, and feedback to the estimation process for the next software tasks.

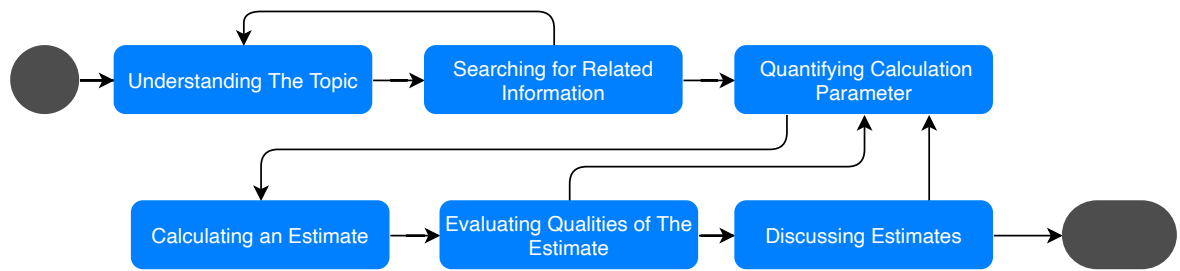


Figure 1.1: Effort Estimation Activities

Expert-based effort estimation approaches generally lack the scalability of formal models, but they can produce more reliable estimates accompanied by justifications [8]. Such methods encourage discussion of estimates within the group of estimators, which may improve the quality of the estimation end results. The practice is often associated with the Scrum software development method, where team members use negotiation and discussion to reach a consensus about their estimates, or determine that a task lacks sufficient detail to estimate accurately. Therefore, several aspects of a project can be considered, and a better understanding of a given task can be achieved.

Finally, Machine Learning (ML) based methods are typically trained on a large dataset to identify correlation between information about software development tasks and their actual cost, resulting in a data model that can be used to predict cost estimates for new software tasks. Different methods are proposed in the literature based on standard ML techniques, such as Artificial Neural Network [9], Random Forest [10] and Support Vector Machine [11]. ML methods are similar to formal models in the sense that they rely on machines to predict an estimate. However, ML methods are considerably more flexible than formal models, since the form of input data can be altered relatively easily to incorporate (for example) internal project discussions such as instant message chats and email as well as more conventional task descriptions and metrics. This enables researchers to search for optimal combinations of inputs to support reliable estimation.

### 1.1.2 Planning Poker

Scrum is an Agile software development method that arranges software development phases into small, iterative periods of time called sprints. Each sprint has planning, implementation, review, and retrospective phases, as illustrated in Figure 1.2. During the planning phase, which is the concern of this thesis, a Scrum team may undertake an effort estimation process called Planning Poker [8]. Usually, the team plays Planning Poker over a list of tasks, called the backlog, to estimate the effort for each backlog item.

Each member of the development team has a set of cards, each labelled with a possible cost

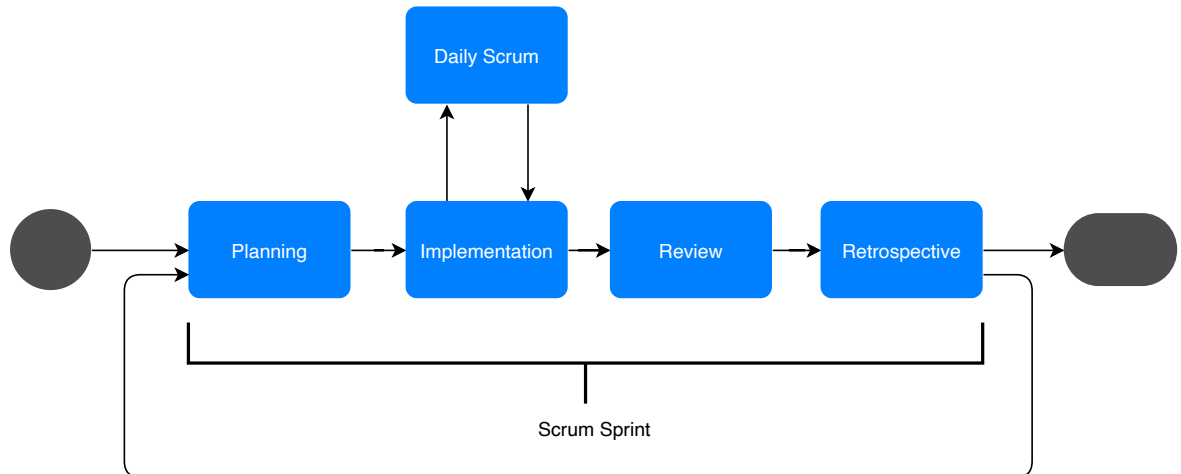


Figure 1.2: Scrum Flow Chart

estimate. Different units of cost can be used, including t-shirt size, person-hours, or story points. Cohn [5] proposes using story points as a means of comparing relative user story complexity, rather than producing an absolute estimate. In Cohn [5]’s approach, story point cards are labelled 0, 1, 2, 3, 5, 8, 13. A final card labelled with infinity can be used to signal that the task under consideration is too complex to be reliably estimated.

Team members start by estimating the effort for a task individually, and pick the card with the most appropriate value to make their estimate. Then, they reveal their cards simultaneously and compare their estimates. If there is no consensus for the estimate then the team members explain their views to each other. In particular, the estimators with the lowest and highest estimates are asked to explain their reasoning. Additional rounds of estimation and discussion are performed until they reach a consistent estimate for the task. Cohn [5] recommends that if consensus has not been reached after three rounds of estimation then the team should revisit the task separately.

### 1.1.3 Human Computation and Crowdsourcing

As von Ahn [12] stated in his dissertation, Human Computation is a paradigm for:

“... utilizing human processing power to solve problems that computers cannot yet solve.”

Software effort estimation is a candidate problem for human computation since current automated approaches still lack the reliability of expert-based estimates.

Initially, human computation was used to differentiate between machines and humans with the aim of avoiding unwanted processing requests such as those generated by unengaged

workers; see von Ahn et al. [13]’s research to develop CAPTCHA for more details. Later, it was used in information retrieval research to annotate and label images and other kinds of information [14].

Crowdsourcing platforms are markets that connect workers to requesters to meet their demands. These platforms have considerably eased the task of creating computational processes that combine human and automatic processing. A requester can be a human or a machine that has a specific task to be accomplished. What makes such platforms a shadow of human computation is the ability to interact with those platforms via Application Programming Interfaces (APIs). Thus, software powered by human computation techniques can post tasks for humans to accomplish and collect the outcome as part of the process.

The power of enabling human processing abilities to be manipulated by machines and included as part of regular machine processes has opened the door for different applications, including software engineering activities such as planning, development, and testing [15]. In particular, when crowdsourcing platforms emerged, human computation flowered in different disciplines.

Different kinds of crowdsourcing platform can be found, including micro-task and domain-specific platforms. Amazon Mechanical Turk (AMT)<sup>1</sup> is an example of a micro-task crowdsourcing platform, and TopCoder<sup>2</sup> is an example of a domain-specific crowdsourcing market that is specialised in software development tasks.

In AMT, a requester (employer) can post a job called a Human Intelligence Task (HIT). Then a worker (contractor) can review the HIT and either accept or reject it. Only workers with qualifications that meet the eligibility criteria can review, view, and work on the HIT. Once the HIT is accepted, the worker can submit the finished work, known as the assignment. After that, the assignment can be accepted or rejected by the requester according to the quality criteria. Assignment evaluation can be performed automatically. More advanced quality criteria checks may require manual approval of the assignment. Requesters have full control over the hiring process and it can be fully automated using the AMT API.

## 1.2 Motivation

Effort estimation plays a critical role in planning activities during the SDLC, providing the basis for developing project budgets, schedules and roadmaps. Numerous studies have reported on the difficulties of producing reliable software effort estimates for tasks, resulting in budget over-runs and/or the delivery of low quality software. For instance, Grimstad et al. [16] found that tight schedules give developers a reason for not assuring the quality of their

---

<sup>1</sup><https://www.mturk.com>

<sup>2</sup><https://www.topcoder.com>

code. Historically, software engineering projects have relied on estimation methods based on expert judgement and consensus formation, such as Delphi [17], as these are considered more reliable than approaches based on formal models [18, 5].

The limitations of such methods, in terms of scalability, are well recognised in the literature, particularly for large scale software development efforts. For example, Taff et al. [19] found 30 years ago that it takes up to two years to plan and estimate such systems, in which time the requirements and scope may very well have changed dramatically.

More recently, with the growing pervasiveness of Agile methods in software development [20], the Planning Poker estimation method has become increasingly popular [8]. Planning Poker resembles Delphi, in that estimates are produced through structured consensus formation within a group of domain experts. However, the practice is designed to either produce an estimate for a task relatively quickly (Schwaber [20] recommends doing so within a few minutes), or to identify a task as requiring further elaboration before an accurate estimate can be produced.

However, this more Agile approach to estimation still lacks scalability for very large software projects. For example, in the context of open-source software, in 2019, the Linux Kernel, Firefox Web Browser, and JBoss projects had, respectively, 6456, 11751, and 17032 new issues awaiting triage [21, 22, 23]. Applying labour-intensive expert estimation methods to such projects is infeasible, due to the scarcity of human resource available. Simply processing the existing backlog of issues would consume all the working time of the software engineers working on the project for many months.

A separate risk of expert-based methods is that of bias. For example, the valence effect [24] is the human tendency to be optimistic, resulting in underestimating the effort required for a given issue. Alternatively, estimators may experience a conflict of interest, such as when the estimators are the same people who will be undertaking the estimated effort. Moharreri et al. [25] reported conflicts of interest in a Planning Poker session resulting in a complicated planning phase and unreliable outcomes. Several studies [26] reported that estimation sessions with outsiders resulted in more accurate estimates than closed estimate sessions. Outsiders are estimators outside the development team.

An alternative approach to effort estimation, addressing the issue of scalability, has been to depend on automated techniques such as formal models or machine learning. Such methods are able to generate estimates rapidly based on inputs, since all processing is undertaken automatically. Unfortunately, in the current state of the art, automated approaches are generally outperformed by expert estimation. Jørgensen et al. [2], in a survey of 16 effort estimation studies, found that in 10, using expert-based effort estimation resulted in more accurate estimates. One limitation of automated methods is the relatively narrow scope of data considered when producing estimates, compared with that considered by experts. Further, automated

methods lack techniques for identifying hidden assumptions, whereas expert-based methods employ negotiation and discussion amongst the experts for this purpose. Therefore, human participation remains critical to reliable software effort estimation [2], yet, human participation introduces scalability and cost issues for large-scale projects.

Reviewing the literature, although there are numerous studies exploring the potential for automated effort estimation, few studies investigated expert-based methods, and even fewer worked on enhancing those methods. Of the effort estimation studies reviewed by Jørgensen et al. [2], only 15% investigated expert-based methods. In addition, Jørgensen et al. reported that the majority of those studies did not suggest any improvements to the methods studied. Therefore, this gap represents an opportunity that needs to be addressed, especially given that expert-based estimation methods are the dominant kind of method, as reported by Jørgensen et al. [2].

In this thesis, the question of whether, by automating or semi-automating an expert-based effort estimation process such as Planning Poker, limitations on scalability and cost and the risk of bias can be addressed. Recent advances in human computation have revealed opportunities to overcome some of those challenges. Human computation is a human-centric processing approach, where machines orchestrate humans to solve a problem that is hard for a machine to tackle. This approach is usually used in tasks where machines can not process reliable enough outcomes, for example, estimating the effort of a software development task.

The development of crowdsourcing platforms offers easy and cheap access to hundreds of people to process different kinds of tasks. For instance, AMT offers a platform to process micro-tasks such as annotation tasks. Thus, this thesis investigates the possibility of automating Planning Poker using human computation, and whether the resulting estimates compare with expert estimates.

## 1.3 Thesis Statement

The statement of this thesis is that **expert-comparable estimates for software development tasks can be efficiently predicted by playing Planning Poker in crowds using machine learning as an assistive management model.**

In particular, the thesis addresses the challenges of finding an appropriate estimation process for crowds, selecting crowd workers for their tasks, and, most critically, automating the quality assessment and management of crowd worker tasks.

Planning Poker is an effective method to aggregate and consolidate expert opinions, but it needs a manual procedure and domain experts. This thesis contends that crowd wisdom can replace experience if a large number of people is involved and they are suitably orches-

trated. A crowdsourcing platform can provide economical access to the crowd without time or location restrictions.

Nonetheless, a large number of crowd workers is hard to manage manually. Workers usually take micro-level tasks, and the quality of their outcomes is questionable. Human computation techniques can automatically coordinate and process the crowd work, automating handover between rounds of Planning Poker task without the need to divide it into micro-level tasks. Crowd handover also helps in identifying and transferring important knowledge between crowd workers. Finally, automatic assessment of crowd worker submissions and feedback to workers enable estimate quality to be managed in real time.

Given the thesis statement above, several questions need to be addressed. As a first step, based on a review of the available literature, a decision was made to investigate whether contemporary advances in natural language processing (NLP) techniques are a viable means of obtaining effort estimates at scale. It is also important to have a common ground for all effort estimation methods that are involved in this thesis, and thus, this thesis will first assess and measure the performance of ML models, since ML models are the most efficient way of predicting estimates. Therefore, the first question that needs to be addressed is: **RQ1: Can the latest generation of NLP techniques produce an expert-comparable effort estimation for a software development task?**

Since this thesis is the first to investigate playing Planning Poker using human computation, the next step is to assess the feasibility of human computation performance in predicting estimates for the same software tasks that are used in the ML-based SEE method. More detailed information is required, including the size of the crowd, the task handling process, the amount of information about the software tasks, and finally the appropriate complete process of Planning Poker that can be managed by a machine. This information is essential to designing an appropriate model for playing such a game, and thus, the second question that needs to be addressed is: **RQ2: Can a machine-based orchestration of a crowd be used to predict software effort estimates by playing Planning Poker?**

After investigating the feasibility of playing Planning Poker using human computation and having an appropriate design for the game, the next step is to address a well-known challenge of running a crowd task: crowd assignment quality. For that, the third question is: **RQ3: Can the quality of crowd-worker software estimates generated during crowd Planning Poker be automatically measured and enhanced?**

After assuring crowd assignment quality, it is time to address the main question of the thesis, which is: **RQ4: Is playing Planning Poker using human computation in a crowdsourcing platform producing an effort estimation comparable to that of an expert for a software development task?**

Finally, to improve the proposed estimation method, it is important to understand how crowd



workers play the game and who is most appropriate for it. This thesis includes several experiments which generate a wealth of data about crowds playing Planning Poker, and thus, it can be used to gain further details. In those field experiments, real software issues are used as objects, and AMT workers are the estimators. As part of the experiment, each issue was annotated with the estimated and actual effort in person-hours. The issues were collected from different open-source projects such as Apache HTTP Server. An ethnography study was conducted to observe the crowd behaviour over a few months while the estimates were being produced. The goal of the ethnography study was to develop all possible personas of crowd workers who participated in the experiments.

The study provides an understanding of crowd worker behaviour when performing effort estimation for software development tasks, and thus, it results in fine-grained insights for future work to improve the quality of crowd estimation. For instance, by knowing what kind of information crowd workers seek, we can either provide that kind of information or direct them to other sources. The thesis will use the data to draw different personas of crowd estimators and explain their behaviour. To do so, the following question needs to be addressed: **RQ5: What are the possible crowd estimator personas and their behaviour?**

Figure 1.3 maps the questions to their corresponding chapters. RQ1 was addressed by two chapters and the rest of the questions were addressed by a chapter for each one. Each chapter divides those questions into smaller research questions that can be addressed by the selected research methods.

## 1.4 Contribution

This thesis contributes to the bodies of research in software engineering and human computation. Six contributions that are detailed below narrow the gap in enhancement of expert-based methods. These contributions aim to help pave the way to better automation options in a human-intensive discipline such as software engineering.

**Demonstration of feasibility of playing Planning Poker using human computation.** The thesis is the first to employ human computation using crowd workers to produce Planning Poker estimates of software development tasks. The work provides the first insights that such an organisation of crowd workers can deliver, and demonstrates that crowds can produce estimates that are of comparable accuracy to those of project experts.

**Demonstration of feasibility of a handover-based method instead of task division in crowd micro-task platforms.** The thesis suggests a different task delivery method to crowd workers in micro-task crowdsourcing platforms. Instead of dividing the

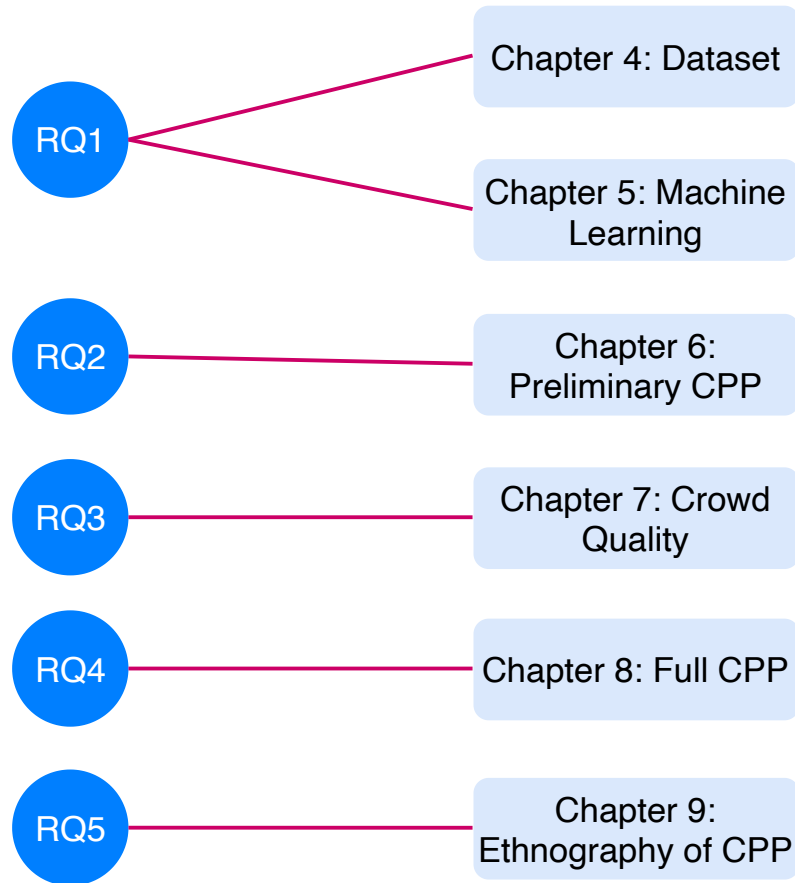


Figure 1.3: Thesis chapters and questions map

task of estimation using Planning Poker into micro-level tasks, an asynchronous handover process between crowd workers was implemented to coordinate and deliver the lengthy estimation process without division. This could be extended to other kinds of large tasks that are hard to divide into smaller independent tasks.

**Automation of crowd assignment quality assessment for software effort estimation.** The thesis augments the quality assessment of crowd work using machine learning. In fact, it reaches a fully automated quality assessment model that can assess the crowd work on the fly. The approach starts with a training phase to collect training samples and assess them manually. Then the ML assessment model is produced based on those manual assessments.

**Improving crowd effort estimation assignment quality using the behavioural economic theory of loss attention.** The theory states that humans are willing to pay additional attention to tasks that involve losses [27]. This thesis is the first to use loss attention [27] to motivate crowd workers to enhance their assignments in Crowd Planning Poker. After reading feedback that is designed according to the loss attention theory and generated

from the quality assessment classifier, crowd workers get a chance to enhance their assignment. The thesis also reports in detail on how significantly the quality of the crowd outcome is enhanced.

**Identification of crowd estimator personas.** An ethnography study was conducted to understand the behaviour of the crowd estimators. It developed different personas to represent all the behaviours of participating crowd workers. The goal is to find a better way to enhance the proposal of this thesis in future studies.

**Publishing data about software issue costs and crowd effort estimation behaviour.** The data sets collated for this project have been published for reuse<sup>3</sup>, including thousands of crowd behaviour records and hundreds of issues that are annotated with actual and estimated effort. This step aims to enable research replication and facilitate future studies that aim to enhance expert-based estimation methods.

## 1.5 Thesis Outline

The rest of this thesis is structured as follows.

Chapter 2 presents the thesis methodology. It explains the overall methodology and details the research methods used, including literature review replication, laboratory experimentation, and ethnography methods.

Chapter 3 is the literature review. It provides an overview of previous work on effort estimation and especially Planning Poker. In addition, related human computation and crowdsourcing research is highlighted and compared to the proposals in this thesis.

Chapter 4 describes the JIRA Open Source Software Effort (JOSSE) dataset. It starts with a background of publicly available SEE datasets. Then, it explains how the JOSSE dataset was collected and processed. It also suggests additional refinement options and their implications for the dataset.

Chapter 5 presents an evaluation of using state-of-the-art NLP to perform software effort estimation. It reports the early experiments of this research to use machine learning in predicting the effort required for a software development task. Then, it shows how the experimental results fall below the expected reliability threshold when compared with expert effort estimation. In addition, it illustrates the state-of-the-art research about using machine learning in effort estimation.

---

<sup>3</sup><https://github.com/crowd-planning-poker>

Chapter 6 discusses evaluating different settings for playing Planning Poker with crowd workers in a crowdsourcing environment, with settings such as the number of crowd workers per round and the amount of information needed to be passed to the workers. In this chapter, the concept of Crowd Planning Poker is assessed in terms of its feasibility.

Chapter 7 explains the quality assessment model of CPP, including how it manages the crowd outcome by assessing the outcome quality automatically and how machine learning is involved in classifying the crowd answers. This chapter also describes crowd behaviour and how that could be used to enhance assignment quality. It then describes the approach taken to enhancing the quality of crowd outcomes in CPP. It illustrates how the CPP quality model is used to enhance the crowd assignment by employing a behavioural economics theory (loss attention). It also shows different crowd responses and how such an implementation can enhance the quality dramatically while reducing the undesirable quality outcomes.

Chapter 8 chapter uses human computation to play Planning Poker. It details the primary contribution of the thesis, Crowd Planning Poker (CPP). It walks through the process of CPP and the experiments that have been conducted to show how CPP can produce an effort estimate comparable to that produced by an expert. It also explains how the crowd can add value to the development task. This chapter assesses Crowd Planning Poker in terms of its reliability and efficacy.

Chapter 9 is about personas of crowd estimators. After asking a large number of crowd workers to estimate several software development tasks, the ethnography study comes up with different personas that describe the behaviours of crowd workers as human estimators. In this chapter, the study analyses the personas, and the workers are surveyed to confirm the findings.

Chapter 10 is the thesis conclusion. It sums up all the chapter conclusions and highlights future work, namely, implementing CPP in a real-world case by asking a potential development team to use CPP as their estimation method and then measuring how successful CPP can be.

## Chapter 2

# Research Methodology

As stated in Chapter 1, this thesis is looking to test the feasibility of human computation as a method to improve the scalability and efficiency of expert-based software effort estimation. Therefore, the thesis follows the empirical research method described by Basili [28]. It proposes the estimation process (Crowd Planning Poker), selects the research methods, experiments with the proposed process, measures and analyses, and then validates the research hypothesis. In the following sections, extra details will follow about software engineering empirical research, the rationale behind the selected research methods, and an explanation of each method that is used in this thesis.

## 2.1 Software Engineering Empirical Research

Basili [29] explains that empirical research methods in software engineering involve an experimentation component that can provide evidence to explain a given observation or support a proposed model. This includes laboratory controlled experiments, surveys, and case studies.

Sjøberg et al. [30], provide further detail on software engineering empirical methods. At the highest level, there are primary methods that collect and analyse original data through laboratory experiments, surveys, case studies, action research, and ethnographic studies. On the other hand, secondary methods use results from previous empirical studies for reanalysis and/or synthesis. These methods comprise literature reviews, replications, and statistical meta-analysis.

Data gathered within empirical software engineering research can be qualitative or quantitative. Quantitative results can be evaluated statistically to ensure internal validity. Qualitative results can be explained by association and observation. It is quite common to use both

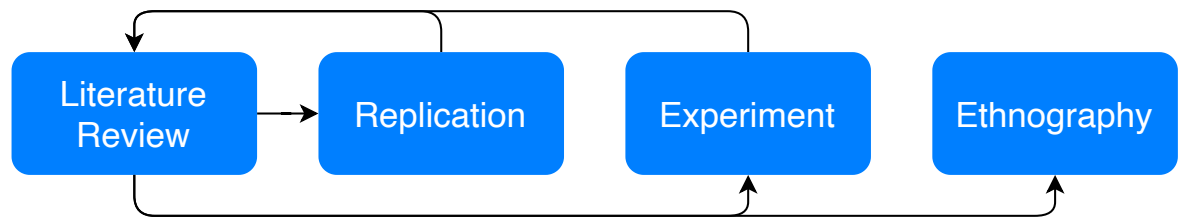


Figure 2.1: Logical architecture that interconnects empirical methods selected for the thesis.

methods to investigate different kind of information. Quantitative methods answer *what* questions, whereas qualitative methods are often better for finding answers to *why* questions. For the purpose of scientifically investigating and researching the thesis, four empirical research methods have been selected. They are literature review, replication, laboratory experiment, and ethnography. Figure 2.1 illustrates the connections between the four methods.

The first stage of the research is to conduct a literature review with the goal of understanding the current state of the art, and keep iterating over the literature review throughout the research life cycle, see Figure 2.1. Replication is then used to synthesise critical findings in the literature review about contemporary advancement in NLP and ML. Then, a series of laboratory experiments are used to validate and evaluate the thesis hypothesis of using human computation to produce software effort estimates. Finally, ethnography is selected to give fine-grained insights and rationales behind the outcomes of CPP experimental work.

Figure 2.1 shows the flow between the selected research methods used during the research of this thesis. The results of the literature review direct the design of the replication, experimentation, and ethnography studies. Then, the results of replication and experimentation are used to further narrow the literature review towards the research gap. These loops resulted in a couple of replication and experimentation trials in this thesis, each trial building on the previous findings. At the end, the ethnography study is used once to provide finer details about the experimentation work.

The rest of this chapter explores each of these methods in further detail in terms of their application to the research project. Section 2.2 describes the literature review method applied for elaborating the state of the art in ML, NLP, software effort estimation and human computation. Section 2.3 details replication methods applied to ML in software effort estimation research. Section 2.4 shows the laboratory experiments conducted to further investigate the application of human computation and crowdsourcing to augment expert-based software effort estimation. Section 2.5 explains the ethnography methods used to investigate fine-grained information about crowd behaviour as a way to provide a rationale and future insights. A summary section will conclude this chapter by providing a synopsis of its content.

## 2.2 Literature review

Literature review is a secondary empirical method, as stated in Sjøberg et al. [30]’s work. It does not collect or analyse original data. Instead, it reviews previous research work in the same area of the thesis topic. Its primary goal is to identify and analyse *existing* evidence [31]. From such a literature enquiry, a researcher can establish a foundation about the research topic, adjust research direction to close literature gaps, confirm speculations, and take informed decisions about the proper research tools and methods. For this thesis, the literature review is used to uncover the current state of the art in software effort estimation using experts, machine learning and human computation. As a result, it directed the research towards the proposed CPP model to narrow the research gap illustrated in Chapter 1. Moreover, it participates in positioning the new evidence produced by the thesis among that which already existed in the literature body of effort estimation and human computation. Additionally, it offers an opportunity to confirm existing evidence of using machine learning (ML) in effort estimation using the replication method.

The main body of the thesis literature review (Chapter 3) was primarily derived from academic publications, including peer-reviewed conference papers, journal papers, and text books. As a secondary source, industry white papers, blogs, and magazine articles are also considered, to provide a wider foundation that helps in understanding the two disciplines, software engineering and human computation, from both academic and industry perspectives. Google Scholar, IEEE Xplore, and ACM Digital Library were the main search engines used to source the academic publications, and Google Search was also used to find relevant industrial publications.

The foundational literature for effort estimation and human computation is detailed in Chapter 3. It starts with effort estimation as a software engineering activity, and then explains estimation methods, measurements, and environments. Moreover, a general taxonomy for software effort estimation is proposed to systematically consider all the relevant content in the literature about software effort estimation. Most of the effort estimation concepts are introduced and discussed with an expatiation of their relevancy to the thesis. The remainder of the chapter is dedicated to establishing the foundational literature for human computation and its ties with software engineering in general. Human computation related concepts such as crowdsourcing are also covered in the chapter.

While Chapter 3 is the main body of the literature review, additional literature review sections are also included to address specific topics. For example, Chapter 5 discusses the use of machine learning algorithms in software estimation. Similarly, Chapter 9 has a dedicated section discussing the literature of ethnography and its role as an empirical research method in software engineering.

## 2.3 Replication

Literature review as a method helped construct the theoretical part of the thesis and consolidating the researcher understanding of the relevant topics. In addition, it was the primary method used to identify the required evidence in the literature to support the thesis claims. However, topics like applying machine learning algorithmics are better accompanied by practical experience to realise the complicated theory behind them, and to confirm what is reported in the literature. Thus, replication as another empirical method is used as a practical extension of the review.

Replication is also considered a secondary empirical method since it relies on the original, and replicated studies. While it is mainly used for confirmation, as stated by Brooks et al. [32], it also plays a significant role in acquiring the knowledge during the learning process [33, 34]. Along with its confirmatory power, replication can be used as a practical side of learning and the literature review can be theoretical side. The researcher conducted a non-exact replication [34] experimenting with ML for effort estimation to be highly confident in the reported ML performance. He also used the replication to deepen his understanding of ML technique, as it is an essential part of the CPP model.

As mentioned by more than one study consolidated in Trendowicz and Jeffery [24]’s book, machine learning (ML) effort estimation is complex and requires a rigorous understanding of its extensive fine details. Thus, replication as an empirical research method was used to do a non-exact replication [34] of the most promising machine learning algorithms. Beside the learning outcomes of using replication, it was critical to have a high degree of confidence in the reported results of ML methods for effort estimation and before using human computation. Thus, the replication of ML methods was mainly to confirm the literature findings and allow the researcher to confidently adjust the research direction and the proposed model.

Replication is used in Chapter 5 where two ML algorithms are selected based on their reported performance in the literature, including an ensemble ML algorithm, Random Forest, and an Artificial Neural Network, Multilayer Perceptron (MLP). The datasets, also derived from the literature, include Deep-SE, Porru, Desharnias, and NASA93. Before replicating the ML experiment, Chapter 4 also introduces a new dataset called the JIRA Open Source Software Effort (JOSSE) dataset. That was a response to the lack of availability in the literature of a dataset that is recent and contains a textual feature, i.e. software task description.

Since the replication follows a non-exact replication model [34], several additional aspects were added after conducting the original experiments, including a new factorisation method using BERT [35] and the new proposed dataset, JOSSE. Such additions give more confidence in using the option of human computation as originally planned, because it was demonstrated that the machine learning based methods tested did not produce effort estimates that were as



reliable as experts.

## 2.4 Empirical Experiments

After acquiring theoretical and practical knowledge, experimentation is used to manipulate independent variables in different experiments to validate the proposed CPP model. Generally, experiments are designed to examine a causality relationship between independent and dependent variables in a given hypothesis [36]. Thus, to design, conduct, and then report an experiment, it is essential to have clear hypotheses. Experimental work in the thesis is a core component that provides required evidence to support the thesis claims. It examines a variety of causality relationships, starting with the impact on the estimate reliability of introducing the crowd as an estimator, and ending with finer details such as the impact of different kinds of information on crowd performance. Since this thesis is the first to introduce CPP, there are several aspects that need to be investigated, such as the size of crowd estimators and the role of information in their estimates. Pilot studies were used to examine these aspects in a relatively short time.

Before conducting an experiment, a thorough experimental design is essential in conducting the experiment and reporting it. During the design, experiment *hypotheses* need to be crystallised. A hypothesis is an idea that assumes an existing relationship between at least two variables. There are two fundamental types of hypotheses: null and alternative hypotheses. Null is the negation of the relationship that is going to be proved by the experiment. The rationale behind hypothesis formation is to draw a conclusion that can be statistically validated [37].

What distinguishes experimentation from other empirical methods is the controlled manipulation of the variables. The variables are formed into a testable hypothesis, through which an investigator can first identify a causality relationship, if there is one, and the process where such a relationship accrues.

Every experiment has two kinds of variables: *independent* and *dependent* variables. The focus is on one or a few dependent variables that are affected by manipulating the independent variables during the experiment. Different values that an independent variable can take during an experiment are called *treatments*. Moreover, the experimental design identifies where the experiment is conducted (also called *treatments*), who will apply the experimental process (*subjects*), and on what the experiment will be carried (*objects*). The next sub-sections outline the experiments conducted in this thesis, summarising the variables, treatments and artefacts.

### 2.4.1 Pilot Experimentation

Wohlin et al. [37] states that experimentation is not a simple process, especially when it includes humans as subjects or objects. Thus, creating a pilot version of the experiment is a plausible option when uncertainty is high or in the case of a totally new experiment that has not been reported earlier in the literature. Surprisingly, Glass [38]’s observation about the scarcity of pilot studies in the software engineering literature is still valid even after twenty-three years. There is very little research [38, 39] about using a *pilot* version of an experiment. On the other hand, the word “pilot” is heavily used in the literature to mostly indicate using different experimental subjects/objects who are easily accessible, such as students [40, 41]. However, other disciplines, including medical [42] and social [43] have a detailed body of literature about using pilot experiments and studies.

According to Glass [38], there are three kinds of pilot studies: rigorous, moderate, and informal. They can be distinguished based on their difficulty. For example, informal requires have neither a statistical approach nor confidence factors, therefore the pilot is about experiment complexity. Moreover, van Teijlingen and Hundley [43] define a pilot study as a “mini of a full-scale study”, and thus the pilot is about experiment size. Others such as Connelly [44] also suggest that a pilot experiment is about 10% of the full version.

The key point is that the pilot version of an experiment is as same as the full version, however, it is simpler and smaller. Therefore, pilot studies can be used as indicators of feasibility or testing of setting. However, their simpler design and smaller size prevent drawing final conclusions.

The importance of pilot studies comes from risk reduction, especially in a new, unexplored area. A risk can be a financial risk or an ethical violation when an experiment includes human subjects or objects. Another factor that points to pilot experimentation is the agility of such experiments to quickly explore the feasibility of different design options [44]. Therefore, the pilot experiments of this thesis were designed to investigate human computation application as treatment to SEE and how to handle crowd quality efficiently. The dependent variable was effort estimation, and the independent variables were task information, process design, and crowd size. The experiment artefacts used are software tasks collected from issue tracker systems for open-source projects as experiment objects and AMT crowd workers as its subjects.

The thesis conducted a total of five pilot experiments. The pilot experiments are designed with a hierarchical architecture (parent–child relationships). There were two parent pilot experiments: CPP feasibility and quality management. The CPP feasibility pilot experiment has two children: one investigating the amount of information that best serves the crowd, and one investigating the crowd size that is suitable for a CPP estimation task. The pilot

experiment (CPP feasibility) is designed to be an indicator of the feasibility of playing CPP in a crowdsourcing environment. Chapter 6 offers extra details.

The quality management pilot experiment led to a further pilot, which investigates the possibility of delegating the quality classification of crowd submission to an ML classifier. The parent pilot experiment is to investigate the crowd response to the newly developed quality model that automatically classifies and enhances their submission. The first pilot study (CPP feasibility), uncovered the quality problem and identify its size. Thus, the second parent pilot experiment was designed after the results from the first pilot study were known. Chapter 7 explains the execution of these experiments.

### 2.4.2 Laboratory Experiments

After assessing the feasibility of playing Planning Poker using human computation and identifying the proper design and configuration using pilot experiments, the way is cleared for a full-scale laboratory experiment to examine the thesis claim. The final (sixth) experiment is designed on a much larger scale (about 66% bigger). It also includes the statistical analysis of the hypothesis. It incorporates the full design of running the experiment autonomously using human computation and avoiding any manual administration work as experimental treatment. The main goal is to examine whether replacing the experts (subjects) with crowd workers in playing Planning Poker will dramatically impact the estimate's reliability or not. More details about the experiment and its results are explained in Chapter 8.

All the experiments used real artefacts, specifically, the objects (software issues) collected from open-source issue tracker systems. The subjects in all experiments were crowd workers hired from AMT, and AMT was used as a micro-task crowdsourcing environment for the pilot and full-scale experiments.

Since CPP experiments include humans (the subjects of the experiments), an ethical approval from College of Science & Engineering Ethics Committee [45] in the University of Glasgow was obtained. In addition, a scheme of “soft rejection” was adopted to examine every crowd submission rejected by the auto classifier. If the rejected assignment is reasonably good, the crowd worker gets paid but the rejected assignment stays rejected and is not included in the experiment results.

## 2.5 Ethnography

The accumulated behavioural data (UI trace log) from the experimental work represented an opportunity to understand *how* crowd workers develop their estimates, but a framework is needed to analyse the data and infer conclusions. Therefore, ethnography is selected as an

empirical method to investigate and understand behaviour. According to Easterbrook et al. [36], ethnography in software engineering can play an important role in understanding the technical community practices and communication within such a community. Ethnography provides a research framework and analysis tools to investigate the behavioural data, and provide an understanding of how the results from the CPP experiments were achieved. While the experiments can answer the *what* question, ethnography can explain *why* it did or did not work. In addition, the ethnography also provides confirmation on the results of the pilot experiments of in this thesis.

Traditional ethnographic research takes a prolonged time and requires the presence of the researcher in the research setting. However, Digital Ethnography [46] has emerged as means of studying a research setting without the researcher being physically present. The traditional concept of collecting and observing the research group still remains, however, new tools and techniques are also used to help in carrying out such studies. For example, studying a virtual community, e.g. a micro-crowdsourcing platform, may become infeasible since there is no single location that can be targeted for visiting and observing. However, virtual observation through recording of user behaviour provides an alternative data gathering method.

As explained earlier, ethnography is an effective empirical research method that helps in understanding a given community. Such a study answers the ‘why’ and ‘how’ questions. Unlike experimentation, ethnography investigates real instances of the proposed model, and thus, the researcher can analyse the variety of the instances.

Since the observation has not been done directly, the outcomes of such an ethnographic study may need an additional tool to help in confirming such findings. In this case, surveys are usually selected for the last confirmation and validation step of digital ethnographic studies. They have been done in several software engineering studies, such as Kim et al. [47]’s work.

The experimental work of the thesis results in a large corpus of behavioural data since crowd workers were monitored across all the experiments. The crowd behaviour was reflected in their interaction with the User Interface (UI) of the software that need to work on to accomplish the estimation task. All the monitored interaction was stored, including mouse movement, typing, scrolling, clicking, etc.

The UI logs were transformed to an eXtensible Event Stream (XES) to be mined and analysed according to process mining techniques. Process mining software ProM and DISCO was used to analyse the UI logs. Several findings and speculations about the crowd behaviour were identified.

Finally, a survey consisting of nine questions was dispatched to workers who participated in the CPP experiments to validate the finding of the ethnography study. The goal of the survey was to find out more about the crowd behaviour in the areas that can not be explained clearly by the UI log. The survey design and the exact targeted crowd group along with the results

can be found in detail in Chapter 9.

## 2.6 Summary

A variety of empirical research methods were selected to investigate the thesis and its claims, following an empirical strategy of starting with a theoretical review, then a practical review, claim validation, and in-depth understanding of the results.

The theoretical review started with a literature review of the thesis topic. Two types of review were conducted: a general review covering the two disciplines of the thesis (effort estimation and human computation), and a more specific targeted review for relevant topics such as machine learning. Therefore, replication as a method was used to undertake a practical review of machine learning and natural language processing (NLP) topics relevant to the proposed CPP model.

A practical review was conducted by replicating machine learning experiments in the area of effort estimation. The main goal was to gain extra confirmation of the reported results in the literature. In addition, a side benefit is that performing the research enhances understanding of the complex machine learning algorithms and deep learning techniques. Replication guided the research to propose a new dataset and try a recently proposed factorisation method, BERT. It also confirmed the literature findings and opened the door confidently for the thesis experimentation.

Experimentation as another empirical method was used to validate the thesis claims. A sum of six experiments were conducted. The experimental work follows the design–pilot–conduct strategy where a pilot version of an experiment is applied before conducting a full-scale experiment. This helped to reveal unexpected issues in the experimental design and participated in enhancing the proposed CPP model. The first three experiments investigated the level of information, crowd size, and feasibility of the CPP design. Then the fourth and the fifth experiments explored the quality management model. The sixth experiment was to validate the thesis claim in a larger scale using a full implementation of the proposed CPP model. While the experiments provided an overall answer, a more rigorous and detailed view needs a lengthy observational study and analysis. Therefore, an ethnography study was designed to understand the crowd behavioural data.

Ethnographics provide detailed insights about the sociology of the research group. They can also explain why and how real instances vary from a given model. However, this method requires the collection of data over a prolonged period of time and of as many details as possible. Traditional ethnography requires the presence of the researcher and direct observation of the targeted research group. However, recently a digital form of ethnography has emerged to accommodate the overwhelming technological and digital society that has taken

over human life. While traditional concepts of ethnography are still valid, its tools have been adjusted to accommodate the technological change. This thesis used the collected UI interaction log of the crowd converted into event streams for analysis. The findings were validated and confirmed by the crowd workers using a survey dispatched to them at a later stage.

## Chapter 3

# Literature Review

The previous chapter explained how the research methods were selected and provided a rationale for each one of the selected research methods. This chapter presents a literature review of the research areas related to the main topics of this thesis: expert based and automated software effort estimation (SEE) methods and human computation.

The next section starts with definitions of SEE and related concepts and then details different SEE contexts. Then Section 3.2 covers expert-based SEE methods to explore their activities and processes. The goal is to have a comprehensive understanding of the existing methods. After that, Section 3.3 considers machine-learning SEE methods. Different models and algorithms of machine learning that are used in SEE are explained and reviewed. Then, Section 3.4 highlights a literature paradox along with a rationalisation of industry and academic preferences towards SEE methods. After, Section 3.5 illustrates different human computation definitions and concepts, including neighbouring research areas such as social computing and crowdsourcing. In addition, it explains the quality challenge of human computation outcomes. Then Section 3.6 explores different software engineering applications using human computation, and highlights the research attempts closest to the thesis proposal. Finally, the last section summarises this chapter and states the next direction of the thesis.

### 3.1 Characterising Software Effort Estimation

Software effort estimation (SEE) refers to the process of predicting how much effort a given software development activity may take. Although this may sound simple, the literature suggests otherwise. Most of the studies published about SEE never define the wider planning activity within which SEE may be carried out. In a literature review of the topic, Grimstad et al. [16] found that only 10% of the reviewed literature (one book and two research

papers) defined SEE precisely, suggesting much of the research is carried out within an uncertain scope. Given this apparent uncertainty and in order to provide a framework for the literature review, this Section first reviews definitions of effort estimation in the literature, then reviews the characterisation of software effort estimation in different contexts, before reviewing different approaches to classifying effort estimation techniques.

### 3.1.1 Definitions

The System and Software Engineering International Standard (SSEIS) [48] does not yet have a definition for SEE, but it has a definition for each word in SEE (*Software*, *Effort*, *Estimate*) as follows.

“*Software*: 1. computer programs, procedures, and possibly associated documentation and data pertaining to the operation of a computer system; 2. all or part of the programs, procedures, rules, and associated documentation of an information processing system; 3. program or set of programs used to run a computer.”

“*Effort*: the number of labor units required to complete a schedule activity or work breakdown structure component, often expressed in hours, days or weeks.”

“*Estimate*: a quantitative assessment of the likely amount or outcome. Usually applied to project costs, resources, effort, and durations and is usually preceded by a modifier (i.e., preliminary, conceptual, feasibility, order-of-magnitude, definitive). It should always include some indication of accuracy (e. g., (+ or  $-$ )  $\times$  percent).”

DeMarco [49] defines *Estimate* in his classic book as:

“An estimate is a prediction that is equally likely to be above and below the actual result.”

Software Engineering Body of Knowledge (SWEBOK) [50] defines *Estimate* as:

“An estimate is a well-founded evaluation of resources and time that will be needed to achieve stated goals.”

Then, SWEBOK describes *Effort estimation* as:



“The estimated range of effort required for a project, or parts of a project, can be determined using a calibrated estimation model based on historical size and effort data (when available) and other relevant methods such as expert judgment and analogy... A software estimate is used to determine whether the project goals can be achieved within the constraints on schedule, budget, features, and quality attribute”

While the definitions above agree that effort estimation is a prediction activity, they diverge on the specification of the measurement as qualitative (high, low) or quantitative (number of labor unites), source knowledge as historical data or experience, and the process as a well-founded evaluation or a calibrated model.

As can be seen from the definitions, there is both some uniformity and divergence. All the definitions agree that an estimation is a prediction of the cost of a software development activity. There is also agreement between the SSEIS and the SWEBOK that estimates are for the purpose of establishing the feasibility of a software project, although purpose is not mentioned by DeMarco.

However, the definitions vary according to the exact nature of the prediction. The SWE-BOK proposes a range of quantitative measures for estimate, whilst the SSEIS identifies units of labour as the means of measuring the cost of a work item (implicitly quantitatively). DeMarco [49] makes no reference to estimation units and indeed, the contemporary Planning Poker method uses categorical rather than quantitative estimation units (Story Points). Further, whilst the SSEIS proposes that estimates should be accompanied by a likely error margin, whilst DeMarco [49] asserts that estimates should have an equally distributed error margin above and below the central value.

Therefore, a better way to build up a foundation for understanding SEE is to look at SEE as a collection of concepts that have different terminologies in different contexts at different scales of software development and project effort.

### 3.1.2 Different Perspectives of Software Effort Estimation

Fundamentally, software effort estimation provides inputs to project planning processes at different scales of granularity. Historically, software projects that followed a waterfall like methodology would adopt a top-down approach to system design and then a bottom-up approach to estimation, allowing granular estimates of individual module costs to inform high level timelines and budgets [51]. For example, Armour [52] explains how estimates are used to prepare a commitment plan for external stakeholders, whilst internally, estimates are used to develop a working plan. Thus estimates are used to develop a project *budget* for internal

consumption [24, 48, 50] and a *bid* that allows for the risk of overruns and anticipated profit [24].

However, numerous authors argue that the process of developing budgets based on effort estimates is challenging in the context of software projects. [51] makes similar arguments to Jr. [53] that software is intangible and thus prone to ambiguity. Additionally, software and software requirements are subject to continuous and rapid change, even during the SDLC. Additionally, it is labour-intense, which makes quality and measurement challenging. As a consequence, slowly established plans based on effort estimates may prove to be inaccurate once development begins.

More recent, agile methods, therefore, adopt a continuous approach to software planning, including estimation. Similar to Waterfall approaches, estimates are produced for small scale software tasks, however, there is less emphasis on aggregating estimates for the purpose of longer term plans. In agile methods, SEE happens more frequently and with a smaller scope. Moreover, SEE is considered as a communication opportunity to help understand and refine the software requirements rather than as a commitment [20]. Therefore, software teams use synthetic internal representations of effort, such as Story Points to convey software effort. These metrics are effective for internal communication within a team, but have no external validity. As a result, Agile development poses a challenge to traditional project budgeting and management, where budgets must be determined in advance to facilitate project control [54].

At the extreme, the #NoEstimates movement started by Zuill [55] views estimates as unnecessary documentation. The rationale behind the movement is that estimates are a form of documentation that Agile principles value less than workable software. The contention in this approach is that the rapidity of delivery of new increments of a software project mitigate the need for estimates to develop longer term roadmaps. This negates the need for budgets and workplans to justify longer term, large scale investment.

On the other hand, #No-Estimate opponents think that estimates are essential to control, fund, and communicate a software development project with the other departments. In fact, they believe that business works that way. Estimates are at the core of business market analysis and trend measurement [56]. An alternative, Beyond Budgeting [57], is a budgeting concept that goes well with Agile development principles. It is better to look at Agile development as on-demand development with a contract over a number of demands.

### 3.1.3 SEE Methods Classification

There have been a plethora of approaches to software effort estimation proposed in the literature, with a particular focus on automation as identified by Jørgensen et al. [2] and Vera

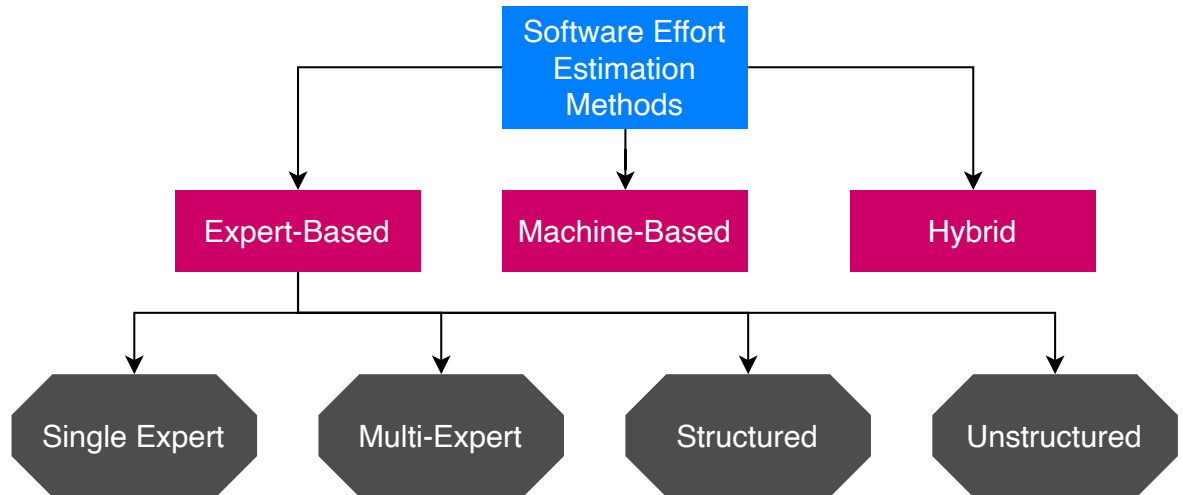


Figure 3.1: Classification of Software Effort Estimation Methods

et al. [58] reviewed different approaches to classification of these methods in taxonomies. Taxonomies are useful for enhancing the understanding of a given topic through mapping of different works onto a common set of definitions and categories. In the context of SEE though, Vera et al. [58] identified considerable diversity in classification approaches adopted in the literature, signalling considerable disagreement amongst authors.

For the purposes of this thesis, a classification is adopted that includes elements from the approaches proposed by Moløkken and Jørgensen [59], Trendowicz and Jeffery [24] and Britto et al. [60]. Moløkken and Jørgensen [59] proposed the categories “Expert-based”, “Model-Based”, and “Other”, however these categories do not explicitly include methods that depend on Artificial Intelligence or Machine Learning technologies. Trendowicz and Jeffery [24] suggest a similar classification, with the three main categories as “Data-driven”, “Expert-based”, and “Hybrid”. However, Trendowicz and Jeffery [24]’s classification is based on two aspects: input type and estimation principle, which make it harder to decide which method falls under which category. For instance, it is hard to say that expert-based methods are not data-driven. Britto et al. [60] also provide a similar classification, with the three categories “Expert”, “Algorithmic”, and “Artificial Intelligence”. However, the artificial intelligence methods also fit under the algorithmic category since Britto et al. did not provide a precise definition.

In the approach adopted in this thesis, the categories employed are “Expert-Based”, “Machine-Based” and “Hybrid”, see Figure 3.1 for an illustration. The expert-based category can follow Trendowicz and Jeffery [24] for further division. It has two sub-categories, based on the number of expert participants: Single-Expert methods and Multi-Expert methods. In addition, the expert-based category can also be divided into Structured methods and Unstructured methods.

Delphi [61] is an early expert-based method. More expert-based methods are derived from

Delphi methods such as Wideband Delphi [62] and Planning Poker [8]. More details about expert-based methods will follow in the next section.

Machine-based methods encapsulate the majority of proposed SEE methods in the literature, including pure model-based methods such as COCOMO [62], artificial neural network methods, analogical methods such as case-based reasoning [63], and others. Such methods rely on a model that is either clearly defined along with its parameters, or indirectly built up, such as in machine-learning techniques. In both branches, such models can rely on a few parameters or require a large amount of data. More details will follow in Chapter 5 about methods that are based on machine learning. Further reading about the different kinds of machine-based methods can be found in Trendowicz and Jeffery [24]’s book under the name of “Data-driven” methods.

Hybrid SEE methods are basically methods that merge expert-based and machine-based methods such as COCOMO-U [64]. Unlike industrial practice, the research literature suggests that such a merge can result in substantially better estimates [65]. The combination can be on the level of the estimation methods, where a third SEE is followed, or based on aggregating final outcomes of expert-based and machine-based methods.

Expert-based methods, are the most commonly used estimation methods. According to [24], 80% of estimates in industry are made by experts without machine-based methods. Similarly, [66] reported that 58% of their respondents used planning poker for estimation purposes. Yet, such methods received the lowest research interest and advancement [2], which places this thesis in a good position for narrowing the gap.

## 3.2 Expert-based Software Effort Estimation Methods

While the literature is not conclusive [7], the majority of studies referred to expert-based methods as the most popular and most often used in the industry as stated by Moløkken and Jørgensen [67]. The thesis definition of expert-based methods follows the literature [7, 24, 60]. It includes any method in which the estimates are produced purely based on human judgement. Usually, a field-knowledgable person or a group of people carry on the estimation process. The processes vary as explained in the following subsections.

Generally, There are two known strategies to follow, bottom-up or top-down. Bottom-up strategies start by estimating the finest grained elements in a work breakdown structure (WBS). Then they aggregate the estimates by gradually aggregating work items into larger scale packages. On the other hand, top-down strategies divide an estimate of the top level project and go down through the structure, allocating effort to the WBS leaves. Jørgensen [7] explored both strategies and concluded that the top-down strategy works better if there is

access to similar historical tasks, and its advantage is that it does not require detailed knowledge in software development. This implies, bottom-up strategy should be the default choice in the case of little or no historical information as stated by Jørgensen [7].

### 3.2.1 Guesstimation

According to the Oxford Dictionary [68], a guesstimate is an estimate that is predicted using guesswork and reasoning. Usually, it is carried out by a single estimator. Guesstimates have been found to be used within the software industry and have been studied in the literature.

Johnson et al. suggests a guesstimation process of four steps:

- Step 1: Identify similar projects that an estimator believes is close to the project which is about to be guesstimated (target). Similarity is based on project size (LOC) and effort. Such projects will serve as input to the LEAP (Lightweight, Empirical, Anti-measurement dysfunction, and Portable) toolkit— A set of software engineering applications that collect and analyze data from individuals that can be used as a data explorer to support effort prediction. [4].
- Step 2: The estimator predicts the target project size.
- Step 3: The estimator browses and analyses the different analytical data that is provided by the LEAP toolkit.
- Step 4: A final effort estimate is predicted by the estimator or selected from the provided analysis.

Johnson et al. [4] found that estimates that are guesstimated are more accurate than those produced by a regression model, if the estimator is supplied with different effort and size analysis information, such as LEAP [69].

### 3.2.2 Wideband Delphi

Most research refers to the Wideband Delphi method [1, 70] as the original expert-based methods. The method went through four development evolutions as described by Trendowicz and Jeffery [24]. Wideband Delphi relies on judgements from several experts, who predict their estimates based on a structured process. Then, estimates are aggregated based on consensus and expert discussion.

The latest version of the method [71] follows an iterative pattern to reach consensus. The process consists of seven steps, and three of them are meetings:

- Step 1: A project manager prepares the initial planning and project description, where the manager defines the problem, scope and relevant historical records.
- Step 2: The project manager identifies an estimation team with a coordinator.
- Step 3: The coordinator starts a kick-off meeting to introduce the planning materials that were prepared earlier by the manager. (The first meeting)
- Step 4: Each estimator predicts an initial estimate based on the kick-off meeting.
- Step 5: The coordinator starts an estimation meeting to collect and present estimates anonymously. The team discuss the estimates, especially the outliers. If they agree, the coordinator moves to the next step. Otherwise, the team repeats steps 4-5 until they reach an agreement or the dedicated time finishes. (The second meeting)
- Step 6: The coordinator merges the outcomes of the estimation session.
- Step 7: The estimator calls for a review meeting to review the outcome documentation and adjust it if necessary. (The third meeting)

Wideband Delphi has several advantages. It does not require excessive information of the project/job being estimated. Additionally, its structured process helps in reducing human biases, and yet it is easy to apply. However, the iterative process may consume a significant amount of time and effort, and its outputs are not reusable [72].

### 3.2.3 Estimeetings

Taff et al. [19] suggested a lengthy estimation method that is applied over six months, and it can run up to two years. Estimeeting is part of a larger process called the “Front-End Process” to convert software concepts to a final list of features that are ready for development. Estimeeting was designed in the context of mega-projects which have several subsystems folded inside them.

The Front-End process is divided into three phases and it produces three documents: feature specification proposal (FSP), feature architecture proposal (FAP), and Detailed Estimates. These three documents are combined in estimeeting sessions to produce the Final Feature List. The Detailed Estimates document, which is what concerns this thesis, is developed by the following steps:

- Step 1: schedule the estimeeting events since they need to be run over a long time and incorporate numerous people.

- Step 2: identify and call the required team members to attend the estimating session as well as the optional team members.
- Step 3: distribute the FSP and FAP to the estimation team to study
- Step 4: hold an estimating session to present and discuss the FSP and FAP.
- Step 5: estimators then record their estimates, after consulting with each other, in the subsystem estimation form (SEF).
- Step 6: a feature engineer follows up on the estimating outcomes and resolves any outstanding issues.
- Step 7: the feature engineer fills in a Feature Estimate Summary Form and collates it with the FSP and FAP to represent the formal output of the Front-End process.

An interesting part of SEF where the estimator records their estimate for each subsystem that is part of a given feature is that it contains a section to break down the work that makes up the estimate. By doing so, an estimator gives a rationale to help understand the estimate.

However, Moløkken-Østvold and Jørgensen [73] described Estimeetings as a complicated process that consumes a lot of time and it might work only for the large project that was designed for. As far as the author can ascertain, there are no other reports of evaluations of estimating in the literature that might address this question of generalisability.

### 3.2.4 Stochastic Budget Simulation (SBS)

The primary feature of SBS as an estimation method is to manage the uncertainty that usually takes place during the early stage of any project. Elkjaer [74] suggested SBS, which uses a three-point (minimum, most likely, and maximum) expert estimate. The goal here is to rely on probabilistic range estimation instead of a single-point estimate. Then SBS uses the estimates to run a statistical simulation based on a selected probability distribution such as the triangular distribution. The SBS process consists of four steps:

- Step 1: create a WBS that groups related matters together.
- Step 2: identify general risks that affect all the WBS items and list them under generic risks.
- Step 3: predict the three-point estimate for each WEB leaf.
- Step 4: use an algorithm to run a simulation to calculate the overall cost and identify the local uncertainty for each item.

As per Elkjaer's description, SBS is more like an analytical tool for a project manager to assess uncertainty for a given project. Instead of making an estimate, SBS produces a range of estimates along with their probabilities, then a project manager uses the SBS output to determine the most likely estimate for the project.

Chou [75] found SBS a usable tool to get an overview of the project total cost. However, he points out that SBS missing out some aspects such as variates correlations which play a critical role in balancing the outcome estimates not to be overestimated or underestimated. Chou [75] suggest that using probability density function examination is a key to manage the variates correlations.

### 3.2.5 Sparse Data Method (SDM) using Analytic Hierarchy Process

SDM, as described by Shepperd and Cartwright [76], is another method to use expert judgement towards predicting effort estimates. It uses an Analytic Hierarchy Process (AHP) to design a hierarchy that decomposes the problem into smaller elements for easier comprehension and more accurate estimates. SDM puts software effort as the root node in AHP hierarchy. Then, it lists subcomponents in the second level, known as prediction elements. The elements can be different development areas or different subsystems. Additionally, SDM has a precondition, which is acquiring the effort for at least one of the subcomponents, referred to as the "reference point". The expert's role is to judge the relative size of each subcomponent to the others in a pairwise comparison. For instance, if the subcomponents represent different development tasks such as coding, testing and deployment, the expert's job is to compare coding to testing and deployment in terms of effort by predicting how much effort will be needed for coding compared to testing and deployment, e.g. coding = 1/2 testing and coding = 2 deployment.

There is a limit on how the prediction elements vary, called Saaty's homogeneity requirement. For better prediction accuracy, the elements should be similar to each other. Moreover, SDM follows a seven-step process:

- Step 1: Identify the prediction elements. They can be a system subcomponent, or different project stages.
- Step 2: Evaluate Saaty's homogeneity requirement for each element and discard any element beyond the limit. Saaty's homogeneity is a measurement that can be used to assess elements disparity, and then cluster similar elements altogether [77]. According to Saaty, the mind cannot compare widely dissimilar elements, and thus, Saaty believes the homogeneity limit gives meaningful comparisons.



- Step 3: Select at least one reference point.
- Step 4: Select the comparison criteria. Shepperd and Cartwright used software effort.
- Step 5: Make a pairwise comparison between the elements.
- Step 6: Calculate the relevant contribution of each element to the root node using the principal eigenvector [78]. Principal eigenvectors reflect numerical judgement-derived priorities, and thus it can be used to measure the weight of each element.
- Step 7: Determine the value of other elements using the reference point.

While Shepperd and Cartwright and others such as Trendowicz and Jeffery consider SDM to be an expert-based method, it is better to look at it as an analogy method that belongs under the machine-based methods. Essentially, experts in SDM do not predict an estimate. Rather, they make a relative comparison and the final estimate is based on the reference point which is an input to the SDM method.

### 3.2.6 Planning Poker

In 2002, Grenning [8] proposed Planning Poker as an estimation method that is derived from Wideband Delphi [1]. It is by far the most recent advancement in the popular expert-based effort estimation methods. Grenning's proposal was made in the context of Agile development and part of Agile planning. Later, Cohn [5] detailed the approach in his book. The name *PLANNING POKER* is also registered as trademark [79].

The method uses a predefined story point series that is printed on poker cards, and estimation sessions are basically a playing session using the poker cards. This is where the second half of the method name comes from. Normally, a deck shows the Fibonacci sequence on its cards, and the player picks the card that represents the number of story points that will be needed to implement a user story, see Figure 3.2. In Agile development, a user story encapsulates the software requirements in the form of a story [80]. For example, "As a student, I can add and delete courses during the registration period so that I will be able to attend the courses and fill my course requirements."

A planning poker estimation team consists of: the product owner, the project development team, a coordinator, and sometimes project stakeholders as observers. A product owner is the member of the development team who is in charge of a given user story and wants to maximise its value. The process of Planning Poker generally consists of four steps:

- Step 1: A product owner presents a user story and discusses it with the planning poker team (estimators).

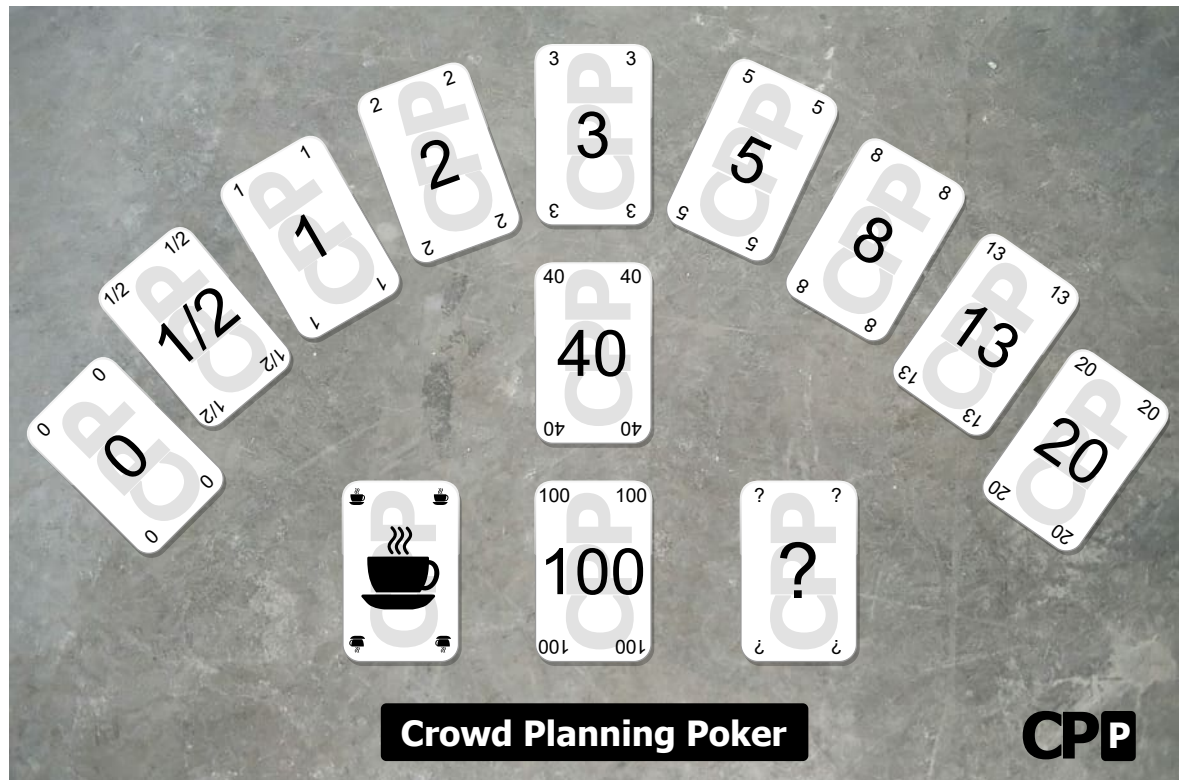


Figure 3.2: Example of a Planning Poker Deck

- Step 1: After understanding the user story, every estimator puts one of the cards in a face-down position.
- Step 3: The estimators reveal their estimates simultaneously by flipping the cards. Then they discuss the most extreme estimates.
- step 4: If the team has reached a consensus, they move on to another issue. Otherwise, the team repeats steps 2–4 until an consensus is achieved. Usually, the session coordinator sets a time limit for each user story to prevent an infinite loop of estimation.

Moløkken-Østfold et al. [26] found Planning Poker as an estimate combination method produced better estimates in comparison with unstructured and mechanical combining of estimates. More recently, Mahnic and Hovelja [81] confirmed the findings of Moløkken-Østfold et al. that Planning Poker is a better way to do software estimation using expert judgements. However, Moløkken-Østfold et al. tentatively indicates that the more diverse the group, the better the estimation, without providing details on the diversity of the group. Moreover, both studies stressed the industrial experience in general without tapping software project experience. In addition, the scalability of Planning Poker has not been addressed in the previous research. Perhaps, being a labour-intensive SEE method and solely reliant on human intelligence and experience is an obstacle for scaling Planning Poker. However, crowdsourc-

ing platforms represent a good environment to test the diversity, experience, and scalability issues.

### 3.3 Machine Learning Software Effort Estimation

With recent advancements and the popularity of machine learning (ML) applications in different disciplines, several researchers have applied ML techniques to help in estimating software efforts. ML is mainly used to build data models that determine the similarity between jobs as whole projects or smaller units such as user stories. The most recent review studies [82, 83] that investigate ML-based methods have identified nine techniques: Artificial Neural Network (ANN), Support Vector (SV) Machine/Regression, Bayesian Network (BN), K-Nearest Neighbors (kNNs), Decision Tree (DT), Genetic Programming (GP), Case-Based Reasoning (CBR), Genetic Algorithms (GA) and Association Rules (AR).

According to both surveys [82, 83], ANN, SV, DT, and CBR are the most investigated techniques. An overview of these methods will follow in the next subsection.

#### 3.3.1 Artificial Neural Network (ANN)

As its name suggests, ANN is a computing model that imitates a biological neural network. It consists of neurons and a network of connections that links those neurons based on weights. Feed-forward ANN is the simplest model.

ANN occupies the first position in Ali and Gravino [83]’s survey, where 60%(45) of the surveyed studies used ANN. According to a recent ANN survey [9], there are four commonly used ANN methods used in SEE: feed-forward neural networks, recurrent neural networks, radial basis function (RBF) networks, and neuro-fuzzy networks. Idri et al. also agrees that a feed-forward neural network with a back-propagation learning algorithm is the most commonly used ANN method. However, Idri et al. divide ANNs into two general categories based on their perception as ANN feed-forward networks or feedback networks. Several studies [85, 86, 87] compared ANN methods in terms of their accuracy and under which conditions an ANN can best perform. For example, in the context of feed-forward ANNs, Hamza et al. [9] suggest that using Levenberg–Marquardt as the learning algorithm works better for data with noise than a back-propagation algorithm.

#### 3.3.2 Case-Based Reasoning (CBR)

CBR, sometimes referred to as analogy-based, is the second most researched SEE method according to Wen et al. [82]’s survey with 37%(43) of the survey studies. CBR has a fun-

damental assumption: alike software projects (for a given set of features) have similar costs [63]. To measure the similarity between the historical cases and the new one, similarity measures such as Euclidean distance and Manhattan distance are used, and CBR studies follow a process of three steps as described by a recent survey [88]:

- Step One: feature and case selection,
- Step Two: similarity evaluation, and
- Step Three: adaptation.

To overcome CBR challenges in selecting features and measuring similarity, other artificial intelligence techniques such as fuzzy logic and genetic algorithms are used. Idri et al.'s survey gives details of different combinations of CBR and other techniques. According to the survey, the Mean Magnitude of Relative Error (MMRE) for CBR ranges from 19% to 35%, which means the estimates are acceptable [88].

### 3.3.3 Decision Tree (DT)

Decision Tree is a computing model with a binary tree structure. The tree starts with a root node and it splits based on predictor variables (tree features) until the end of the tree where nodes become leaves (have no children). DT, also known as Classification and Regression Trees (CART), has been developed into different approaches such as Random Forest and Treeboost, and they are also combined with other techniques such as fuzzy trees [89].

DT gained its popularity due to the ease of its concept and structure. According to abdelali et al. [10], DT is the most used method to predict effort estimation. Ali and Gravino [83], Wen et al. [82] rank DT among the top three ML techniques.

However, conventional DTs have several drawbacks such as overfitting and lack of global optimisation [10]. Therefore, abdelali et al. [10] suggested Random Forest as an improvement that resolves the traditional DT limitations, and their study evaluation shows a promising improvement. Recently, Abdelali et al. [90] suggested an ensemble method of optimal trees that performs significantly better than regression trees.

### 3.3.4 Support Vector (SV)

The Support Vector method, like many other ML methods, can be used for classification of non-numerical inputs, in which case it is called Support Vector Machine (SVM); when it is used as a regressor with numerical inputs it is called Support Vector Regression (SVR).

According to Nayak et al. [91], SVM has been proposed within statistical learning theory and structural risk minimization, which guarantees the risk to be bounded, and thus, it can handle a large number of feature spaces.

Several studies suggest that SVM has better performance than other techniques. For instance, Corazza et al. examined SVM and published two papers [92, 11] stating that SVM outperformed CBR and BN.

### 3.3.5 Performance of ML-Based

There are a couple of accuracy indicators that are used in the literature to measure different SEE methods in general, including the ML-based methods. The most popular indicators according to Wen et al. [82], Ali and Gravino [83] are: MMRE (Mean Magnitude of Relative Error), MdMRE (Median Magnitude of Relative Error), and Pred (25) (Percentage of predictions that are within 25% of the actual value). For MMRE and MdMRE, lower values indicate better methods, but for Pred(25) the higher the value, the better the method.

Considering MMRE as an accuracy indicator for different ML-based methods, Wen et al. [82] found that ANN and SVR are the most accurate methods, with a median MMRE of 35%, followed by CBR and DT. Ali and Gravino [83], on the other hand, suggest similar results by putting ANN in first place followed by SVM and DT. Both surveys also show that ML-based methods, especially ANN and CBR, outperform non-ML methods such as regressions and COCOMO.

One important aspect of such a comparison is its context. These methods have been competing against each other using several datasets, including Desharnais [93], NASA [94], COCOMO [1], Albrecht [95] and others. Therefore, a method, say DT, that performs better than ANN in one study based on a certain dataset may not outperform ANN for other datasets, as pointed out by Ali and Gravino [83]. For example, ANN and SVM outperform DT in two papers [96, 97]. However, DT outperforms ANN and SVM in five studies [98, 99, 97, 100, 101]. Moreover, the fact that ANN is the most investigated, and perhaps enhanced, method, may skew the comparison in favour of ANN. Finally, the datasets that are used are quite old. The most recent one is Tuketuku[102], which is a decade and half old. Therefore, even if such a comparison is acceptable, it reflects the methods' performance against quite old software development datasets which have significantly changed nowadays.

### 3.3.6 Contexts Where ML-Based is Recommended

In general, ML-based methods demand a large number of historical data points in order to learn and train their models. However, some methods can work better under certain con-

ditions, e.g. a small data set. Wen et al. [82] identified several conditions where different methods perform better than others. The conditions can be grouped into three categories: the size, nature, and quality of the dataset. For instance, CBR cannot work with a dataset that has low quality, i.e. is missing some values. ANN also performs poorly with small datasets. In the larger landscape, Trendowicz and Jeffery [24] proposed a sophisticated decision tree to select which estimation approach is better for a given context. It is not just for ML-based methods; it includes all SEE methods. Trendowicz and Jeffery [24]’s approach identifies six goals that are based on thirteen criteria, such as data requirement and complexity.

### 3.4 Review of Comparative Research of Expert-Based and ML-Based SEE

Although machine-based SEE methods appear to be more sophisticated and advanced, an overwhelming number of studies indicate that expert-based SEE methods are the most used methods for estimating software development effort. Kassab and Destefanis [103] conducted the most recent survey, with 117 participants completing the survey. The survey covers a wide range of project industries, including finance (13%), utilities (9.7%), and defence (8.2%). The participants worked on different kinds of projects, such as web-based (22%), database (19.6%), and web services (13.6%). The survey confirms previous industry studies [6, 104, 105] that expert-based methods are by far the most dominant either in waterfall-based or Agile-based SDLC.

Several Agile research studies also suggest that expert-based methods are the most popular methods. Usman et al. [18] surveyed 60 software professionals around the globe, with participants from Europe and America representing 56% of the sample. Expert-based methods such as Planning Poker were the most dominant methods, with more than 63.33% of participants using them. The most recent case studies [106, 107] also confirm that there is not much change in the practice of effort estimation, and expert-based methods are still the most used SEE approach.

On the other hand, Jørgensen [65] referred to a literature paradox regarding the fact that expert-based methods are dominant, but most SEE literature investigates and improves non-expert SEE methods [2, 108]. Software engineering textbooks and software tools for SEE all, directly or indirectly, promote non-expert SEE methods [109]. One reason that may provide some rationale for such a paradox is that expert-based methods are simple and intuitive. The following subsections provide reasons for the popularity of expert-based methods.

**Complexity of non-expert methods.** A prerequisite of non-expert SEE methods is access to a large amount of data in order to start predicting acceptable estimates. However, such data might not be available, especially in the early stage of projects. Yang et al. [104]’s survey suggests that most estimation activities happen during “Initial project proposal stage” (57%), “Feasibility study” (67%), and “Requirement” (74%). In the same survey, the majority thought that model-based SEE methods are costly to adapt and yet offer an insignificant benefit.

On the other hand, expert-based methods have no prerequisites other than an expert’s intuition [2]. They follow simple steps and use common sense. Trendowicz and Jeffery [24] recognises the simplicity of expert-based methods as one of their strengths.

Another source of complexity is the SEE literature itself. This complexity is best reflected in the inconsistent taxonomies [58] and measurements [2]. At the same time, industry standards such as [48, 110] have yet to precisely define SEE vocabulary and processes.

**Expert and non-expert estimates have similar accuracy.** Surprisingly, expert and non-expert SEE methods produce similar estimate accuracy, and thus the software industry opts for the most intuitive methods, i.e. expert-based methods. That leads Jørgensen [65] to ask if it is reasonable to pursue improvements in accuracy in future SEE research and development.

Moløkken-Østfold et al. [105]’s survey lists different reasons for selecting expert-based SEE methods, and the most selected reason is that expert estimators had successful previous estimates. This may be translated as a form of trust of expert judgement over machine-based judgements.

While studies such as Addison and Vallabh [111], Moores and Edwards [109] indicate that estimate accuracy is important, it does not imply that estimates are expected to be 100% accurate. In fact, *Accurate* has been interpreted differently in the industry. For instance, Moores and Edwards [109] indicated that estimates in the range  $\pm 20\%$  are considered accurate estimates by industry professionals. Unless suggesting a significant accuracy gap between expert and non-expert methods, the software industry will continue its adoption of expert-based methods.

**Lack of supportive details, and flexibility.** Among the weaknesses of machine-based SEE methods noted by Trendowicz and Jeffery [24] is the lack of supportive details. While estimates are the ultimate outcomes, it is hard to trust an estimate just because a machine says so without more information. Supportive information, which expert estimators usually provide, can be a rationale behind the estimates. Such side details provide an environment for development team members to negotiate and reach a better understanding, as illustrated in

the Planning Poker method [8]. In fact, Agile research such as Tanveer et al. [106] suggests another purpose of SEE, which is synchronising team understandings.

Moløkken-Østvold [112] indicates that the lack of flexibility in machine-based methods is another reason behind the popularity of expert-based methods. Flexibility can be interpreted in several ways, such as in changing the model to consider different kinds of information or in reusing the model in different contexts [24].

In summary, the literature strongly suggests that expert-based estimation methods remain the most reliable, but they lack the potential for scalability of automated methods, such as those based on ML. The next section explores research in human computation as a potential way of developing scalable and reliable software effort estimation methods.

## 3.5 Human Computation

Human computation is an emerging research area in which human intelligence is used in a machine-managed process, computation, or algorithm. As a consequence, there are several competing (and in some cases conflicting) definitions of the field. Quinn and Bederson [14] review these in an attempt to propose a general conception of human computation and its relationship to other fields. They argue that Human Computation lies largely within the more general area of Collective Intelligence, alongside Social Computing and Crowdsourcing. All these fields involve the utilisation of human intelligence and decision making within computational processes, but from different views and with different applications [14]. Thus, Quinn and Bederson created a classification system that relies on six dimensions, such as motivation, quality control and aggregations, in order to distinguish between different research areas and find gaps between them. What concerns this thesis is the area of human computation where a human crowd is utilised to substitute machines for decision making purposes within an algorithm. This section reviews work that addresses this concern: the nature of human computational tasks, they are decomposed and how a human worker is managed as a computational resource.

While early applications made the human computation tasks independent and identical, Little et al. [113] and Bernstein et al. [114] succeeded in using a human computation worker to accomplish dependant tasks that result in an accumulated effort. Kamar et al. [115] mentioned that a human computation worker is able to do different kinds of task, such as solving a problem and decomposing a goal into a list of tasks. Little et al. [113] took the extreme end of involving human computation workers in an algorithmic style. In fact, Little et al. is able to invoke human workers via an imperative-style programming language to perform a certain task and then get the response back to the program. Most of the research that aims to



harvest human intelligence in relation to human computation can be categorised under three research areas: Crowdsourcing, Collective Intelligence and Social Computing.

### 3.5.1 Crowdsourcing

Asking human crowds in an open call to solve a certain problem in exchange for incentives is not a new method. In the 1714, the British government called upon public to solve a navigation problem in exchange for £20,000 Sobel [116]. Since then, the concept of solving a problem by utilising the crowd continued evolving until 2006 when Jeff Howe used the term *Crowdsourcing* in his article: “The Rise of Crowdsourcing” [117].

Howe defines crowdsourcing as:

“Crowdsourcing represents the act of a company or institution taking a function once performed by employees and outsourcing it to an undefined (and generally large) network of people in the form of an open call.”

Based on Howe’s definition, crowdsourcing is a special type of outsourcing. It has two explicit characteristics and one implicit, as follows.

- **Explicit:** Task accomplished by a group of *anonymous individuals*
- **Explicit:** The anonymous individuals are recruited by an *open call*.
- **Implicit:** The anonymous individuals *exchange money* for the accomplished tasks.

Since 2006, the term “crowdsourcing” has been used to describe different initiatives and projects, such as Wikipedia and Amazon Mechanical Turk. However, such usage of “crowdsourcing” amplified the confusion around this term. Thus, Arolas and González-Ladrón-de-Guevara [118] reviewed 32 articles that contain definitions of the term in three different contexts, which are: the crowd, the initiator, and the process. Then, Arolas and González-Ladrón-de-Guevara concluded with their definition, which is close to 2006’s definition with a slight generalisation. Hosseini et al. [119] also reviewed all the articles that defined crowdsourcing that were published between 2006 and January 2014. They ended up with 113 definitions of crowdsourcing. Hosseini et al. [119] converted the text representation of each definition to an itemised list of features that can be found in any crowdsourcing initiative. Based on Arolas and González-Ladrón-de-Guevara [118], Hosseini et al. [119], crowdsourcing is not limited to 2006’s definition any more. In fact, it is better interpreted as a set of features where the variation and availability of such features result in different forms of crowdsourcing initiatives. However, there is a set of core features that are accepted by all

collected definitions. These are: large number of people, the workers are humans, there is an initiator of the crowd task, the task is defined. That leads us to our own combination of crowdsourcing features. These are: micro-tasks, large population, anonymous crowd, competence and skilled crowd, accumulated effort, self-managed, defined project goals and sponsored work.

Crowdsourcing has been used in different disciplines such as Computer Science, Business and Management, Medicine, Law, and Sociology. Most of the crowdsourcing research has been done under the discipline of Computer Science Hosseini et al. [119]. More details and applications of crowdsourcing will follow in the subsection on software engineering applications in human computation.

### 3.5.2 Collective Intelligence

Collective Intelligence has the most loose definition to describe its boundaries. In fact, it almost includes any team work, online or offline, that collects individuals' intelligence into united cohesive work. Malone et al. defined collective intelligence as "Groups of individuals doing things collectively that seem intelligent" [120].

Originally, the concept of collective intelligence was meant to describe group decisions with the claim that a decision made by a group of individuals is better than a decision made by a solo brain. Scholars extended that concept to include group processes and group organisation [121].

Research studies in the area of collective intelligence concern the theoretical part of group thinking, and it represents a good start in understanding the foundation of thinking, acting, and organising groups of individuals Miorandi et al. [122], Bonabeau [123].

### 3.5.3 Social Computing

Regardless of our purpose, we, as humans, interact with each other by nature, and this interaction has been carried out over different mediums, such as face-to-face, over the phone, or using a computer application. The last example is what social computing is about. Wang et al. [124] list several definitions along with their own.

Social Computing is concerned with utilising human intelligence from the angle of what can be produced by facilitating interaction between individuals. For example, blogging and interacting over social media, e.g. Twitter, are how human intelligence has been harvested to create social and blogging content. In areas where face-to-face interaction is difficult, e.g. Distributed Software Development, Social Computing can play a significant part in facilitating human interaction Niazi et al. [125].

By facilitating human interaction, Social Computing can generate a huge amount of useful data that can be used in understanding crowds. For example, Ortu et al. [126] was able to collect a huge amount of communication data, about three million data records, from open-source projects. Moreover, making human interaction easier will increase the number of participators in the process of harvesting their intelligence, as illustrated in Storey et al. [127]’s study.

### 3.5.4 Quality Assignment in Human Computation

As it has been mentioned earlier, most of the publications in the area of human computation concern its applications. However, there are a number of publications about enhancing and controlling the quality of human computation outcomes. Allahbakhsh et al. [128] reviewed previous studies that investigated crowd quality controls, which are the measurements that can be taken to ensure a quality outcomes. Allahbakhsh et al. list quality-control approaches along with a quality-control taxonomy. For example, one of the quality-control approaches featured in the study is: “Effective Task Preparation”, which is a technique that devotes good preparation of the task by providing an unambiguous description of the task. Such a technique makes the task in a defensive design— that is, cheating is not easier than doing the task. The rest of the publications have been categorised into four categories: gold-standard quality controls, machine-learning quality controls, statistical, and crowdsourcing quality controls.

#### Gold-Standard Approach

Gold-standard quality control refers to the way of asking crowd workers questions with known answers and then checking whether the workers answered correctly. Then, the workers are scored based on these gold-standard questions. There are several studies that implemented this method in different ways, such as Corney et al. [129], Donmez et al. [130], Bhardwaj et al. [131]. More interestingly, Oleson et al. [132] suggest a novel way to create gold-standard units in a programmatic way based on previous gold units. Oleson et al. test workers with the created gold units to measure their quality score and then judge their contributions.

#### Machine-Learning Approach

Basically, this approach involves machine-learning algorithms to learn from crowd workers’ inputs to score the workers. For example, Zhu et al. [133] train their algorithm using quality-related measurements, such as time spent on a task, and then used that to predict the quality

of the submitted answers. Mashhadi and Capra [134] also use machine learning with novel inputs, i.e user travel duration between known locations, to control the quality in a Ubiquitous Crowdsourcing setting. Based on that, they can identify mobility patterns (in term of duration between locations) and the score of the user's previous contribution to produce a weighted average of the user contribution.

### **Statistical Approach**

In such a quality-control method, a statistical model such as expectation maximisation is used to generate a success probability for a crowd worker contribution. Ipeirotis et al. [135] use an enhanced version of the expectation maximisation (EM) algorithm. EM can deduce the error rate of workers using the maximum likelihood of total submitted answers. Ipeirotis et al. enhanced the EM algorithm by differentiating between the error rate and worker bias. Sarma et al. [136] also worked on enhancing the EM algorithm to be used in a global quality setting rather than a local one. Their algorithm finds the global optimal estimation of correct task answers. They leveraged the idea of grouping similar items and treating them as one item. In addition, they filter items based on roles that are implied by a requester.

### **Crowdsourcing Approach**

Letting a crowd worker (the judge) review an outcome of another crowd worker (the worker) is also used to control and measure the crowd outcomes. Baba and Kashima [137] suggest a two-stage quality control for creative tasks. In the first stage (creation stage), an artefact is being created, and in the second stage (review stage), the artefact is being reviewed. Baba and Kashima's contribution is the statistical model to measure the judge ability and the reviewer bias and convert these into a document quality score.

Although there are some research publications about quality control of crowd worker outcomes, they still do not cover all the angles of the outcomes, for instance, controlling and measuring the quality of subjective answers for software engineering management tasks. Only current methods such as redundancy are being used to ensure the quality in software engineering crowdsourced tasks LaToza et al. [138]. However, there is a need for a crowd-adoption method to ensure the quality of crowd workers' outcomes.

## 3.6 Software Engineering Applications in Human Computation

Computer science has the most application research in the area of human computation and crowdsourcing. Several researchers studied crowdsourcing applications in different software engineering niches. Mao et al. [15]’s paper is a survey paper about different software engineering applications. It covers all the previous research work from Jan 2006 to Mar 2015. Mao et al. divided the application of crowdsourcing in software engineering based on the phases of the software development life cycle. For example, in software requirement engineering, Mao et al. mentioned Lim and Finkelstein [139]’s paper as an example of how crowdsourcing is being used in requirement elicitation. Mao et al. include another 16 research papers in this specific niche. Moreover, Mao et al. mentioned several crowdsourcing challenges such as task decomposition and quality assurance.

In addition, Stol and Fitzgerald [140] aimed to show and emphasise the challenges of using crowds in software development with a real-world case. Stol and Fitzgerald [140] present a case study of using crowds to develop software. Stol and Fitzgerald list all the challenges of adopting crowds to develop software, e.g. the task decomposition challenge. They conclude that crowds are still not cheaper, better in quality, or even faster. Notably, Stol and Fitzgerald [140] used crowds as if they are contractors, which may explain their findings. Nevertheless, Stol and Fitzgerald [141] built a framework for outsourcing software development to the crowds. They concluded that there is a need to research this area using their framework. Interestingly, Stol and Fitzgerald [141] presented excellent questions that can be used to narrow any research in this area.

Li et al. [142]’s book has a paper titled “Crowd-sourcing for Large-Scale Software Development” that discusses open source, distributed development and crowdsourcing. Li et al.’s paper is one of the rare papers that includes all the topics together. It summarises the topics’ features in one table. Moreover, it lists an architecture, development process, and maturity model for the crowd software development. An example of thinking out-of-the-box, Yu et al. [143] provided a dataset from a gaming platform that simulates the software development process using crowds. Yu et al. [143] list some technical challenges in human computing such as measuring the temporal changes in worker behaviour.

In software development and programming, there are couple of studies that suggest solutions for writing software code using human crowds. For example, LaToza et al. [138] suggest a crowd development platform to develop software. Others, like Tsai et al. [144], Minder and Bernstein [145], Cochran et al. [146], LaToza et al. [147], have suggested different solutions for relatively the same purpose.

The software engineering discipline encompasses implicit images of systematised harvesting

of human intelligence, for example, open-source development Raymond [148], distributed (a.k.a. global) software development Jiménez et al. [149], and search-based software engineering Harman and Jones [150]. These images suggest that human computation is native in the software engineering discipline. Human computation as a research area is considered an emergent discipline, and similar software engineering applications in crowdsourcing have started adapting and moving to human computation. Several studies discussed most of the applications, such as Kittur et al. [151], Casey [152]. More specific research studies are dedicated to certain software engineering areas, including team communication, task planning, and team management.

**Team communication.** Cubranic et al. [153], Omoronyia et al. [154] proposed an automated method of keeping team communication intact and present whenever an individual worker needs it. They bring communication data into context. For example, while you are working on part of a certain artefact, you can figure out all the communication relating to that artefact and which people have worked on it. Data mining and machine learning have been utilised to bring such innovation into the communication context. Other studies Handel and Herbsleb [155], Fitzpatrick et al. [156], Guzzi et al. [157], Korkala and Maurer [158] suggest tools and practices to help workers be more efficient by using human computation in their communications.

**Task planning.** There are several studies discussing how crowds can do planning, such as Flostrand [159], Kaivo-oja et al. [160], however, very few studies discuss planning as an activity that can be enhanced by human computation. Mao et al. [15] discussed the cost perspective of job planning. Task decomposition, as part of planning, has been discussed. Alkhatib et al. [161] reviewed the history of piecework and lists how is it possible to learn from that literature how we can decompose large work into small pieces. While Hoßfeld et al. [162] offer a model of work granularity, Kulkarni et al. [163] suggest an application that uses crowds to decompose large tasks. After decomposing tasks, prioritising them is also important, to get the important work done first. That can be done using some data mining techniques, as has been done in the context of prioritising issues and software bugs Lamkanfi et al. [164].

**Team management.** Individuals are able to perform simple and small tasks, however, real-world tasks are large and complex. Thus, forming a team and organising it in a way to accomplish certain tasks is essential. Team formation has been discussed by several studies, such as Li and Shan [165], Gao et al. [166], Anagnostopoulos et al. [167]. However, these teams are being formed to be managed manually by humans. There are few studies in which

teams need to be formed and managed by machines; one example is Park et al. [168]. Organising team work was the concern of Valentine et al. [169]. They suggest flash organisation, a reflection of traditional organisational theory in the context of human crowds. Nevertheless, most of the studies need data, such as the workers' skillsets, for their solution to work properly Huang et al. [170], and other research studies require human intervention Wang et al. [171]

## 3.7 Summary

This chapter explored the state-of-art in topics related to the thesis. It started by reviewing software effort estimation research in different contexts, including software development project management, software maintenance, and Agile. The aim was to cover software effort estimation from different perspectives and sort out the meaning for each one. Therefore, a general understanding of the estimation process was captured and described in a general classification of existing SEE methods.

Then, Section 3.2 started with the first branch of SEE methods, expert-based methods. It listed six expert-based SEE methods that are featured in the literature and related to the thesis topic. Then, this section explained the steps of each method and reviewed related literature.

After that, Section 3.3 reviewed machine-based SEE methods. It started by describing the popular machine-based methods along with relevant literature for each method. Then, it explored comparative research investigating the performance of the methods. In this section, more details were devoted to which contexts machine-based methods have excelled in and are recommended for, according to the reviewed literature.

Then, Section 3.4 reviewed comparative research of expert-based and ML-based SEE. It explored the literature that compares both SEE methods from different perspectives, including complexity, performance, and flexibility. This section concluded that expert-based estimation methods are recommended by the literature for reliability. However, they lack scalability, and thus, human computation may offer better options to design a reliable and scalable SEE method.

After that, Section 3.5 reviewed relevant literature to human computation and surrounding topics, including crowdsourcing, collective intelligence, and social computing. Human computation is an emerging research area and thus sometimes it gets confused related topics on social computing, collective intelligence, and Crowdsourcing. This section reviewed all of them and explained the overlapping areas. It also reviewed quality in crowdsourcing, which is a well-known issue that hinders the progress of advancing research that uses crowdsourcing.

Finally, Section 3.6 explored related and recent applications of human computation in software engineering. It illustrated how several software engineering topics have been advanced using human computation, including team communication, management, and task planning. As explained in previous sections, expert-based SEE methods attracted less attention from research communities. At the same time, these methods are the most used ones in the industry of software development, but are also known to be labour intensive. Thus, in this thesis, research is undertaken to address the gap in the literature by investigating means of addressing the scalability of expert based estimation methods (specifically planning poker) through a combination of automated and human-computation based approaches.



## Chapter 4

# JIRA Open Source Software Effort Dataset

The previous chapter identified the potential for the development of novel approaches to software effort estimation, based on a combination of machine learning and/or human computation as a research gap. Empirical research on software effort estimation (SEE) requires previously collected data on software effort measurements in order to evaluate different estimation methods. The research presented in this thesis requires a dataset of software development tasks that included descriptive text (corpus), actual cost, and expert estimates, as outlined in Chapter 2. Since the literature features no such dataset, this chapter introduces a new dataset, JIRA Open Source Software Efforts (JOSSE), that meets the necessary requirements.

This chapter describes: (1) the method for compiling a new dataset, (2) the methods adopted for refinement of the raw data, (3) a characterisation of the resulting dataset in terms of quality, and (4) a method for further preparation of the dataset in a form that can be used for machine-learning experimentation.

The next section will review topics associated with SEE datasets in previous studies and explain relevant research concerning quality measurements of SEE datasets. Section 4.1 describes previous studies that have developed and used SEE datasets, and then summarises a selection of popular and publicly available datasets. Section 4.3 explains the new dataset collection method that is inspired by Ortu et al. [126]’s work. Section 4.4 details the dataset processing method that incorporates Bosu and Macdonell [172]’s taxonomy. In addition, the impact of several further options for refinement, drawn from the literature, are evaluated in terms of their impact on specific quality metrics. Finally, a summary section concludes this chapter and illustrates the next step in this thesis.

## 4.1 Publicly Available SEE Datasets

Several datasets have been recorded in the SEE literature. Usually, datasets came out of SEE evaluation studies. Then, later researchers adopted the same dataset if it was publicly available. According to ML SEE surveys [82, 83, 173, 174, 175], the top five datasets that are used are: COCOMO [1], Desharnais [93], NASA [102], ISBSG [176] datasets, and Zia's dataset [177], and there are over twelve datasets that have been used more than four times.

SEE researchers reuse current datasets in the literature [93, 178] with the aim of evaluating their advances using the same datasets. The PROMISE repository [179] keeps some of those datasets publicly available (20 were listed on the date of submission). The majority of these datasets contain no textual properties, e.g. task description. Instead, they consist of numerical properties such as lines of code for targeted software projects. In addition, they are project-based datasets, meaning that the data points (cases) are software projects.

To give an example of existing datasets, four popular and publicly available datasets are described in this section. Two of them have a text corpus among their properties, and two have only numeric properties. Two datasets are drawn from open-source communities, and the rest represent commercial projects. Tables 4.1 and 4.2 illustrate summary information about the datasets and their properties. The following is a brief description of the datasets.

- **Deep-SE:** The Deep-SE dataset is derived from Choetkiertikul et al. [180]'s study. Its data was collected in a similar way to JOSSE. The data was collected from nine open-source communities (Apache, Appcelerator, DuraSpace, Atlassian, Moodle, Lsstcorp, MuleSoft, Spring, and Talendforge) and belongs to seventeen different projects. The actual effort is represented by story points. This dataset offers a corpus among its properties, however, it does not contain expert estimates.
- **Porru:** Porru's dataset is obtained from Porru et al. [178]'s study. It was also collected using the same method as JOSSE dataset, which is explained in Section 4.3. It consists of 4908 data points that are collected from eight open-source projects (Aptana Studio, Dnn Platform, Apache Mesos, Mule, Sonatype's Nexus, Titanium SDK/ CLI, Appcelerator Studio, Spring XD).
- **Desharnias:** Desharnais [93] collected nine numeric features of 81 projects in the late 1980s. While this dataset is old, recent studies have reported experiments using Desharnais [93]'s dataset, for instance [181]. The 81 projects belonged to a commercial software development organisation. Four of the 81 projects were removed since they are incomplete projects. The dataset was obtained from the PROMISE repository [179].

Dataset	# of Records	Effort Unit	Min (minute)	Max (minute)	Mean (minute)	Median (minute)	STD (minute)	Skewness	Kurtosis
Deep-SE	23313	SP	1	100	6	4	10	6	45.69
Porru	4682	SP	1	6765	5	3	99	68.1	4652.2
Desharnias	77	P/H	546	23940	4834	3542	4161	2.04	5.3
NASA93	93	P/M	8	8211	624	252	1130	4.26	23.1

Table 4.1: Distribution of effort in experiment datasets. Effort unit abbreviations stand for the following: P/I = person-minute, P/H = person-hour, P/M = person-month, and SP = story point.

Dataset	# of Projects	# of Records	# of Used Attributes	Has Corpus	Publish Year	Work Unit	# of Expert Estimates (%)
Deep-SE	16	23313	1	Yes	2019	Issue	0 (0%)
Porru	8	4682	1	Yes	2016	Issue	0 (0%)
Desharnias	N/A	77	4	No	1989	Project	0 (0%)
NASA93	N/A	93	21	No	2006	Project	0 (0%)

Table 4.2: Summary details of the four datasets that are publicly accessible

- NASA93: This dataset is a collection of 24 numeric features for 93 software projects belonging to NASA. The dataset features 15 COCOMO standard attributes among the feature set. It describes old software projects that took place in the 1970s and 1980s. The source of the dataset is the PROMISE repository [179].

As shown in Table 4.1, the effort costs are skewed to the right (more data points with larger costs) in all the datasets. According to the datasets' kurtosis, the datasets have heavier tails (more data points) than normally distributed datasets, indicating that effort estimates are widely distributed.

## 4.2 Software Effort Estimation Datasets Research Studies

Software effort estimation datasets can be based on several different units of work, including software project, issue, and story. A dataset is a collection of work units that are associated with related properties, such as number of code lines. Different research areas name dataset records differently. For example, Case-Based Reasoning (CBR) research calls them cases [182], while Machine Learning (ML) research calls them data points [180]. For the rest of this thesis, the number of work units in a given dataset will be referred to as data points, since part of the thesis will be an evaluation of ML methods.

While the focus of SEE research is to produce better software estimates, less attention has been paid to the development of datasets to support evaluation, as noted by several studies [183, 184]. In addition, the size of available datasets is quite small compared to the needs for, for example, machine-learning-based techniques. In fact, the average number of records (cases in a dataset) in the top three datasets used by ML-based research according to Wen et al. [82] and Ali and Gravino [83] surveys is around 290. Along with that, there are few studies that have investigated the available datasets in depth. The rich analysis illustrated in González-Ladrón-de-Guevara et al. [185]’s study about the ISBSG [176] dataset shows evidence that it is not the case that any dataset can fit any ML model. NASA’s datasets [102] were also investigated by Shepperd et al. [186]. They uncover some unwanted qualities of the datasets, including data discrepancy, incomplete data points, and lack of detail about dataset preprocessing and data origins.

Further, whilst many studies have proposed or enhanced ML-based estimation methods, only a small number of studies investigate the role of quality of datasets in the production of estimates [172, 187, 188, 189]. In turn, this tendency may be due to the lack of research on methods for assessing dataset suitability for effort estimation. However, [188] have demonstrated that effort prediction studies may be invalid if the quality of the dataset adopted for evaluation is not considered.

Of the available work in this area, Phannachitta et al. [189] focused on a single aspect of dataset quality, *inconsistency*. According to Phannachitta et al. [189], inconsistency in a dataset means the existence of data points that demonstrate similar features (independent variables), e.g. lines of code, with a significant difference between them in the dependant variable (effort estimate). Dataset inconsistency can be identified by tools such as TEAK [190] and FISi [191]. As an advancement, Phannachitta et al. [189] proposed *Filter-INC* to double-check data points that are identified as inconsistent by TEAK [190] and FISi [191]. *Filter-INC* considers those data points as inconsistent only if the inconsistency still persists after eliminating conflicting data points, and thus, *Filter-INC* reduced the data loss by using an inconsistency cleansing procedure.

Another single aspect of research is described by Kocaguneli et al. [192]’s study, which evaluated datasets’ *relevancy*. Relevancy is a critical aspect, especially if the data points are collected from different environments. Kocaguneli et al. [192] stated that an estimate for a new case should be limited to an identified number ( $k - values$ ) of similar projects (analogies), and similarity is measured by the Euclidean distance of analogous features given the features of the new case. By doing so, data points that are imported from other environments can be used safely along with current local data points.

Further, more holistic frameworks assess multiple aspects of quality in SEE datasets. In particular, Bardsiri et al. [187] suggested several statistical tests such as correlation between

dataset variables and data distribution to evaluate the dataset fitness. Recently, Bosu and Macdonell [172] assessed 13 popular datasets from different quality aspects such as dataset inconsistency. However, Bosu and Macdonell [172] did not apply their approach on datasets that contain a text corpus among their features, such as the Agile story points dataset.

Bosu and MacDonell [193]’s dataset quality taxonomy offers insights into dealing with quality aspects of datasets. Specifically, the taxonomy addressed eleven quality issues grouped into three categories: accuracy, relevance, and provenance. Inspired by the taxonomy, the thesis developed five refinement procedures for the JOSSE context, e.g. assessing a text corpus attribute in the JOSSE dataset.

For this thesis, two datasets were collected: JOSSE and Planning Poker Industry (PPI) datasets. The following section will explain JOSSE and its collection process in detail. However, PPI can not be disclosed since it was collected from a commercial company and further summarizing details about the PPI dataset will be provided in Section 5.2.1.

### 4.3 Collection of the JOSSE Dataset

At the time of starting the thesis, early 2017, Ortu et al. [126] had proposed a large and general dataset collected from the JIRA issue tracker system for several open-source software projects. Ortu et al. [126]’s method of mining JIRA was inspiring. They selected open-source projects that used JIRA as their issue tracker system, and then extracted the issues’ populated attributes (18 attributes) along with relevant objects including users, comments, and attachments. That resulted in a large amount of data which is stored in an SQL database. Issue attributes that concern this thesis, e.g. actual cost and expert estimates, were not among the extracted attributes in Ortu et al. [126].

Expert estimates are the estimates that were predicted by the software development team or a member of the team working on the task. They are called *expert estimates* because the development team are the most knowledgeable individuals about the software project. They are also aware of their expertise, and thus, they are best placed to predict the most reliable estimates for a given software development task in their software project.

Using a similar method, issues that are annotated with an expert estimate and actual cost were selected to be the raw data for the proposed dataset. To produced an SEE-focused dataset, data objects that surround software issues such as attachments were not included, and the only extracted attributes were: issue ID, title, description, actual cost, estimated cost, number of comments and number of change logs. The dataset was named the JIRA Open Source Software Efforts (JOSSE) dataset.

The JOSSE data has been collected from three open-source communities: Apache, JBoss,

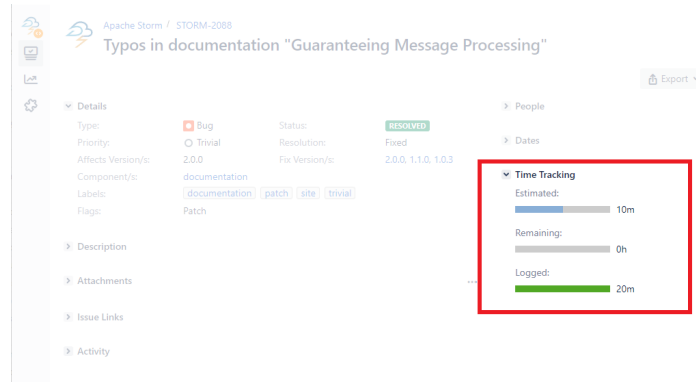


Figure 4.1: A screenshot of JIRA issue tracking system. Inside the red box are details of the time-tracking information for an issue.

Case	
PK	id
	corpus
	features
	expert_estimated_effort
	actual_effort
	reference

Figure 4.2: JOSSE Datasets consist of one table called *Case*. All the fields are text except *expert\_estimated\_effort* and *actual\_effort* which are numeric.

and Spring. Two criteria are used to find relevant issues, these are Spent Time and Status. Spent time is referred to as Logged Time and it represents the actual effort spent on finishing an issue work. Person-hour is the unit that is used in spent time; see Figure 4.1 for an example of how JIRA system presents time tracking information.

The dataset consists of 23,184 issues that are annotated with actual effort, and 4,327 issues that are annotated with estimated effort and actual effort. All the issues have a text corpus that is produced by combining the issue title with its description. For every issue, the actual effort and issue features are provided. Estimated effort is also extracted for those issues that are annotated with such information. Since those estimates were produced by the development team, they named as expert estimates in the JOSSE dataset. Issue features consist of the number of comments and number of activities for the issue. The number of issue activities is extracted from the issue log. It represents the sum of all the events that happened for a given issue. The original issue key is used as an identifier in the dataset. Finally, for traceability, each issue is supplied with a reference link that refers to the issue web page. The dataset consists of issues belonging to different projects from each community. Table 4.4 and Table 4.3 present a summary overview of the dataset, and Figure 4.2 shows the single table (“case”) in an SQLite database where the dataset is stored.

Project	# of Records	Min (minute)	Max (minute)	Mean (minute)	Median (minute)	STD (minute)	Skewness	Kurtosis
ACCUMULO	1118	3	120	34	30	25	1.44	1.63
AEROGear	205	20	1963	668	480	496	0.86	-0.06
AMBARI	1702	10	180	62	50	38	1.28	0.98
ARROW	1659	10	230	64	50	50	1.36	1.21
ARTEMIS	90	10	440	104	50	112	1.54	1.5
BATCH	324	2	480	123	60	129	1.52	1.51
BEAM	1366	10	450	124	90	102	1.3	0.98
CALCITE	137	10	230	70	40	60	1.13	0.15
CARBONDATA	1721	10	590	170	130	130	1.15	0.72
DAFFODIL	176	2	74	19	13	18	1.23	0.63
EXOJCR	709	4	2640	687	480	648	1.29	0.89
FLINK	672	10	30	18	20	5	-0.16	-0.18
GEODE	1269	10	140	44	30	30	1.33	1.04
GTNPORTAL	206	10	495	152	120	129	1.1	0.29
HDDS	158	10	220	67	50	43	1.38	1.57
IGNITE	513	10	41	21	20	8	1.06	1.43
INT	413	1.5	480	129	90	118	1.63	2.17
JBAS	99	10	1920	450	360	447	1.51	1.92
JBEAP	136	5	300	61	20	75	1.82	2.62
JBESB	124	2	962	235	120	268	1.55	1.65
JBFORUMS	109	15	1050	251	150	243	1.4	1.51
JBLAB	230	10	2460	554	330	581	1.56	1.78
JBPORTAL	150	5	2100	374	180	471	1.9	2.84
JBTM	193	2	1080	234	120	277	1.55	1.62
METRON	106	10	290	92	70	69	0.97	-0.05
MNG	144	5	360	75	45	77	1.65	2.42
MXNET	304	10	590	140	85	135	1.36	1.2
NETBEANS	293	10	200	61	40	45	1.18	0.47
NIFI	175	10	150	51	40	32	1.22	0.82
RF	280	15	990	369	242	257	1.02	0.32
SLING	109	10	120	40	30	27	1.23	0.52
SPR	180	2	269	61	34	64	1.48	1.5
STDCXX	198	5	420	119	120	83	1.47	1.97
STORM	839	10	230	69	50	51	1.21	0.83
STS	247	4	780	199	138	170	1.33	1.34
SWS	89	2	156	32	17	36	1.58	1.85
TS	322	10	250	89	70	53	1.15	0.75
ZOOKEEPER	214	10	460	119	80	106	1.26	0.99
Total	16979	1.5	2640	136	60	248	5.06	33

Table 4.3: Distribution of JOSSE dataset

Project	Expert Estimates		Outliers		Dissent Records		Project Domain
	#	%	#	%	#	%	
ACCUMULO	6	0.5%	149	13.3%	2.7	2.7%	Database Software
AEROGEAR	185	90.2%	9	4.4%	2	2.0%	Mobile Development
AMBARI	41	2.4%	123	7.2%	2	2.0%	Data Processing software
ARROW	18	1.1%	174	10.5%	2.5	2.5%	Data Processing software
ARTEMIS	0	0.0%	9	10.0%	0.8	0.8%	Enterprise system
BATCH	290	89.5%	33	10.2%	1.3	1.3%	Software development
BEAM	15	1.1%	112	8.2%	1.9	1.9%	Data Processing software
CALCITE	2	1.5%	7	5.1%	1.5	1.5%	Database Software
CARBONDATA	15	0.9%	138	8.0%	2	2.0%	Database Software
DAFFODIL	0	0.0%	15	8.5%	1.2	1.2%	Data Processing software
EXOJCR	489	69.0%	52	7.3%	1.9	1.9%	Software development
FLINK	7	1.0%	47	7.0%	1.8	1.8%	Data Processing software
GEODE	0	0.0%	119	9.4%	1.9	1.9%	Data Processing software
GTNPORTAL	166	80.6%	14	6.8%	1.5	1.5%	Web Development
HHDS	0	0.0%	22	13.9%	1.3	1.3%	Data Processing software
IGNITE	2	0.4%	110	21.4%	2.3	2.3%	Database Software
INT	385	93.2%	47	11.4%	1.6	1.6%	Software development
JBAS	33	33.3%	6	6.1%	2	2.0%	Enterprise system
JBEAP	4	2.9%	24	17.6%	3.8	3.8%	Enterprise system
JBESB	5	4.0%	25	20.2%	0.7	0.7%	Enterprise system
JBFORUMS	0	0.0%	14	12.8%	4.2	4.2%	Web Development
JBLAB	45	19.6%	27	11.7%	1.8	1.8%	Communication Platform
JBPORTAL	25	16.7%	22	14.7%	1.3	1.3%	Web Development
JBTM	114	59.1%	24	12.4%	3.2	3.2%	Enterprise system
METRON	0	0.0%	5	4.7%	1.6	1.6%	Security Framework
MNG	113	78.5%	21	14.6%	3.8	3.8%	Software development
MXNET	0	0.0%	24	7.9%	1.4	1.4%	Data Processing software
NETBEANS	1	0.3%	24	8.2%	1.9	1.9%	Software development
NIFI	4	2.3%	15	8.6%	2.4	2.4%	Data Processing software
RF	236	84.3%	162	57.9%	1.3	1.3%	Software development
SLING	1	0.9%	5	4.6%	2.9	2.9%	Software development
SPR	9	5.0%	25	13.9%	2.3	2.3%	Software development
STDCXX	178	89.9%	28	14.1%	2	2.0%	Software development
STORM	4	0.5%	87	10.4%	1.9	1.9%	Data Processing software
STS	98	39.7%	35	14.2%	2.2	2.2%	Software development
SWS	5	5.6%	14	15.7%	3.2	3.2%	Cloud Computing
TS	0	0.0%	22	6.8%	2.2	2.2%	Web Server
ZOOKEEPER	6	2.8%	20	9.3%	1.8	1.8%	Database Software
Total	2502	14.7%	1809	10.7%	2	2.0%	

Table 4.4: Summary details of JOSSE dataset



The original version of JOSSE consists of all the collected issues (23,184) without any further refinement. The original version may need to be refined in order to serve the purpose that it is used for. The JOSSE dataset has been stored in a GitHub repository that is publicly accessible<sup>1</sup>. The repository contains all the necessary scripts to replicate and reproduce the dataset from its original raw data.

The next section will discuss several refinement options that can be applied to the dataset and report on the impact of the refinements in terms of dataset quality. However, it is up to the researchers who are planning to use the JOSSE dataset as part of their research to decide which refinement should be applied given the context of the dataset usage. For example, those doing experiments involving NLP techniques, such as BERT [194], may require a readability refinement, since BERT has been trained on a human language text corpus for a language such as English.

## 4.4 JOSSE Dataset Refinement Options

After the collection of the raw data points and depending on the research context, some data refinement options are necessary to eliminate any undesirable data points that may negatively impact the research outcomes.

Bosu and MacDonell used their taxonomy as a dataset quality assessment framework to benchmark thirteen popular SEE datasets [172]. Their goal is to evaluate how a given dataset fits for the purpose that it is intended to be used for. The taxonomy consists of three main categories: Accuracy, Relevance, and Provenance.

*Accuracy* is further divided into five elements that represent different data issues. If any of them exist, then the accuracy of that dataset might be compromised, and thus, it negatively impacts the dataset's fitness for modelling. The five data accuracy concerns are as the following:

- **Outliers:** data points that deviate from the distribution of a given dataset. Such data points result from irregular events.
- **Noise:** wrong data points that have slipped into the dataset.
- **Inconsistency:** the lack of data pattern and agreement. Usually happens when measures are interpreted in different meaning or contexts.
- **Incompleteness:** missing some data points or data point properties. It also refers to measures that represent unfinished components compared with other data points, e.g. effort of incomplete projects in a dataset of completed projects.

---

<sup>1</sup><https://github.com/crowd-planning-poker>

- Redundancy: happens in the case of duplicated data points. It also occurs when different predictor variables are correlated, called multicollinearity.

*Relevance* measures the quality of relationships between data points in a given dataset. There are three sub-elements beneath this category:

- Heterogeneity: the number of different environments where data points are collected. Environments here may refer to an organisation or, in large organisations, a single project.
- Amount of Data: Since ML-SEE relies on statistical methods, the amount of data is a significant player in the learning process and pattern deduction. It refers to the number of data points that will be used to train the ML model.
- Timeliness: data currency. Old datasets may train the wrong model for current software development tasks. For instance, a predictor variable, such as team experience, might have a strong impact on a predicted effort at the time when the data was collected. However, the same predictor (team experience) might not be as strong a predictor now as it was before due to technology changes.

*Provenance* is about trust in the dataset. In the literature, it relates to research replication and the dataset accessibility that facilitates such replication. There are three elements under provenance:

- Commercial Sensitivity: When a dataset is collected from a commercial environment, it might be sensitive to the organisation, and thus, researchers may not have permission to disclose such a dataset. Therefore, research replication is limited.
- Accessibility: the ability of researchers to access the datasets and algorithm scripts for the purpose of replication. Public repositories such as PROMISE [179] are important in providing enduring research accessibility.
- Trustworthiness: the evaluation of datasets and associated research proposals. The more extended and rigorous the evaluation, the more confidence it gains.

Inspired by Bosu and MacDonell's taxonomy, five refinement procedures were applied to enhance accuracy, relevance, and provenance. The procedures are: project-based quantification, outlier detection, assurance of data point cohesion, assessing corpus readability, and tracing data origin and reproducibility. The following subsections explain each phase and evaluate the impact on associated quality metrics.

Community	Projects with < 100 DP Count	%	Median	IQR	Skewens
RedHat	86	89%	5	19.25	1.91
Spring	21	81%	8	26	1.58
Apache	226	91%	4	10.5	2.70

Table 4.5: Distribution of JOSSE data points per project for each of the three open-source communities. DP stands for data point.

#### 4.4.1 Quantity of Data Points Per Project

After the initial collection and storage as a SQLite database, the dataset consists of 371 projects that belong to the three open-source communities. However, grouping data points based on their project, some projects have as few issues as one, and thus, it may be necessary to identify a minimum number of data points for each project. A summary of the statistics of projects with less than 100 data points is given in Table 4.5.

The data point quantities are not normally distributed, and thus, median and interquartile range are more representative as a data summary. There is a large number of projects with fewer than 100 data points per each community, with Spring having the lowest percentage of those projects (81%). The mean median of data points for those projects is 6 data points.

Whether those projects are removed will depend on the usage of the dataset. For example, if the goal is to train a classifier on cross-project issues, then there is no need to remove those data points since they are useful in that context. However, if the decision is made to remove all the data points that belong to projects with fewer than 100 data points, the remaining number of projects is 40 and the total number of data points is 18,943; an overall loss of 18.3% of the data points.

#### 4.4.2 Dataset Outliers

The dataset's outlier data points are detected using David and Tukey [195]'s method to identify data points outside the lower and upper fences. The outlier detection has been done on a project basis, i.e. data points are grouped based on their projects, then the outliers are identified. If outlier removal is required, two phases may need to be considered: project-based and individual-based removal.

The first phase identifies outliers as a percentage of the whole project data points. Any project with outlier data points that represent more than a certain threshold is removed. Then, an individual-based removal of data points should follow to purify projects that have a minor number of outliers. Such a removal scenario is based on an assumption that the outliers

represent a minority of the whole project data points, and if they are not a minority then the project may not be suitable to draw patterns from.

To illustrate an example of the removal scenario above, any project with more than a quarter of outliers is considered for removal. There are two projects (“CAMEL”, “SCB”) with 31.1% and 29.6% outlier percentages respectively, and the total number of data points belonging to both projects is 384. Continuing to the second removal phase (individual-based), there are a total of 1,482 data points identified as outliers. Table 4.4 shows the percentage of outliers for each project. The total number of remaining data points if the outlier removal scenario is considered is 16,979, representing a data loss of 10.4% from the complete dataset.

To visualise the difference before and after outlier removal, Figure 4.3 shows box plot charts of the dataset before and after outlier removal. The upper part of the figure (4.3a) shows a large number of outliers, as represented by the black dots outside the whisker range (David and Tukey [195]’s fences). On the other hand, a significant reduction of outliers is illustrated in the lower part. Some projects still have a few outliers shown in the box plot, which represent the outliers of the new distribution after the removal of the original outliers.

### 4.4.3 Dataset Dissension

As mentioned in several studies, effort inconsistency [189] and data inconsistency [172] referred to the same thing which is called data dissension here in this thesis. It could impact prediction accuracy if it is present. According to Bosu and Macdonell [172], inconsistency is a lack of data point harmony in terms of their property values. Phannachitta et al. [189] refers to inconsistency when the assumption that similar projects have similar efforts is violated. In other words, if data points have similar effort, they should exhibit similar property values (similar to each other).

While the term inconsistency has been used differently in the literature, this chapter uses data dissension to refer to how cohesive the data points belonging to a given project are. The challenge with the JOSSE dataset is that the main feature is a corpus, and measuring similarity between different corpuses must consider lexical similarity as well as semantic similarity. Thus, Phannachitta et al. [189]’s method may not suit JOSSE since it has been designed for datasets with numeric properties. Nevertheless, Phannachitta et al. [189] provided insights to inspect data dissension in the JOSSE dataset.

Text similarity is a concern of Natural Language Processing (NLP) research, and BERT[194] is among the state-of-the-art advancements in that field. Devlin et al. [194] proposed a pre-trained deep learning data model that can be used to encode a given text into word vectors. The lexical and semantic meaning of each word is embedded in the word vector. The BERT

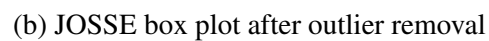
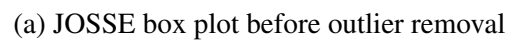


Figure 4.3: Two box plots representing the dataset before and after the outlier removal.

model has been trained on a large amount of documents, so one word may have different word vectors based on the context.

For the case of assessing the JOSSE dataset cohesion, issue corpuses were converted to BERT embeddings and then a cosine similarity between issue vectors was calculated. To illustrate the harmony between data points (issues), heat maps were created for each project, see Figure 4.4. Each heat map square represents a 25-issue random sample from each project. The dissension is represented in a colour range from yellow (100% similarity) to navy blue (83.5% similarity). Overall, the JOSSE dataset is cohesive, that is, the data point properties exhibit high similarity, except for a negligible number of data points per project (less than 5% dissension). From 1% to 5% of the data points were are not similar (have a BERT cosine similarity beyond David and Tukey [195]’s fences) to the rest of the data points. For instance, project ZOOKEEPER has 5% dissension, while BATCH has a 2.6% dissension; Table 4.4 gives the dissension percentage for each project in the dataset.

While dataset dissension identification is vital, different ML models have different tolerances of data dissension, and thus it is up to the ML model to include or exclude such data points. A data point being not similar to the rest of the data points does not necessarily invalidate it.

#### 4.4.4 Dataset Readability

Since the main property of the data points in the dataset is a corpus, it is necessary to examine the content of that corpus. Some ML models may make assumptions about the corpus, e.g. written in readable text and following a given language’s grammar. There are several readability assessment techniques, such as Flesch–Kincaid [196], however, they might not be useful for assessing a text corpus with grammar errors or code snippets that contains a stack trace.

The data point’s corpus was originally a combination of an issue description and its title. Most of the issue’s description contains a stack trace that is marked between snippet delimiters, e.g. “<code>”. However, there are many other descriptions that contain stack traces without the delimiters, which makes it difficult to separate such snippets from the descriptions. While some ML models are not necessarily impacted by language grammar errors, others, such as BERT, are sensitive to such errors since they are trained on grammar-free corpuses, and word position and form has consideration in BERT embeddings. Thus, the corpus for each data point needs to be assessed for language readability.

A grammar checker can be used to assess the corpus’s readability. The more grammar errors are in the corpus, the lower the corpus readability score. One way to measure readability using a grammar checker is to compare the number of grammar errors against the number of corpus words, and based on that, a percentage for the errors can be produced.

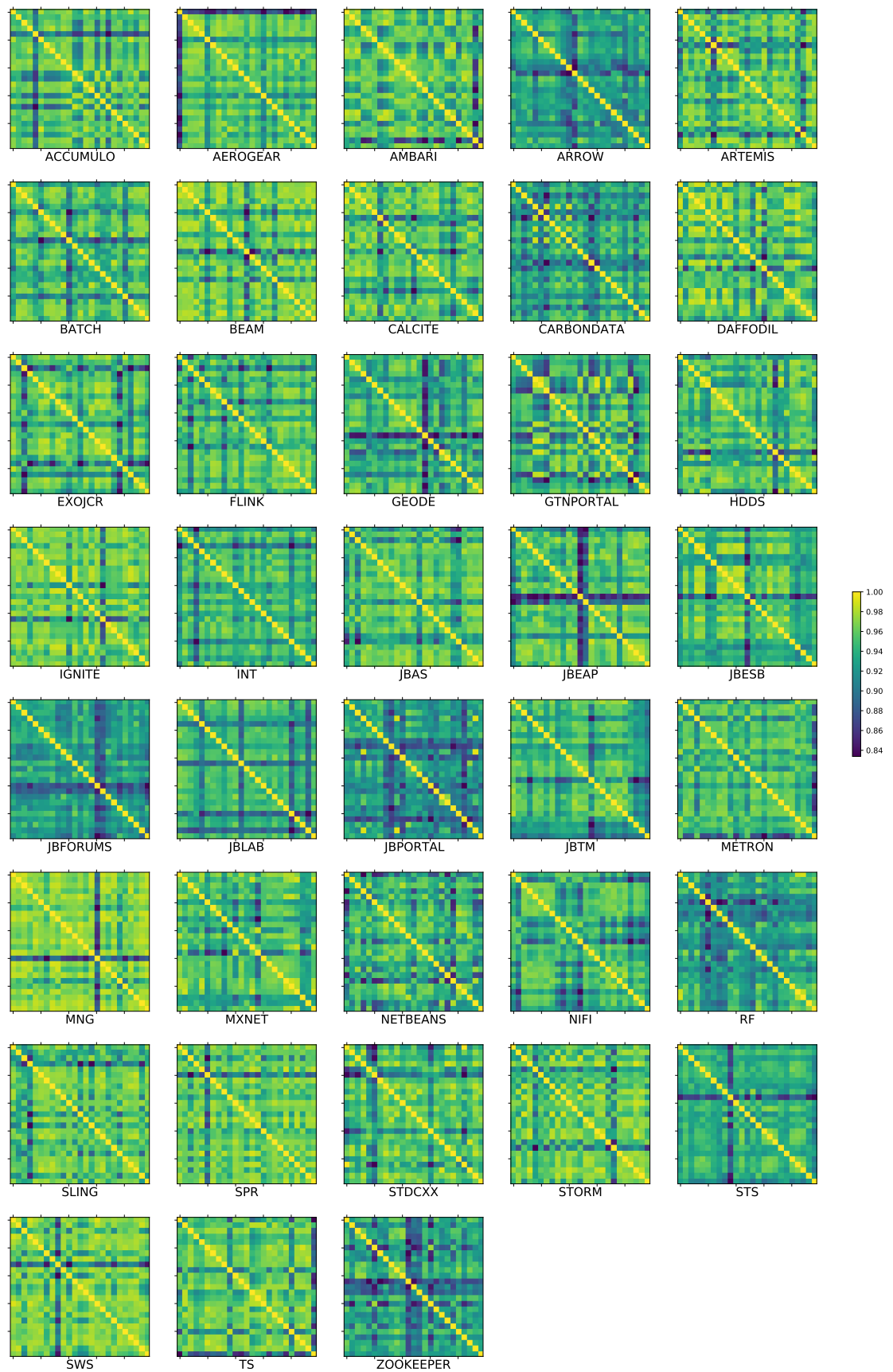


Figure 4.4: JOSSE dissension heat maps. Each map represents a sample of 25 data points that are selected randomly for each project.

Corpus	Mean	Median	STD	IQR
Has stack trace	46%	36%	43%	49%
No stack trace	15%	8%	23%	14%

Table 4.6: Statistics of corpus grammar error percentages for two kinds of corpus: one with stack trace and another without stack trace. STD stands for standard deviation and IQR stands for interquartile range.

The number of issues with a marked stack trace (between delimiters) in the JOSSE dataset is 2,631 data points, which represents 15% of the dataset. 85% of the data points may contain non-readable text such as code snippets that contain a stack trace. LanguageTool [197] is used as a grammar checker to evaluate two kinds of text corpus. The first type of corpus contains the stack traces of those issues that included a stack trace, and it does not contain the descriptions. The other kind of corpus contains the descriptions of all 2,631 issues, and no stack traces.

Table 4.6 illustrates some statistics after evaluating all 2,631 data points from the dataset. It shows that the corpuses with a stack trace only have a mean of 46% grammar errors compared to the number of words and a standard deviation of 43%, while the other kind of corpus (without a stack trace) has a skewed distribution; it has a median of 8% grammar errors and 14% as the interquartile range.

Based on that, if corpus readability is critical for a given ML model, then this procedure can be used to eliminate any data points that have grammar errors above a certain threshold.

#### 4.4.5 Discretising Software Effort Estimates

Taking inspiration from the Planning Poker method, software effort estimation can be discretised and grouped into categorical times. Estimate categories often adopt a metaphor that suggests increasing uncertainty with estimate magnitude. For example, Grenning [8] suggests using a Fibonacci sequence to indicate the margin of error between estimate sizes as they increase in magnitude. Cohn [5] states that approximate person-effort categories are more appropriate because it is often unrealistic to expect person-hour precision estimates to be accurate for software tasks. Therefore, classification can be used to build machine learning models instead of regression.

Further, Cohn [5] argues that teams eventually develop a tacit interpretation of the relationship between the relative categorical estimate and actual person-time costs, as the completed tasks are compared to the team's available person-hour budget over several sprints. Similarly, Menzies and Shepperd [198] asserts that data discretisation concentrates signals in datasets, which significantly enhances the ML model's performance. The discretisation is an essential



consideration, since the datasets that are adopted feature expert-based effort estimates.

It is necessary to employ categorical units similar to those used in datasets to train ML models to map continuous effort estimates. For example, if data collected from projects uses a Planning Poker method to report effort estimates, the Fibonacci series can be used to develop the categorical units.

Using a Planning Poker model as a strategy helps narrow the gap between expert-based and ML-based prediction, as Jørgensen et al. [2] concluded that such a combination is one step towards better estimation. The Planning Poker model inspires the research in designing a new discretisation category that avoids some drawbacks, such as discretisation noise [199], and brings in the concept of magnitude of error to discretisation. Thus, this discretisation is referred to as magnitude discretisation in this thesis.

#### 4.4.6 Dataset Domain and Origin

While the dataset is a collection of software engineering issues including software development, such as new feature implementation, and software maintenance, such as bug fixing, these issues belong to a wide variety of software projects. For instance, the ZOOKEEPER project is software to manage processes of distributed applications, while the SPR project is a Java programming framework.

Moreover, the collected issues belong to three open-source communities:

- Spring (<https://spring.io/>) is a Java programming framework.
- JBoss (<http://www.jboss.org/>) is an application server.
- Apache (<https://httpd.apache.org/>) is an HTTP server.

For each data point in the dataset, there is a reference link where further information about the data point can be acquired. Data point references also provide provenance and trustworthiness. The whole dataset can be reproduced from the reference links if necessary. For more details and to ensure data relevancy, Appendix A lists all the projects along with a brief overview about the project and its domain area.

#### 4.4.7 The Quality Taxonomy Assessment

Following the quality criteria of Bosu and Macdonell [172]’s taxonomy, Table 4.7 shows the JOSSE assessment against each criterion of the taxonomy. Three criteria have been adjusted to fit in the context of the JOSSE dataset, including noise and outliers. Noise has

	RedHat	Spring	Apache
<b>Dissension*</b>	2.15%	2.12%	1.99%
<b>Outliers*</b>	10%	12%	16%
<b>Amount of data</b>	3,953	1,808	17,427
<b>Timeliness</b> Year*	2004-2018	2007-2011	2004-2019
Timing Information	No	No	No
<b>Inconsistency</b>	No	No	No
<b>Incompleteness</b>	Yes	Yes	Yes
<b>Redundancy</b>	No	No	No
<b>Heterogeneity</b>	Yes	Yes	Yes
<b>Commercial Sensitivity</b>	No	No	No
<b>Accessibility</b>	Yes	Yes	Yes
<b>Provenance/ Trustworthiness</b>	Yes	Yes	Yes

Table 4.7: JOSSE evaluation against Bosu and Macdonell [172]’s quality taxonomy. Columns with \* are adjusted to fit in the JOSSE context. *No* implies that there were no traces of a given quality data/issue, e.g. inconsistency.

been replaced by data dissension, since the noise method that is used by Bosu and Macdonell [172] is not applicable to the JOSSE text corpus properties. In addition, data dissension is not necessarily a bad sign for the dataset, as described in Bosu and Macdonell [172]’s taxonomy when they discussed *noise*. Further details about data dissension and inconsistency, outliers, amount of data, accessibility, and provenance can be found in sections 4.4.3, 4.4.2, and 4.4.6 accordingly.

Timeliness has been adjusted to represent the year of data point creation rather than the dataset publication date. Incompleteness concerns the missing features from each data point. Redundancy refers to data point duplication; it concerns whether a dataset has duplicated data points. Heterogeneity of a dataset covers the data point environment and context; it concerns whether data points in a given dataset come from a single environment, e.g. one software development project, or multiple environments. Commercial Sensitivity concerns whether a dataset has sensitive information relating to a commercial organisation. For example, a dataset collected from a private software development project may reveal sensitive data about the project stakeholders that should not be published.

Interpreting the JOSSE assessment results listed in Table 4.7 in the context of Bosu and Macdonell [172]’s assessment of the 13 SEE datasets and using their approach, JOSSE has less noise, more data points, and is more recent. The average dissension (reported as noise in Bosu and Macdonell [172]’s assessment) of JOSSE is 2.1%, which equals to the lowest noise reported among the 13 datasets. The average noise of all the 13 datasets is 17%. On average, the JOSSE dataset has a slightly higher outlier percentage (12.6%) than the 13 datasets (9.6%). The median number of records in the 13 datasets is 62, with one dataset (ISBSG16)

that has 7518 records. The JOSSE dataset has 23,188 records. The 13 datasets have records about software projects, whereas the JOSSE dataset records are software development tasks, and this is a key difference between the datasets which explains the large number of JOSSE records. The records of the 13 datasets dated back to the year of 1989, and the most recent records were created in 2015. The JOSSE dataset records were created between 2004 and 2019.

Generally, JOSSE offers less noise, more granularity, more data, and more recent records of open-source software development activities compared with the 13 datasets evaluated by Bosu and Macdonell [172].

## 4.5 Summary

This chapter reflects upon recent research that concerns SEE datasets. It aims to establish an understanding of existing datasets from previous research and provide a new dataset that answers to the overall shortage in SEE dataset. The proposed dataset also features data properties that are not necessarily available in other datasets, such as expert estimates. Such a dataset is an important advancement, taking into account its role in the further chapters of the thesis.

Whilst the dataset is important, few studies pay attention to collecting and maintaining SEE datasets. In fact, the most popular datasets that are used in the recent ML SEE research are quite old. Therefore, this chapter analysed research datasets more closely and suggested the JOSSE dataset as a public dataset to be used in future studies of ML SEE. It also reviewed dataset quality studies, such as Bosu and Macdonell [172]’s comparative study of different datasets. Then, it reflected the learned lessons in the JOSSE dataset to ensure its quality and fitness to be used as training data for ML SEE models.

Then, this chapter drew a detailed step-by-step explanation of possible procedures. Five steps have been designed to ensure dataset accuracy, relevance, and provenance, as explained in Bosu and MacDonell [193]’s taxonomy. Now that a fresh dataset has been proposed and prepared for ML algorithm evaluation, the next chapter is an attempt to use JOSSE along with other datasets to evaluate the effectiveness of BERT encoding on the accuracy of ML SEE predictions.

## Chapter 5

# Evaluation of Language-Based Transfer Model for Software Effort Estimation

Machine Learning (ML) and parametric models have been used to predict software effort [1, 82]. Most of the studies have used numerical features, e.g. the number of source code lines [182]. Other than a small number of studies [180, 200], the literature lacks research into the use of a text corpus that describes the software task to predict effort.

This chapter contributes to evaluating the state-of-the-art natural language processing (NLP) language model Bidirectional Encoder Representations from Transformers (BERT). BERT has advanced the application of NLP dramatically by being able to consider the context of a given text [201]. The aim, in this chapter, is to determine whether an ML-based approach to effort estimation is reliable using the recent NLP advancements. This chapter uses JOSSE discretised effort, and treats the effort prediction as a classification problem. Therefore, unnecessary precision in effort prediction is avoided to gain more reliable predictions. In addition, treating effort estimation as a classification problem narrows the methodological gap between Planning Poker and ML methods in dealing with uncertainty.

The following section gives a brief background about ML and effort estimation. Since effort estimation is considered as a classification problem in this chapter, it is important to understand the background and the differences between the methods and concepts used, including feature extraction, training algorithms, and data models. Thus, this chapter starts with a background section explaining how features are extracted from a corpus-based input, i.e. software task description, and detailing the main steps in ML effort estimation. It also introduces the ML models that will be used in the experimentation work of this chapter. After that, Section 5.2 illustrates experimental design and research questions. It details the datasets, experimentation methods, and evaluation metrics used. Then, Section 5.3 states the

experiment results and answers the three research questions. After that, Section 5.4 explores the experimental findings and discusses them along with relevant literature. Finally, Section 5.5 summarises the chapter and draws insights about the next step of this thesis.

## 5.1 Background on Machine Learning For Corpus-Based Software Effort Estimation

Early approaches to software effort estimation were SEE formal methods including CO-COMO [1] and Case-Based Reasoning (CBR) [182]. Both ML and parametric methods are similar in the sense that both rely on statistical bases such as regression analysis. However, the key difference between them is the form and amount of input data.

ML algorithms can be used to build an implicit data model of associations between dependent and independent variables. According to Trendowicz and Jeffery [24], parametric methods predict software development effort based on defined relationships between independent variables such as project complexity and required effort. On the other hand, ML algorithms do not declare relationships. Instead, ML-based approaches assume the existence of some, potentially weak, correlation between given task attributes (independent variables) and task cost (dependent variable), and let the data model define them implicitly, and thus, ML-based approaches are more flexible. However, ML algorithms need to be trained on a large dataset of software development tasks that are similar to the ones that need to be estimated (the targeted case) in order to build their data models. In contrast, parametric methods have predefined models that take a limited amount of information about the targeted cases as inputs to predict new estimates.

Inspired by Planning Poker (expert-based estimation), where the estimation process starts by comprehending a description of a software development task, this chapter trains and builds an ML model based on features extracted from the same description text. Additionally, the chapter considers expert estimates as its comparison baseline. However, ML methods do not accept natural language as an input as humans do. Instead, the corpus needs to be transformed into a different format that can be processed by machines, perhaps a numerical representation of the corpus. The following subsection explains feature extraction and transformation techniques, and then it explains the learning algorithms that are used this chapter to build ML models. After that, it discusses how estimates can be better discretised to strengthen signals in the extracted features.

### 5.1.1 Converting Text Corpus for Use in ML Models

The goal of any ML task is to train a machine (creating a data model) on the association between data point features and its target using historical data. The model is then used to predict the target of a new data point instance.

As explained in Chapter 4, a data point represents a software development task in a given dataset. Each data point has attributes that can be classified as either independent variables (features) or a dependent variable (target). For the data points in this study, the independent variable is the description of a software development task, and the dependent variable is the task estimate. Sometimes, datasets have a baseline and ground truth targets, as in the JOSSE dataset. Those targets are used in training and evaluation processes respectively.

The first step in the experiment is to convert the dataset developed in the previous chapter such that the issue description (text corpus) can be used as input for a language-based transfer model. Feature extraction, therefore, is the process of extracting meaningful numeric values, called features, from a given corpus, for example, the frequency of a word in a given description. The input of the feature extraction process is the corpus, and its output is a numerical matrix of the extracted features. For each data point attribute, the features are extracted from the description text to form one vector representing that data point. This process is referred to as vectorisation. Then the vectors are grouped in the output matrix representing the whole corpus with one row for every data point.

The discipline of ML offers a diverse range of corpus transformation methods that run from a simple method such as Bag of Words to a more advanced and complex method that can capture the corpus context, such as BERT [194]. The method of using BERT word embeddings of a text corpus in a further classification task that produces a vector of each class's probability is called transferred learning. The rest of this section will give more details about the different transform methods that are used in the experiment.

**Bag of Words (BoW)** converts a corpus into a matrix of vectors of word frequencies, where each row represents a document, i.e. a data point text attribute, and the columns represent all the words in the corpus. Each cell in the matrix has a real number representing the number of occurrences of a word (the column) in a document (the row).

A BoW matrix weights corpus words in terms of frequency instead of importance. For example, common words like “is” and “the” will receive the highest weight, which is not preferable. The BoW approach is typically enhanced by giving more weight to important but uncommon words. The Term Frequency–Inverse Document Frequency (TF-IDF) uses a word's frequency to reflect its importance, assuming that the word's frequency-inverse across documents is a proxy to its importance. The outcome matrix consists of ratios from 0 to 1

representing the word's significance for a given document. However, TF-IDF increases the matrix sparseness, giving zeros to common words; thus, it may not be practical for a corpus with numerous common words.

BoW and TF-IDF lack other properties a word may carry, such as multiple meanings in homonyms and paragraph contexts. A word needs to be considered as a collection of features depending on its context. As Firth [202] said, summarising the underlying linguistic theory of Distributional Structure [203]:

“You shall know a word by the company it keeps!”

The collection of features a word represents in a language model is called word embedding, and it is generated using a model trained on a large text corpus. The following subsection will explain more about word embedding and its role in transfer learning in the discipline of Natural Language Processing (NLP).

**Transfer Learning (TL)** is the idea of using a model that is trained to solve one problem in another separate but relative problem. In the case of NLP, TL happened after realising that corpus transformation needs language understanding in the first place. Therefore, before solving an ML problem that includes a corpus, a machine should solve the language problem first. Thus, machines need to understand the language (have a language model).

A language model can be created by taking a large text corpus and extracting the association between the corpus words and sentences using machine learning algorithms such as an artificial neural network (ANN). For example, BERT has been trained with 3,300 million words from BooksCorpus and English Wikipedia [194]. By doing so, the model can take the context (surrounding words) of a given word into account and it can differentiate between one word in two different contexts.

In the context of feature extraction, a language model such as BERT takes a sentence (a sequence of words) as an input and transforms the sentence into word embeddings, a numerical matrix to represent the semantic of the sentence. Using these embeddings, it is possible to build another data model to solve a given problem, i.e. SEE, and thus, it is called transfer learning.

In some cases, the general language model, i.e. BERT, needs to be tailored to a specific domain, e.g. software development, and thus, it gets retrained.

**Fine-tuning** is the process of retraining a language model. Since language models are trained on a large general corpus, the model is not necessary to capture the domain-specific context in the thesis problem (SEE). Besides, fine-tuning helps change the language model

according to the new problem's targets, i.e. effort cost. Therefore, fine-tuning adds a domain-specific layer to the general model. It trains that layer using a domain-specific dataset, e.g. the JOSSE dataset.

### 5.1.2 BERT and RF as Machine Learning Models

According to ML surveys [82, 83], Random Forest (RF) and artificial neural network (ANN) classifiers fall under the top three classifier categories; hence, BERT is an ANN-based language model. Thus, they were selected for this thesis. Other classifiers, such as Support Vector Machines (SVM), were not included since their prediction performance is low. Additionally, few research studies have been devoted to SEE and RF, and thus, this chapter narrows the gap of detailed comparative research as explained by Nassif et al. [89]. Similarly, the area of applying contextualised text embedding to SEE has barely been investigated, and no study has evaluated BERT for SEE.

This section will briefly explain RF and BERT. Then, it will highlight recent research efforts that involve RF in contextualised text embedding in SEE. Since no studies have applied BERT, the following discussion will consider comparing RF and ANN.

#### Random Forest (RF)

Random Forest is used to assemble estimates from a group of decision tree (DT) models, hence the name forest. The forest is grown by training a decision tree on a random sample from a training set. The training is also done based on a partial set of features selected randomly for every DT. The goal of randomly sampling the training set is to avoid a common DT overfitting issue by reducing the forest's overall variance without impact on the bias side. Variance and bias are two sources of prediction error. The first happens when the noise data is modelled, causing an overfit issue. On the other hand, bias happens when a model does not include a legitimate relationship between the data points, causing an underfit issue. Figure 5.1 illustrates an example of an RF model to estimate effort.

#### Bidirectional Encoder Representations from Transformers (BERT)

The Bidirectional Encoder Representations from Transformers (BERT) is a language model based on the ANN structure. BERT is considered the state-of-the-art language model [194, 201]. It uses the Transformer, a deep learning model that can handle sequential data, e.g. words in a sentence, in parallel [204]. BERT has been trained on a large text corpus, including Wikipedia and Books Corpus. There are two sizes of BERT: BERT-base, with a neural network architecture that consists of 110 million parameters (edges); and BERT-large, with



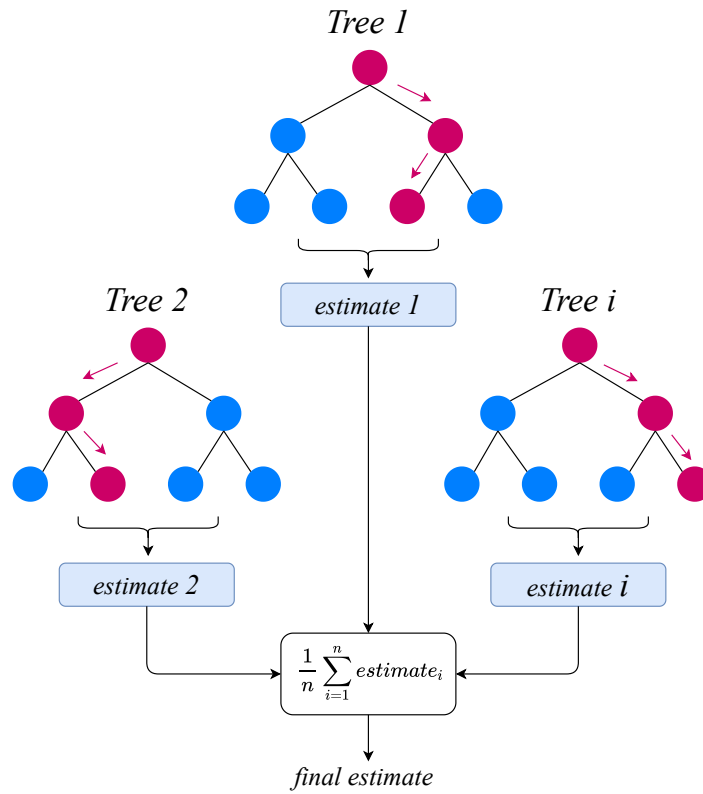


Figure 5.1: Simple Random Forest for Estimating Effort.

340 million parameters. As illustrated in Figure 5.2, the text sentences are split into tokens and then fed to an encoding stage to encode each token in terms of its language, position, and segment to result in token embeddings ( $E_{[CLS]}$  and  $E_n$ ). Then the output of the encoding stage is fed to the pre-trained transformer model that results in token vectors ( $C$  and  $T_n$ ). A BERT special token  $[CLS]$  starts each sentence and it aggregates all the tokens in that sentence as a  $C$  vector in the last layer of BERT (see Figure 5.2). The last stage depends on the ML task. In the thesis case, it is a classification task, and thus, the transformer output is fed to a feed-forward neural network classifier. The classifier output is the probabilities of the classes (time categories).

### RF Compared To ANN

Satapathy et al. [100] compared several ML algorithms for their argument about using use-case points as a software size parameter instead of function points. ANN and RF are among the compared algorithms. Using Satapathy et al. [100]’s sizing approach, RF outperforms ANN in their study. Another study [89] compared RF with multiple linear regression (MLR) and classical Decision Tree (DT). Nassif et al. [89] found that RF outperforms MLR and DT significantly on two different datasets. A similar RF comparative research was done by abdelali et al. [10]. Before comparing RF to DT, abdelali et al. investigated the optimality of

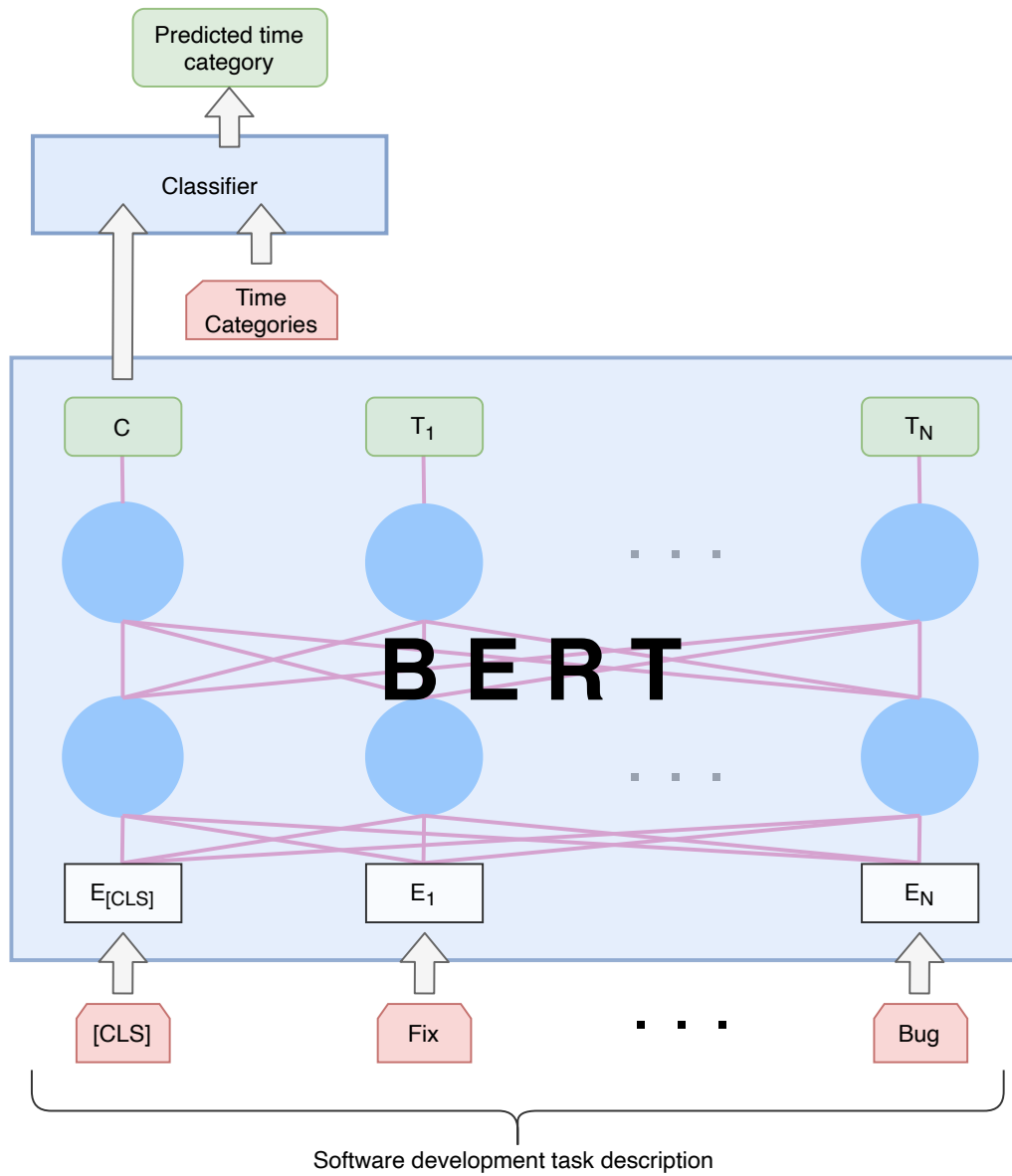


Figure 5.2: Fine-tuning BERT for the SEE problem and use of its aggregated sentence vector  $C$  along with time categories to build the classification model. The *Time Categories* represents the estimates boundaries as explained in Table 5.3. The *Classifier* is a linear feed-forward ANN.

RF parameters for three different datasets. They tune the RF model by varying the number of selected features ( $m_{try}$ ) and the number of trees to grow ( $n_{tree}$ ). The tuned RF outperforms DT, according to their study. While DT as an ML method category is quite popular in the literature [82, 83], only a few studies investigated RF, as explained above. Ali and Gravino [83] also noticed how scarce RF studies are, and they recommend doing more research to investigate various aspects of RF. However, several studies research RF outside the SEE area, such as [205].

### SEE and Contextualised Text Embeddings

The majority of ML SEE research has been evaluated using datasets that have no corpuses. A few studies involve text in their estimation process, such as Abrahamsson et al. [206]’s study. Among text-based SEE studies, only two studies consider a contextualised text-embedding representation, such as BERT [207, 208]. Ardimento and Mele [207]’s study focuses on finding the overall bug-fix time by utilising the text in description and developer comments. The study extracts the corpus from the Bugzilla issue tracking system and builds a data model to predict whether a bug resolution will be slow or fast. Fávero et al. [208] have used BERT to represent the bug’s corpus along with other features such as bug priority. Their experiment results in an effective time prediction. Fávero et al. [208]’s paper compares two pre-trained embedding text representations (Word2Vec and BERT). They used the DEEP-SE dataset Choetkiertikul et al. [180], and their experiment resulted in promising outcomes which suggest that using BERT as a contextualised text-embedding representation increases the prediction accuracy. Fávero et al. [208]’s study is built on an unverified assumption that pre-trained text representation is better than other feature extraction methods, e.g. TF-IDF. Moreover, two studies that use text-based features for SEE, while they do not consider the context, have sophisticated text embedding Ionescu [200], Choetkiertikul et al. [180]. Both studies suggest that text-based estimations offer a more profound linkage between the issues than numeric properties only.

## 5.2 Experiment Design

This section illustrates the experimental design to evaluate BERT linear and RF classifiers to predict the effort estimation category. Both classifiers are selected because they fall under the top three classifier categories mentioned by both ML surveys [82, 83]. In addition, the feed-forward ANN linear classifier is the default classifier that comes with the Transformers library [209]. The experiment also compares two feature extraction methods, BERT embeddings and TF-IDF vectorisation. Finally, it also evaluates prediction performance with expert

estimates (baseline) for those projects that have expert estimates. The experiment is designed to answer the following Research Questions about ML-based SEE methods (RQM):

- RQM1: How accurate are the selected ML models in predicting actual effort?
- RQM2: Is there a significant difference between BERT embeddings and TF-IDF vectorisation?
- RQM3: How comparable are the selected ML model predictions to expert estimates?

### 5.2.1 Experiment Datasets

The four datasets selected for the experiment are JIRA Open Source Software Effort (JOSSE), Planning Poker Industry (PPI), Deep-SE, and Porru. JOSSE and PPI have been collected as part of the thesis research, and Deep-SE and Porru are the only ones published in the literature that are accessible and task-based with corpus and logged effort estimates. In all the datasets, task description is the independent variable and estimate category is the dependent variable. Three datasets are drawn from open-source communities (JOSS, Deep-SE, and Porru), and one (PPI) is based on commercial projects. Tables 5.1 and 5.2 give summary information about the datasets and their properties. It lists the number of records inside each dataset (Deep-SE is the largest), and the minimum, maximum, mean, median, and standard deviation of the dependent variable (estimate) in both person-hours and story points.

The JOSSE dataset was collected from three open-source communities, including Apache, JBoss, and Spring. It consists of 16,979 issues annotated with actual effort, and 4327 issues are annotated with estimated effort and actual effort. Refer to Chapter 4 for full details. The Deep-SE dataset is derived from Choetkiertikul et al. [180]’s study. They collected their data in the same way as for JOSS, by mining JIRA systems. The data was collected from 9 open-source communities (Apache, Appcelerator, DuraSpace, Atlassian, Moodle, Lsstcorp, MuleSoft, Spring, and Talendforge) and belongs to 17 different projects. The actual effort is represented by story points. While this dataset offers different attributes, e.g. lines of code, only task description and logged effort were extracted. Porru’s dataset was collected using the same method as JOSSE and Deep-SE. It consists of 4908 data points collected from 8 open-source projects (Aptana Studio, Dnn Platform, Apache Mesos, Mule, Sonatype’s Nexus, Titanium SDK/ CLI, Appcelerator Studio, Spring XD) [178]. Finally, the PPI dataset was collected as part of the thesis research work. Since PPI was collected from a commercial development house, the company has not permitted the publication of the data since it contains sensitive information that may impact the company interest. Because of that, the next subsection is dedicated to giving more details about PPI.

Dataset	# of Records	Unit	Min	Max	Mean	Effort Median	STD	Skewness	Kurtosis
JOSSE	16979	P/I	2	2640	136	60	248	5.06	33
PPI	282	P/I	5	1560	431	240	416	1.09	0.17
Deep-SE	23313	SP	1	100	6	4	10	6	45.69
Porru	4682	SP	1	6765	5	3	99	68.1	4652.2

Table 5.1: Distribution of effort in experiment datasets. Effort unit abbreviations stand for the following: P/I = person-minute and SP = story point. STD stands for standard deviation.

Dataset	# of Projects	# of Records	# of Used Attributes	Has Corpus	Publish Year	# of Expert Estimates (%)
JOSSE	38	16979		1 yes	N/A	2502 (14.7%)
PPI	N/A	272		1 yes	N/A	282 (100%)
Deep-SE	16	23313		1 yes	2019	0 (0%)
Porru	8	4682		1 yes	2016	0 (0%)

Table 5.2: Summary details of datasets.

The researcher got an opportunity to collaborate with an industry partner to observe how an Agile team plays Planning Poker for seven weeks. Among the research activities, the researcher was able to collect data about the software development issues of a commercial software product. Therefore, to distinguish this dataset, it was named the Planning Poker Industry (PPI) dataset.

The PPI data was collected from a commercial in-house software development team that develops a web application software as a service. The company works in the tourism industry, and the data was collected from the company issue tracker system (Trello). The issues were classified into different smaller projects or sprints. Two criteria were used to find relevant issues: the actual time spent on the issue and issue status. The team annotated the spent time and the estimated time in the issue's title using square brackets. The data collection took place during an observation period of the team practice of Planning Poker.

The dataset consists of 282 issues that are annotated with actual effort and estimated effort. All the issues have a corpus that is produced by combining the issue title with its description. For every issue, the actual effort and issue features are provided. Expert-estimated effort was also extracted. The issue features consist of a number of comments and a number of activities on the issue. The number of issue activities was extracted from the issue log. It represents a sum of all the events that happened for a given issue. Moreover, the original issue key is used as an identifier in the dataset. Tables 5.2 and 5.1 present a summary overview of the dataset.

While the data points (issues) have been classified into different categories, they all concern

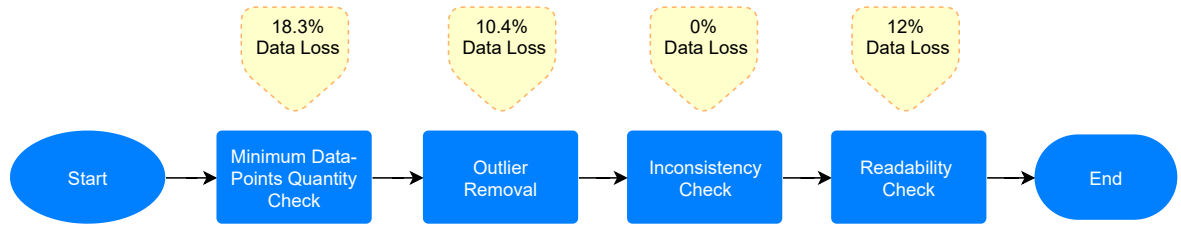


Figure 5.3: Dataset Refinement Process of both datasets (JOSSE and PPI). More details about each process step are explained in Chapter 4.

Category	Low	Middle	High
One hour	0	1	1
Half a day	2	4	5
A day	6	8	10
Half a week	11	20	30
A week	31	40	60
Two weeks	61	80	120

Table 5.3: Adopted time-based categories and their boundaries for a software task effort estimate. The numeric values represent the number of hours.

the same software, and thus they are treated as one group consisting of 282 data points.

**Dataset Refinement** The datasets have gone through all the refinement options explained in Chapter 4. As illustrated in Figure 5.3, there are four refinement stages, including Data Points Quantity Check, Outlier Removal, Inconsistency Check and Readability Check. In addition, Chapter 4 explains data discretisation and readability as additional data processing procedures.

All the datasets are annotated with the actual effort the software development task took. However, datasets use different scales and units. For example, JOSSE is annotated in person-seconds, whereas Deep-SE is annotated in story points. The actual effort is transformed into time categories with a magnitude scale based on the Fibonacci series. Therefore, the experiment’s dependent variable (effort) is discretised as detailed in Chapter 4. The person-second and person-hour costs reported on the issues were translated into approximate person-day and person-week categories, labelled as one hour, half a day, one day, half a week, one week, two weeks, and more than two weeks. The translation followed the same scheme as in the community issue tracker system (JIRA), where a working day is equal to 8 hours and a working week equal to 40 hours. This enabled a comparison between predicted estimates and the person-second or story point costs reported on the issues (Table 5.3). To draw boundaries between the scale categories, a relative midpoint between the two categories was selected. Table 5.3 illustrates the low, middle, and high possible person-hours for each category.

The details of the refinement of the JOSSE database were given in Chapter 4. For the PPI

dataset, after the initial collection and storage as a SQLite database, David and Tukey [195]’s method was used to identify the outliers as the data points outside the lower and upper fences. A total of 10 data points (3.5%) were identified as outliers and removed. The total number of remaining data points after removing the outliers was 272.

A text cosine similarity on BERT [35] was applied to the dataset, and the data points of PPI were found to be consistent; that is, data point features, i.e. the corpus, exhibit high similarity except for a negligible number of data points (1.7%).

### 5.2.2 Estimation Method

The method used to estimate effort using an ML model has two phases: data preprocessing and model training. The data preprocessing phase is essential for a dataset that has a corpus. During the preprocessing phase, features used to train the ML models are extracted from each issue’s corpus, as explained in Section 5.1.1. This experiment uses two feature extraction methods, BERT and TF-IDF, as illustrated in Figure 5.4.

Using the BERT language model as a feature extraction method starts by splitting the text into single words and replacing each word with its corresponding BERT token using the BERT dictionary. Then, the BERT-base [35] pre-train model is used to extract corresponding embeddings of the tokenised corpus as fixed-length vectors. These vectors represent the lexical and semantic meaning of a given word in its context. In this experiment, the embedding process considered only the first 400 words of a given corpus due to limited computing resources. The March 11th, 2020 version of the BERT-base model was used for the encoding.

TF-IDF was used as an alternative feature extraction method. BoW produces a vector with a length equal to the total number of distinct words in a given corpus. Then, each digit in the vector reflects how many times the corresponding word occurs in the corpus. The count matrix was normalised using term-frequency times inverse document-frequency (TF-IDF) representation to avoid the dominant effect of popular words.

After producing BERT embeddings and TF-IDF normalised vectors for the data points, the vectors were sent to the training phase. The training was done on two ML classifiers (BERT linear and RF). The BERT linear classifier is a single layer of a Feed-Forward Artificial Neural Network (FFANN) on top of BERT. Data points were divided into testing and training subsets using K-Fold Cross-Validation (CV). The training and testing were done on a project basis, which means that datasets with multiple projects, such as JOSSE, were divided into several subsets based on their projects. The Scikit-learn [210] implementation of RF and the McCormick [211] implementation of BERT and its classifier were used to run this experiment. Both models used the default configuration of their original authors [210, 211]. A

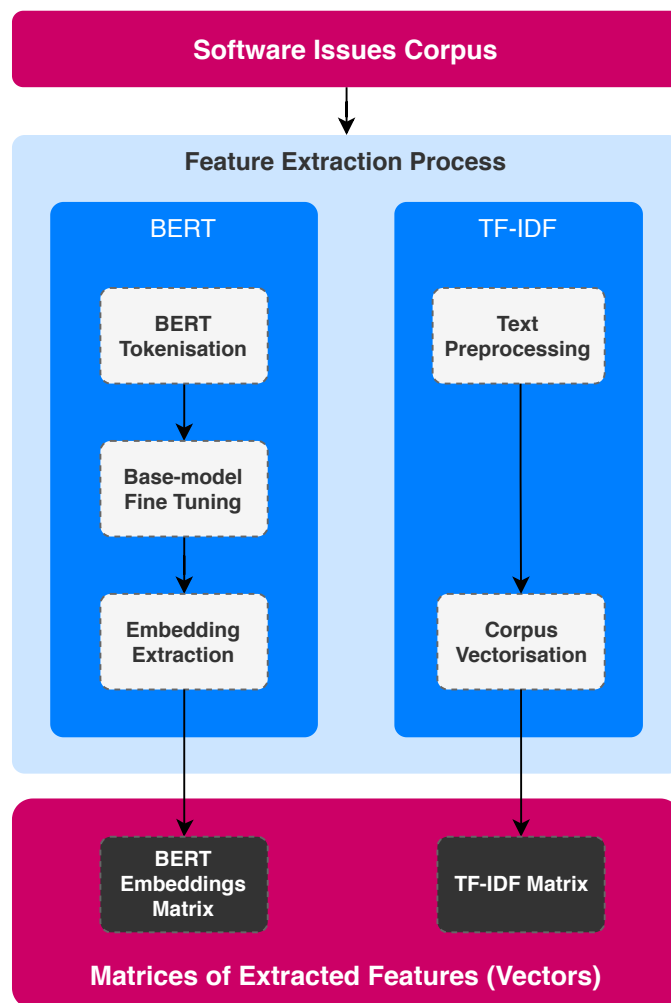


Figure 5.4: BERT and TF-IDF Feature Extraction Methods



replication pack of the actual code along with the datasets is publicly available at a GitHub repository<sup>1</sup>.

### 5.2.3 Evaluation Metrics

Usually, SEE studies use error measures, e.g. Median Magnitude of Relative Error (MdMRE), as explained in Ali and Gravino [83]’s recent survey. Error measures are used as a proxy to performance for a regression ML problem. However, in this thesis, the ML task is a classification problem since the data has been discretised, as explained in the previous section (5.2.1). Thus, accuracy is used as a performance measure of the models.

Two accuracy measures are reported, the Area Under the Curve Receiver Operating Characteristic (AUC-ROC) [212] and F-score [213]. Both metrics are less impacted by imbalanced data points, which happens to be the case for selected datasets.

Those metrics are calculated during model performance testing. The testing method used is K-Fold Cross-Validation (CV) [214]; a five-fold CV is implemented. In each fold, four-fifths of the data is used for training and one fifth is used for testing. Statistical tests of their significance are carried out using Kruskal-Wallis tests and ANOVA for AUC-ROC and F-score, respectively, at a significance level of 0.05.

## 5.3 Results

This section details the results of predicting estimates using FFANN and RF classifiers. The first part gives the prediction performance measurements using one feature extraction method, BERT, across two classifiers, RF and FFANN, to assess the impact of the classifier on accuracy. The second part evaluates feature extraction methods by using an RF classifier across two feature extraction methods. As stated earlier in the introduction, the aim is to examine BERT as a transfer learning model in the SEE problem, and thus BERT is compared with TF-IDF using the same classifier. Finally, to put the results in context, the third part compares ML models with expert estimates to identify more meaningful aspects from an expert estimation point of view.

Table 5.4 shows the summarised results across the datasets. The performance measures of F-Score and AUC-ROC are aggregated by averaging across all projects in the dataset. For more detailed results, Table B.1 in Appendix B lists the performance metrics based on individual projects for each dataset.

<sup>1</sup><https://github.com/crowd-planning-poker>

Dataset	ML Classifier	Feature Extraction	Projects	Folds	F-Score	AUC_ROC
Deep-SE	FFANN	BERT	7	68	0.434	0.633
	RF	BERT	7	68	0.351	0.613
	RF	TF-IDF	7	68	0.361	0.585
JOSSE	FFANN	BERT	35	167	0.680	0.561
	RF	BERT	35	167	0.612	0.551
	RF	TF-IDF	35	167	0.657	0.537
Porru	FFANN	BERT	4	17	0.404	0.571
	RF	BERT	4	17	0.314	0.556
	RF	TF-IDF	4	17	0.336	0.578
PPI	FFANN	BERT	1	5	0.502	0.618
	RF	BERT	1	5	0.388	0.604
	RF	TF-IDF	1	5	0.616	0.754

Table 5.4: Results of different models across different datasets. The results for JOSSE, Deep-SE, and Porru are aggregated from results of individual projects inside each dataset. Mean is used as the aggregation function.

### 5.3.1 RQM1: Accuracy of ML models

The results given in Table 5.4 suggest that using BERT for feature extraction and BERT’s linear classifier (FFANN) for classification is slightly better than the other options. Across all datasets, the FFANN-BERT combination achieved better F-Score and AUC-ROC results than other combinations (RF-BERT and RF-TF-IDF), except for the PPI dataset, for which RF-TF-IDF performed better.

However, we have investigated the performance on a project level using Kruskal–Wallis for AUC-ROC and ANOVA for F-Score. The Kruskal–Wallis test results in 2.92 with a p-value of 0.232 for AUC-ROC, and the ANOVA test results in 1.668 with a p-value of 0.192. Both tests for both metrics show no significant difference between the three combinations, and thus, the null hypothesis cannot be rejected. Both classifiers have similar accuracy performance using BERT and TF-IDF feature extraction methods.

To visualise the performance metrics, Figure 5.5 shows both AUC-ROC (5.5a) and F-Score (5.5b) for the three combinations using box plots. The figure illustrates the slight, but not significant, difference between them.

The AUC-ROC metric gives an overall accuracy measurement of a classifier for different classification probability thresholds. Its accuracy measurement focuses on classifier specificity. Classifier specificity measures how good the classifier is in identifying data points associated with a negative class. A classifier with 100% specificity means that it never misses a negative data point. Figure 5.5a shows that FFANN-BERT combination did the best. BERT as a feature extraction method also performed better than TF-IDF. Only two projects, PPI

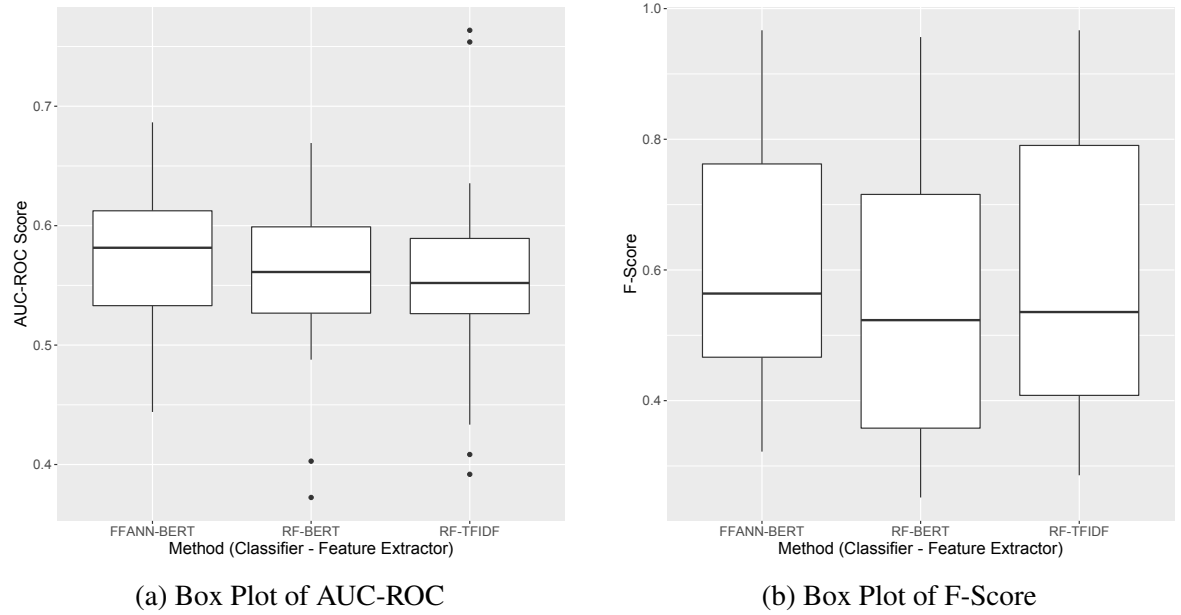


Figure 5.5: Box plot of performance metrics for different models/feature extraction methods.

and JBEAP (outliers), performed well using the RF-TF-IDF combination. While these are slight differences, there is no statistically significant difference.

On the other hand, F-Score metric gives an accuracy measurement of a classifier based on maximum classification probability (a single threshold). Unlike AUC-ROC, F-Score focuses on classifier precision. Classifier precision measures how good the classifier is in identifying data points with a positive class. A classifier with 100% precision means that it never misses a positive data point. Figure 5.5b shows that the FFANN-BERT and RF-TF-IDF combinations performed similarly. This time, BERT as a feature extraction method performed worse than TF-IDF. Although all the projects are software development projects, the noticeably wider IQR of F-Score indicates that projects still vary in the association between task descriptions and time logging.

While these are slight differences, there is no statistically significant difference. For a multi-class problem, as in the case of the SEE problem, a strategy is used which pits one class against the rest of the classes, and then the average of the classes' metrics is calculated as a summary metric.

### 5.3.2 RQM2: Evaluation of Feature Extraction Methods

To examine whether any of the classifiers had impact on the accuracy performance, a narrow comparison between two combinations of FFANN-BERT and RF-BERT was performed where the only feature extraction method used was BERT. Applying a statistical significance test of Kruskal–Wallis on the AUC-ROC scores of all projects, resulted in 1.1288 with a

p-value of 0.29. Similarly, applying ANOVA on F-Score resulted in 3.34 with a p-value of 0.071. Both tests on both metrics indicate that there is no significant difference, and thus the classifiers have no impact on the accuracy performance.

Next, a comparison between the two feature extraction methods using an RF classifier (RF-BERT and RF-TF-IDF) is performed. RF was selected since FFANN is built upon BERT and its implementation expects BERT-format input, whereas the RF implementation accepts both TF-IDF and BERT matrices. The aim is to see whether BERT embeddings built upon a large language model will result in a significant difference compared with a context-less feature extraction such as TF-IDF.

Applying the statistical significance test of Kruskal–Wallis on the AUC-ROC scores of all projects resulted in 0.505 with a p-value of 0.48. Similarly, applying ANOVA on F-Score resulted in 1.23 with a p-value of 0.27. Both tests on both metrics indicate that there is no significant difference, and thus BERT embeddings are not necessarily better than TF-IDF for the SEE problem.

### 5.3.3 RQM3: ML models compared with expert-based estimates

To put those performance metrics in context, the F-Score of the expert-based estimates reported in the JOSSE dataset was calculated. Table 5.5 shows the performance of the expert-based estimates for individual projects of JOSSE.

The average F-Score is 0.7, with the best performance being 0.812 and the worst being 0.54. Comparing these scores with the best performing ML model, expert estimates are better than the ML-based estimates. The ANOVA test of F-Score for those projects resulted in 4.685 with a p-value of 0.046, indicating a significant difference between expert and ML-model estimates.

Three projects, BATCH, EXOJCR, and INT, have a noticeably large number of issues (290, 489, 385 respectively). Two of them (BATCH and INT) achieved the best F-Scores (0.813 and 0.843). This might shed light on the practice of effort estimation in open-source projects. In addition, the percentage of issues annotated with expert estimates (see Section 4.3 in Chapter 4) in those projects from Table 4.4 in Chapter 4 is high (BATCH: 89.4% and 93.2%). This may indicate that those projects were taking effort estimation seriously, which may have helped achieve higher F-Scores.

In the same context, ten members of the communities were contacted to determine how the estimates were conducted. Only five of them responded and indicated that there was no particular procedure or instruction around effort estimations. One respondent stated that the development team tries to experiment with different SEE methods but always relying

Table 5.5: F-Score of expert estimates reported for several JOSSE projects.

Project	Number of Issues	F-Score
AEROGEAR	185	0.678
BATCH	290	0.813
EXOJCR	489	0.536
GTNPORTAL	166	0.713
INT	385	0.843
JBTM	114	0.554
MNG	113	0.723
RF	236	0.786
STDCXX	178	0.637
F-Score Mean		0.698

on their “gut feeling”. That respondent was a tester. The respondent takes the following considerations while estimating time for a software development task:

- The required learning about the development task and its deployment.
- Manual deployment.
- Reproducing the software issue and creating a fix.
- Automated deployment.
- Creation of test cases.
- Team members scheduled and holidays.
- Contingency time.

Further, the respondent commented that learning about the development task and its deployment may take a long time depending on the difficulty of the system environments.

Other respondents explained that they rely on their experience when estimating the issues, with or without a structured process for predicting such estimates.

## 5.4 Discussion

This section discusses the results from different perspectives and compares them with relevant literature. It also highlights threats to validity and proposes human-in-the-loop as a next step for advancing SEE.

As presented in the results, the lack of significant difference between both classifiers in accuracy performance is also reported by previous studies, such as the systematic literature review of Wen et al. [82]. According to the review, the average of the Mean Magnitude of Relative Error (MMRE) for DT-based models in 17 experiments was 55%, whereas it was 37% for ANN-based models reported in 39 experiments. While ANN-based models performed better, the difference is not significant as reported earlier in Section 5.3 in this chapter. Nonetheless, the reported metrics cannot be compared with those in this chapter, since the SEE problem in this chapter is a classification problem, whereas for the above survey it was a regression problem. However, the improvement delta reported in the survey agrees with what is reported in this chapter.

From a different but close perspective, ensemble methods using DT-based models ensemble significantly outperform ensemble methods using ANN-based models, as reported by Idri et al. [174]’s survey. The average of MMRE for DT-based ensemble in 5 experiments was 17.57%, whereas it was 48.32% for ANN-based ensemble reported in 16 experiments. Since Idri et al. [174]’s survey contradicts the trend reported by Wen et al. [82], that weakens the overall difference between the models.

Datasets are another factor, other than model algorithms, that may affect accuracy performance. Thus, the datasets included in this experiment were taken through a series of refinement steps, starting with ensuring an adequate number of data points for each project (at least 100 data points). Then, outlier data points were removed. Inconsistent or unreadable data points (i.e. with code or stack trace) were checked and removed (refer to Chapter 4 for more details). This helped to enhance BERT classifier accuracy performance by 16%. However, all performance measures are close to a random prediction measure. Theoretically, random predictions have a measure of 0.5 on both F-Score and AUC-ROC. The reported results of the datasets shown in Table 5.4 are close to random prediction, with RF-TF-IDF using the PPI dataset being the only exception.

Interestingly, TF-IDF shows slightly better performance using the RF classifier over FFANN, which is not expected, since FFANN relies on a state-of-the-art language model, i.e. BERT. This can indicate that language-based models do not necessarily offer the best feature extraction method for non-language applications, as in effort estimation.

Another reason behind BERT’s lack of performance is the technical nature of the language of the task descriptions. Although such models have been pre-trained on proper English content and fine-tuned using domain-specific datasets such as JOSSE, software issues may not be written in proper English, as explained in the previous chapter in Section 4.4.4. Technical writing may lack proper English grammar and sentence structure. Nonetheless, BERT embeddings achieved the best performance when used with BERT’s linear classifier (FFANN).

**Threats to Validity** A possible threat to validity is that the data are collected from open-source projects, where time control and project management are more relaxed compared with commercial projects. Thus, time logging for task effort and expert estimates might not follow a specific protocol or process, as explained in Question 3’s answer.

To mitigate this risk, a commercial-project dataset was collected and included in the comparison, where the development team followed a defined effort estimation and logging method, i.e. Planning Poker. In addition, the discretisation step mentioned in Section 4.4.5 is designed to minimise the risk in two ways. Firstly, it reduces the granularity of time to larger time buckets, and secondly, the efforts compared are in discrete, not absolute, values.

Changes to expert estimates in the JOSSE dataset after realising the actual effort is a threat that is mitigated by reviewing the activity log of software issues with expert estimates. None of them had a log entry indicating that the field *timeestimate* has been changed after updating the field *timespent*. *timeestimate* and *timespent* fields correspond to the expert estimate and actual time respectively.

Conclusions about accuracy performance can be threatened by the nature of the datasets. For instance, imbalanced datasets have artificial effects on some accuracy scores. To minimise such a threat, the measures F-score and AUC-ROC, which can handle imbalanced data, were selected. In addition, the statistical tests of Kruskal–Wallis and ANOVA were used to draw final conclusions.

Threats to external validity is also mitigated by incorporating issues from different large software projects, including both industrial and open-source projects. While the issues vary, they do not represent all kinds of software. For instance, critical system software projects, where the tightest time management is expected, are not included.

**Human-in-the-Loop for SEE** The comparison between expert and ML-model results confirms what was previously reported in the literature [2]. While the expert estimations are significantly better than ML, they are still unreliable and there is room for improvement. Perhaps by combining both ML and expert in one estimation method, each could strengthen the other.

While ML-based effort estimation studies are advancing, they may benefit from involving human judgement in the ML process. Effort estimation is not a trivial process, especially for intangible deliverables like software. Neither text-based features nor project characters are enough to build a reliable data model. Perhaps involving humans in the loop may help in comprehending estimation complexity.

## 5.5 Summary

This chapter reflects upon recent research in the area of ML SEE. It focuses on ML methods since they are the most researched methods in the recent literature [82]. ML methods are sensitive to the data that they are trained on, and thus datasets represent the other half of ML SEE research.

It draws a comparison between an ensemble model (RF) and a pre-trained language model (BERT) regarding their performance in an experimental study. It used four datasets, specifically, JOSSE, PPI, Deep-SE, and Porru. It also used expert estimates in the JOSSE dataset to compare ML models with experts.

The results suggest that there is no significant difference between the presented methods. However, BERT-BERT shows slightly better performance. On the other hand, the results show that expert and ML estimate performances are similar, with the experts' performance slightly better. Both findings confirmed what was already reported in the literature using different experimental settings [181, 2].

While ML SEE has received the most attention from the community, expert-based methods have been neglected [2]. Nevertheless, expert-based methods are the most popular method used by industry. Therefore, future research will be advocated in augmenting expert-based methods to be scalable and reliable.



## Chapter 6

# Crowd Planning Poker: A Preliminary Study

In the previous chapter, software effort estimates for four datasets were predicted using state-of-the-art machine learning (ML) and natural language processing (NLP) models. As demonstrated, the model prediction performance was not sufficiently reliable, especially when compared with expert estimates. Therefore, this chapter investigates whether human computation and crowdsourcing may offer a potential solution for scalable and reliable software effort estimation (SEE).

The work described in this chapter is inspired by an observation made by Grenning concerning the applicability of human computation and crowdsourcing to software effort estimation [215]. Grenning speculated whether an expert-based estimation method, i.e. Planning Poker, can be played using crowd workers to produce reliable and expert-comparable estimates. However, as far as the research is aware, no research was ever undertaken to investigate this possibility.

As a next step, this chapter details a series of pilot experiments to test the feasibility of applying human computation to Planning Poker, creating a new estimation method, Crowd Planning Poker (CPP). Five aspects of CPP will be explored: context and goals, participants, input and output, communication, and process. Each pilot is necessary in order to design and examine the adapted process of CPP. In addition, this chapter explores the optimal setting for conducting CPP. Therefore, the chapter's contribution is in being the first to offer insights about CPP that are supported by experimental evidence.

The chapter is structured as follows. The next section reviews considerations for deploying Planning Poker using human computation. Then, Section 6.2 describes the approach taken to implementing Crowd Planning Poker. After that, Section 6.3 illustrates the experimental design of the series of pilots. It lists the experiments and their aims, states the dataset used and the data selection method, defines the experimentation method, and explains how to evaluate

and test the results. Then, Section 6.4 details the results of the pilots with a subsection for each experiment. After, Section 6.5 discusses the takeaways from the experimental work, including the extra insights from crowds and the economic prospective of playing Planning Poker using human computation. The last section summarises this chapter and introduces the next step of the thesis.

## 6.1 General Considerations of Planning Poker

In this chapter, Planning Poker will refer to the original design of the process that is popular among Agile development communities. An Agile development team plays Planning Poker by calling for a meeting on a regular basis, e.g. every two weeks. The team uses a deck of Planning Poker cards, see Figure 3.2, and follows the process described in Chapter 3, Section 3.2.6. As discussed in Chapter 3, Planning Poker is a labour-intensive manual process. Speculating on how this limitation might be mitigated, in a review of Surowiecki [216]’s keynote talk on the Wisdom of the Crowds at Agile 2008, Grenning [215] wrote:

“I was wondering how Wisdom of Crowds would relate to people on agile teams doing estimation and planning. I was specifically interested in how his research applied to Planning Poker, a practice used throughout the world on agile teams”

The Wisdom of the Crowds is Surowiecki’s conjecture that a large crowd of non-experts, when suitably organised, can make a collective decision that is more reliable than a single or small group of experts. The intuition here is that individual idiosyncratic bias can be reduced when a group of people participate in collaborative decision making [217]. Therefore, Grenning’s proposition is that implementing Planning Poker within a crowd could result in more reliable estimates than those produced by small groups of experts.

Since Grenning [215]’s speculation, no research has investigated the problem of combining crowdsourcing and Planning Poker. However, Planning Poker bears a strong resemblance to Wideband Delphi [62], and a recent study by Kaivo-oja et al. [160] applied Delphi using human crowds for forecasting for Finnish companies. Kaivo-oja et al. [160] found the crowd-based Delphi better and more efficient than the conventional Delphi method. Flostrand [159] also studied the Delphi method in comparison with crowdsourcing as a method of future forecasting, and proposed a simple tool that can help executives note the differences and similarities of the methods. More widely, Chapter 3 presented a survey of studies on SEE, none of which discuss the application of effort estimation using a combination of crowd and Delphi or any of the Delphi family methods. Most of the studies discussed concepts related

to the theory of crowd wisdom in corporate management and collaboration context, without supportive empirical research.

However, Planning Poker was never meant to be played in a crowd environment, and thus several aspects must be considered before deploying Planning Poker in the new environment, i.e. crowd platforms. The rest of this section will review challenges concerning Planning Poker's context and goals, players, communications, input and output, and process. Then it will discuss characteristics that distinguish Planning Poker from other SEE methods that it would be desirable to imitate.

*Context and Goals* originate from the Agile development methodology. Reviewing the Planning Poker principles that are presented in Cohn [5]'s book, the Agile development context makes Planning Poker more than an estimation method. In fact, it is an opportunity to illustrate the first Agile value: "Individuals and interactions over processes and tools" [218]. Therefore, an Agile development team aims to leave a Planning Poker session with a working plan. The team goals of playing Planning Poker include effort estimation, task understanding, decomposition and refinement, prioritisation, responsibility assignment, and identification of side tasks, e.g. software installation [24]. Eliciting this richer range of qualitative data from a crowd of non-experts may not be realistic.

*Participants* in Planning Poker are mostly the development team members. Other stakeholders, such as clients, may take an observer role to provide relevant information outside the team's sight. Such participants are the most knowledgeable individuals about the matter, hence, they are referred to in this thesis as experts. Moreover, inaccurate estimates will impact on the performance of the team directly. The team participates in all Planning Poker stages and rounds, playing the game until they reach a final estimate. However, crowd workers do not incur long-term consequences from inaccurate estimates, or have an opportunity to learn from them in the context of a specific project. In addition, they may not participate in all estimation rounds because of communication challenges.

*Inputs and Outputs* of Planning Poker are only constrained by the knowledge and experience of the players. Planning Poker is played by the same software developers who work on the project, and thus, they know the background of the development project and the nature of the team velocity. As mentioned earlier, the team's aim is to come up with a working plan as an outcome. However, the limited time of the crowd workers in the micro-task crowdsourcing environment caps the amount of input and output data. Thus, the selection of background information type and amount is critical.

*Communication* between Planning Poker players happens synchronously as part of the estimation process, since they are all in a meeting setting. The players' communication includes verbal, visual, and body languages, which make the communication more fluent and easy. However, in a crowd environment, enabling a crowd worker to communicate in a syn-

chronous way is complicated [219] due to different backgrounds and the workers operating in different time zones. Therefore, conducting a communication-intensive process such as Planning Poker in a crowd environment is a challenge that needs to be addressed.

*The Process* of Planning Poker takes place in one go, at a team meeting where all participants play the game and estimate the effort simultaneously. If the game terminates before reaching the final estimate, the team is likely to restart the process from the beginning. As mentioned in the beginning of this section, a Planning Poker team iterates over a given software development task until reaching a consensus. They pass through sequential stages, starting by presenting the development task and ending by producing a final estimate. While Planning Poker's simultaneousness and sequentiality require it to happen in one go at a team meeting, micro-task crowdsourcing platforms can run Planning Poker over a longer time and in a series of events to accommodate certain limitations, such as communication synchronicity.

Further, there are several characteristics of Planning Poker that differentiate it from other SEE methods. These are important to imitate and keep in Crowd Planning Poker, as follows.

*Bias avoidance* is one feature of Planning Poker, and it is enabled by asking team members to predict their initial estimates secretly. This practice helps in reducing peer pressure and gives each team member a chance to have a time to come up with an estimate on their own. It also creates a gamification mechanism, and supports discovery of reasoning and relevant experiences as a justification rather than referring to other member's estimates as grounds. Therefore, the team will be in a better position to have a fruitful discussion when the estimates get revealed.

*Team comprehension* of a task at hand is shared between team members by discussing each estimate, especially the border estimates. It provides a wider range of possibilities for the team to consider. Such a discussion is vital in having an balanced team estimate when border estimates (optimistic and pessimistic) cancel or at least smooth each other. Moreover, it provides a framework for an active estimate. This means the final estimate is grounded in a shared rationale, has the team support, and, at least, an assigned team member.

*Consensus and aggregation* are considered more appropriate for finalising an estimate than an arithmetic equation, e.g. average. When the team reaches an agreement about an estimate, it means each team member has learned something from the discussion and adjusted their estimate accordingly. The adjustment originates out of understanding. There is also a chance that the team will not reach consensus. This can be useful for the team in identifying issues that require further refinement in order for a consensus estimate to be reached.

*Iterative estimation* processes provide assurance and evaluation of an estimate, which are important when dealing with high uncertainty levels. They also provide traces of changes in the team members' individual estimates, which can help to understand how the team came up with such an estimate, and thus Planning Poker can provide a traceable estimate that can

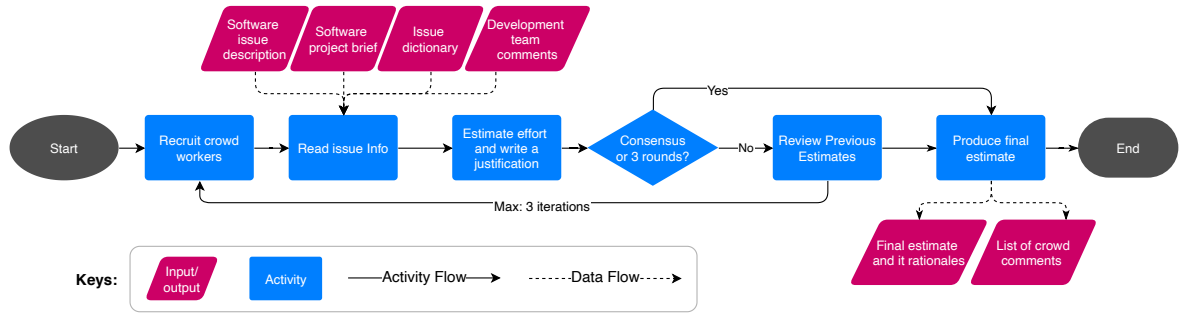


Figure 6.1: General model of the crowdsourcing Planning Poker task.

be tracked back to its origins.

In the light of these considerations, and having no prior literature on the topic, a decision was taken to conduct a preliminary study. There are several unknown aspects of the topic that need to be specified before investigating Planning Poker as an application of human computation, specifically, estimation process, amount and type of background information, and size of crowd team. Having considered these, the next step is to design a CPP process that has the capacity to imitate desirable Planning Poker characteristics in a crowd environment and address the raised challenges.

## 6.2 Crowd Planning Poker (CPP) General Model

Micro-task crowdsourcing environments offer quick and flexible access to human capital, i.e. crowd workers. At the same time, those environments demand a specific design for any crowdsourced job. Such a design needs to take advantages of the environments' flexibility and addresses their challenges. This section illustrates the proposed design of Planning Poker to be played by crowd workers in a micro-task crowdsourcing environment. It explains the design of a general model of Crowd Planning Poker (CPP) in a step-by-step process. Then, it iterates on the previous considerations identified earlier, showing how the design takes into account all aspects.

Figure 6.1 shows a flowchart of the CPP general model. In the first step, the crowd workers are provided with the core information describing a task (summary title and supplementary description). They may also have access to additional contextual information about the nature of the wider software project, for example, the programming language, software framework or technology platform used to develop the software, and the software developers' comments on the issue. In addition, workers may search for further information using a search engine query dialog.

In the second step, crowd workers are asked to provide an effort estimate for the described task. During this step, workers may also have the option of searching for publicly available

information that may inform their judgement. All the information was recorded for later analysis. Once the worker has reached a decision, they are asked to submit their estimate and a short justification to complete this iteration of the task. Each estimate received was manually evaluated by the experimenter for quality according to the procedure described in Section 6.2. Estimates that were determined to be invalid were removed from the experiment and not used further.

In the third step, the consensus achieved between the crowd workers is calculated. Each worker is then invited to perform a task to review a summary of the legitimate estimates provided by the rest of the crowd and then to submit their own estimation. The intention here is to mimic the consensus-forming behaviour in Planning Poker, by allowing workers to consider the boundary estimates provided by the crowd in the previous round.

*Context and Goals* in Planning Poker take Agile development as an environment; CPP can be played in micro-task crowdsourcing environments. In particular, it can be played in Amazon Mechanical Turk (AMT) as human intellectual tasks (HIT). Therefore, CPP focuses only on predicting software task estimates and supporting the estimates with a brief rationale. Given the environment of AMT, crowd tasks are supposed to be small and precise, and thus, limit CPP to just predicting an effort estimate. Unlike Planning Poker, the CPP goal is limited to effort estimation, and thus, additional goals of Planning Poker, including task understanding, decomposition, and refinements, are not pursued in CPP. From a design perspective, CPP is better understood as a Planning-Poker-inspired process to produce software effort estimates.

Figure 6.2 shows a sample of HITs. The first HIT is a CPP HIT posted by the researcher, and the next three items are other HITs from another requester. The CPP HIT is unfolded to show its specifications, including HIT description, allocated time, required qualifications, and rewards.

CPP is designed to enhance the scalability of Planning Poker, and thus, it can be played in large-scale projects that encompass large numbers of people, such as open-source projects. Large development teams in software development companies can also play CPP. Given that context, CPP narrows the type of software development tasks to those mentioned in open-source issue tracking systems. Open-source communities offer real-world data, and there are a couple of studies that utilise them as experiment datasets. For example, Qi et al. [220] used data from open-source projects that are hosted on GitHub. Qi et al. train a CART classifier to predict effort estimates based on the collected data. Their results (the predicted estimates) are comparable to those predicted based on existing datasets such as COCOMO [1]. Thus, Qi et al. conclude that open-source data can effectively provide estimates for new projects.

*Participants* in Planning Poker are the development team members, and thus, commitments and accountability can be considered. However, CPP participants are crowd workers recruited from AMT. Accountability and commitment are not options in such a context. Fur-

Requester	Title	HITs	Reward	Created	Actions
Mohammed Alhamed	Timing a software development issue	17	\$0.23	1m ago	Preview <a href="#">Qualify</a>
<p><b>Qualifications Required</b></p> <ul style="list-style-type: none"> <li>Previous Participant has not been granted</li> <li>HIT approval rate (%) is greater than 70</li> <li>Research Study Consent for University of Glasgow is 1</li> <li>Software Development Experience is 1</li> </ul>					
Crowdsurf Support	Timing review - Earn up to \$0.18 per timed media minute	6	\$0.18	16s ago	Preview <a href="#">Qualify</a>
Crowdsurf Support	Spanish: Timing review - \$0.18 per media minute	3	\$0.18	2d ago	Preview <a href="#">Qualify</a>
Crowdsurf Support	Special Handling - Entertainment Timing - Earn up to \$0.20 per media minute	1	\$0.20	1d ago	Preview <a href="#">Qualify</a>

Figure 6.2: Screenshot of AMT showing CPP HIT specifications

ther, holding a crowd worker to go over all CPP iterations might be challenging, and thus, the CPP process needs to accommodate such a limitation. CPP participants have no role in the software development and their commitment can not be elicited. CPP participants can provide estimates based on their experience, and so their qualifications are assessed. As shown in Figure 6.2, crowd workers need to satisfy the CPP qualifications, specifically, software development experience, research consent, HIT approval rate, and previous participation in CPP. Some qualifications, such as HIT approval rate, depend on workers' history, and others, such as software development experience, require workers to take a test in order to satisfy the qualification.

The CPP process collects a large number of estimates in order to produce a reliable estimate, and thus, it recruits a large number of workers. CPP participants are better considered as referrals (arrows) pointing towards the right estimate range. Given that analogy, it might be better to have more distinct crowd workers for a CPP estimation session, and playing only part of the game may become favourable. According to Moløkken-Østfold et al. [26], team diversity enhances estimation reliability.

*Inputs and Outcomes* of CPP are finite and pre-identified to accommodate the limitations of the micro-task crowdsourcing environment. CPP takes a limited number of inputs, specifically, software development issue description and title, software project brief, definitions of abbreviations and acronyms mentioned in the issue description, and comments of the development team on the issue. The goal of the project brief is to provide some context to the crowd workers, and the development team comments help in presenting the development team's view to the crowd workers during the estimation process.

When it comes to the outcomes, CPP offers an effort estimate of the software development

---

Please, select one of the estimates below:

<input type="radio"/> A day	<input type="radio"/> An hour or less
<input type="radio"/> Two weeks	<input type="radio"/> A week
<input type="radio"/> Half a week	<input type="radio"/> ⋮
<input type="radio"/> Half a day	<input type="radio"/> More time

Write your justification and comments:

Figure 6.3: Effort estimates available for crowd workers to select.

issue and perhaps an insightful rationale that supports the estimate. For the purpose of evaluating CPP estimates using Planning Poker, crowd workers are presented with a list of options of estimates, imitating the Planning Poker card deck. Figure 6.3 shows a screenshot of the estimates available for crowd workers.

The estimates' unit is time and they are grouped in categories approximately as for a Fibonacci series. The 'More time' option is similar to the Planning Poker card with the infinity symbol, and it indicates that the task cannot be completed or none of the available estimates are enough.

*Communication* in a micro-task crowdsourcing environment represents a challenge due to the limited time available to process the core requirements of a given task, and thus, crowd workers may find communicating with other workers is not an option for them. Therefore, the CPP design uses an asynchronous form of communication, and only very limited and carefully selected background information is communicated with crowd workers. The CPP design treats communication as a very scarce resource, and thus, it limits the number of communications to two and aggregates the information in concise, to-the-point messages.

Figure 6.4 illustrates how CPP communicates two messages asynchronously with crowd workers. The first message (annotated with number 1 in Figure 6.4) contains three elements: CPP instructions, software development issue description, and additional details about the issue. The design of both the CPP instructions and the software issue description reveal them in a sequential process to help the worker focus on one element at a time. Additional details are not part of the main sequential process. Instead, the worker branches out of the main process to look for additional information, which contains a concise description of the software project, terms and abbreviations in the task description, and development team



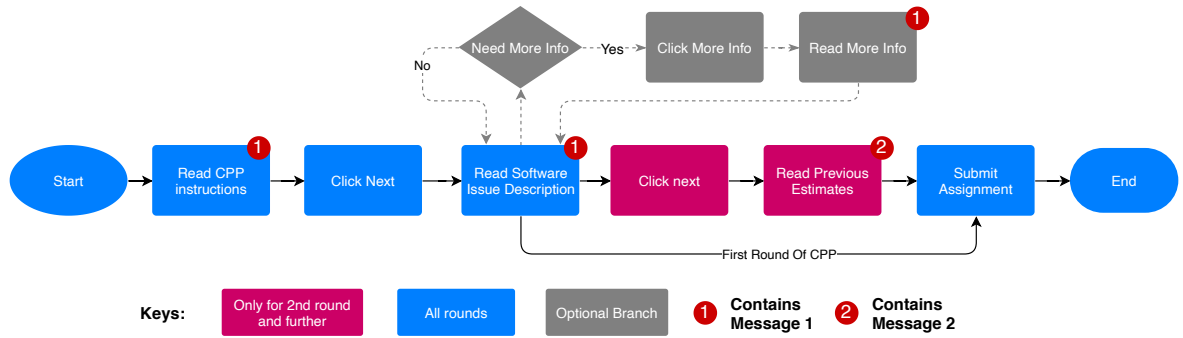


Figure 6.4: Asynchronous communication flow chart of two messages.

comments on the task.

The second message (annotated with number 2 in Figure 6.4) contains the crowd estimates from the previous CPP round along with their justifications. The estimates are aggregated using Max, Min and Median, and shown to the workers in the new round along with their justifications. Participants have no way to communicate with each other during a CPP round. Instead, they read the preceding worker's estimate and rationale from the previous round.

By following the flow chart in Figure 6.4, workers' estimates and their justification are revealed in the second round if the first round of CPP has finished and the consensus threshold has not yet been achieved. The aim of exchanging participants' estimates and justifications is to help achieve consensus between participants in the next round.

*Estimation Process* in CPP is asynchronous. The micro-task crowdsourcing environment and the nature of asynchronous communication in CPP may require longer time to complete the CPP rounds. However, it can be done at any time, without the limitation of team availability that exists in Planning Poker. Micro-task crowdsourcing marketplaces offer 24 hours 7 days access to human capital, and due to different time zones, crowd workers can take work at any time during the 24 hours. To take advantage of this feature, the iterative model of the CPP process mentioned earlier in this section is designed to be performed by crowd workers around the clock and as soon as the issue has been created.

*Bias avoidance* in CPP is considered. CPP collects the initial estimates without any prior estimate-related information. The aim is to produce an initial estimate without biases or hints, imitating the Planning Poker initial secret estimate. Such a design gives crowd workers a chance to retrospectively review their experience and solely predicate an estimate and justify it. Further, CPP collects the estimates from a much larger and diverse group of crowd estimators, which helps in reducing bias and bringing insights that may not cross the development team's thinking. Therefore, the crowd team of estimators is in a better position to produce a reliable estimate and insightful rationale that comes from a wide range of experiences. It is also necessary to mention that conflicts of interest are avoided in CPP, since the crowd team of estimators are not the same as the development team.

*Consensus and aggregation* in CPP takes two directions. Initially, CPP uses crowd consensus to produce the final estimate. The consensus is calculated using Fleiss' Kappa [221]. It is a measure that is used to assess the agreement reliability of raters. According to Landis and Koch [222]'s interpretation, a kappa in the range 21%–40% shows a *Fair agreement*, and that is where CPP stops iterating over an issue estimation. To help in bringing the crowd team of estimators to a consensus on an estimate, the previous estimates are presented in the next rounds.

Since CPP recruits a large number of crowd workers with a wide range of backgrounds, there is a chance that the team will not reach consensus. In the case of no consensus between the workers, alternative aggregation methods, specifically, averaging and median are used to produce the final estimates. The final estimate is also marked with a lower confidence level.

*Team comprehension* of the software development issue is shared by providing rationales from a wider range of backgrounds from the large number of crowd workers. While CPP cannot sync the development team's understanding as is the case in Planning Poker, it shows how the outside world thinks about the issue and its estimates.

*Iterative estimation* in the CPP process helps communicate the messages between crowd workers, and it shows the estimate's evolution over time, which can be used as a confidence level. CPP limits the number of estimation rounds to three. The goal of the first CPP round is to produce an estimate, and the succeeding rounds are to bring the team of crowd estimators to a consensus. Four confidence levels: Confident, Semi-Confident, Low-Confidence, and Poor-Confidence are used, depending on how many rounds the CPP takes: one round, two rounds, three rounds with consensus, and three rounds with no consensus, respectively.

*Assessing crowd assignments' quality* is a key requirement to reliably crowdsource CPP. A legitimacy score of a crowd assignment is used to decide the inclusion or exclusion of the assignment. Predicting effort is a subjective task, and thus, crowd estimates cannot be accepted or rejected based on the actual effort or expert estimate. At the same time, it is hard to accept all the assignments that are submitted by crowd workers, since there is a chance of unwanted assignments that may end among other legitimate assignments.

Therefore, an assignment legitimacy score was developed to classify the assignment into one of three classes: Considered, Ambiguous, and Disengaged. Each crowd assignment went through a manual legitimacy classification process of two components. First, the justification provided by the crowd worker was evaluated for the presence of the following components:

1. a task breakdown,
2. a time assignment for each working block,
3. a general discussion about the task topic, and

4. an explanation of the applied estimation process.

Secondly, self-reported experience responses to the two questions concerning experience were evaluated to test whether a relevant answer had been provided:

1. number of experience years, and
2. experience field.

A classification process by a group of four researchers produced a final class as shown below.

- Considered: crowd justification contains at least three of the four elements mentioned above, e.g. a breakdown of the task with time assignment for each sub-task. The crowd worker's experience is within the software engineering discipline and they have previous experience in developing software.
- Ambiguous: crowd justification contains two of the four elements mentioned above, and the crowd worker has experience within the software engineering discipline.
- Disengaged: crowd justification does not contain the expected elements and there is no relationship to the estimation process, e.g. talking about something else.

Finally, the legitimacy score was calculated as the percentage of considered assignments from received assignments.

## 6.3 Experimental Design

As stated earlier, this is a preliminary study that aims to discover a working setting and design of CPP. Therefore, three pilot experiments were designed to investigate the feasibility of CPP using human computation and answer the following pilot research questions (PRQ).

- PRQ1: What are the working settings of crowd size and amount of information that enable the crowd to predict an expert-comparable estimate?
- PRQ2: What is the proper process design for CPP that can imitate Planning Poker features in a micro-task crowdsourcing environment?
- PRQ3: Given a software task that requires between one hour and two weeks of effort, can a crowd team produce a cost estimate that is of comparable accuracy to that of a project expert?

The first experiment is the Information Experiment. It concerns what information is necessary for crowd workers to predict an estimate. The workers have very limited time, and thus, reading material should be limited to what is necessary. Moreover, it is important to differentiate between the kinds of information that are compulsory and optional. These points will be further detailed in the Information Experiment. This experiment partially answers PRQ1, i.e. regarding the amount of information.

In the second experiment (Crowd Size Experiment), the focus shifts to examine how large the crowd group should be, i.e. how many crowd workers should be hired for each CPP round. CPP should be efficient, and having the right number of workers is important. Hiring fewer workers may impact the estimate accuracy, but a large number of workers may result in inefficiency. This experiment answers the team size part of PRQ1.

The last experiment is the Process Design Experiment and it examines whether the proposed design of CPP will help crowd workers in their estimations. While the focus is on the overall design of the process, it also evaluates different user interface options to avoid any misunderstanding of the CPP process. The third pilot experiment is designed to study the proposed CPP process and examine whether any adjustments are necessary. It therefore answers PRQ2.

The collective results from the three experiments will be used to answer PRQ3. The next three subsections illustrate more design details of the three pilot experiments.

### 6.3.1 Dataset

The dataset that is used in this experimental work is JOSSE, which was detailed in Chapter 4. Unfortunately, it was not an option to use the PPI dataset mentioned in Chapter 5, since the researcher has signed a non-disclosure agreement with the commercial organisation. Other datasets, such as Deep-SE [180], have no expert estimates among their attributes.

The expert-estimated and actual efforts are reported as literal person-minutes in the JOSSE dataset. In contrast, many sources advocate using an approximate estimation unit in Planning Poker, such as story points Grenning [8]. Cohn [5] argues that software tasks are notoriously difficult to estimate accurately to a high level of precision, making approximate person-effort categories more appropriate. Further, Cohn [5] argues that teams are able to gradually develop an idiosyncratic, tacit understanding of how these units relate to actual person-time costs over a series of sprints, based on the team's review of its performance.

The literal person-minutes efforts reported in JOSSE were grouped into categories so that they could be compared with approximate costs produced during a CPP activity. The estimate unit labels used were *one hour*, *half a day*, *one day*, *half a week*, *one week*, *two weeks*, and *more than two weeks*.

Issue Id	Experiment	Community	NoC	NoW	EEE	AE
PBR-413	Info. Level	JBoss	1	51	3h	2.5h
GTNPORTAL-1606		JBoss	2	64	8h	6h
RF-11453		JBoss	14	14	24h	16h
JBBUILD-335		JBoss	3	59	80h	72h
EXOJCR-1259	Crowd Size	JBoss	13	149	24h	75h
JBTM-1568		JBoss	19	48	80h	48h
IGNITE-10965		Apache	7	182	8h	4.5h
AEROGEAR-5533		JBoss	3	165	8h	2h
ENTMQBR-1619	Process Design	JBoss	14	86	8h	4h
MSITE-68		Apache	4	175	3h	3h
INT-541		Spring	2	29	1h	3h
AMQCPP-223		Apache	1	39	60h	20h
INT-2312		Spring	16	26	24h	23h
EXOJCR-420		JBoss	14	79	4h	20h
NETBEANS-905		Apache	3	70	168h	79h
EXOJCR-1104		JBoss	13	194	48h	98h
HBASE-12128		Apache	24	143	120h	72h

Table 6.1: Properties of software development tasks selected from the JOSSE dataset. NoC stands for number of comments, NoW for the number of words in the issue description, EEE for expert-estimated Effort, and AE for actual effort.

Seventeen issues were randomly selected from the JOSSE dataset for the three experiments. Four issues were allocated to the Information Experiment, another four issues allocated to the Crowd Size Experiment, and nine issues allocated to the Process Design Experiment. Before picking the issues, the dataset was filtered to avoid selecting an issue with no expert estimate, no comments or no description. Table 6.1 lists the software development issues according to each experiment. It also shows the issues' source community, number of comments, number of words in the issue description, expert effort estimate, and actual effort in hours.

For the Process Design Experiment, the focus moved to examine the CPP process, and thus, more issues were needed to represent a wider range of issue types, including software bugs, feature requests, and software enhancements. Also, the condition of issues having a similar time category for the actual and expert-estimated effort is not needed. Table 6.1 illustrates more details about the selected issues, and Chapter 4 gives more details about the dataset's origins, collection process, and quality.

### 6.3.2 Measures

Several measures were required to address the research questions, including estimation performance, quality score, crowd consensus, and crowd interaction with CPP jobs and the

interface. Estimation Performance, quality score, and crowd consensus are required to answer all three questions, whereas crowd interaction will mainly be used to answer PRQ2 and PRQ3.

Starting with estimation performance, the three most used measures of effort estimate accuracy are MMRE, MdMRE, and Pred(25); more details about them are explained in Chapter 5. Magnitude of Relative Error (MRE) is also used to indicate crowd and expert accuracy for each round in each issue.

To decide on the proper size of the crowd team and the right amount and type of information, crowd interactions were measured by counting the number of workers a task may take until reaching a consensus. In addition, the number of crowd interactions with CPP interface, including the number of clicks, mouse movements, and highlights were recorded and counted to examine how much information (text corpus) the workers will need and can handle.

### 6.3.3 Experiment Trials and Variables

In the first study (Information Experiment), this process proceeded through three iterations ignoring the consensus calculation, hence allowing the effect of iteration on consensus forming to be studied. We anticipate that if CPP was used for real estimation, then the consensus calculation could be used to decide whether to terminate the activity early.

The dependent variable of the first two experiments is assignment legitimacy. As a reminder, the aim of the two experiments is to determine working settings for CPP, and not necessarily to produce an expert-comparable estimate. Thus, estimate accuracy is not the focus. Moreover, the independent variables of the two experiments are: actual effort (as recorded in the JOSSE dataset), amount of task information provided to the crowd for the information experiment, and crowd size for the second experiment.

A variety of actual efforts were ensured, including: hour, half-day, day, half-week and two weeks, using the labels for effort estimates as described in Section 6.3.1. Different actual efforts were necessary in order to determine whether the crowd could distinguish between tasks of different complexity.

Information available to the crowd workers about the task being estimated is divided into two categories. Basic information, which comprises issue title and supplementary description, and Extended information, which includes the following information resources for the crowd worker to review:

- Contextual project details.

- Definitions of all terms in the issue title and description identified by the experimenters as being ambiguous, abbreviations, or project-specific labels (such as the name of a component).
- All the comments that the development team made on the given issue.

Initially, the crowd size was set to five workers per CPP round. Workers who submitted Ambiguous or Disengaged assignments were not considered. While this setting was used in the Information Experiment, a crowd size of up to 20 workers per CPP round was allowed in the second experiment (the Crowd Size Experiment).

In the Process Design Experiment, the dependent variable is estimate accuracy, and the independent variables are the actual effort, issue type, and consensus. After excluding a major part of the process, consensus, in the previous experiments, this experiment examines the full design of the CPP process as proposed above. In addition, it ensures that crowd workers can estimate different kinds of issues using the same CPP process.

Table 6.2 lists the three experiment trials along with the different variable settings of each trial.

Trial #	Experiment	Info. Level	Crowd Size	Consensus	Different Issue Types
1	Info. Level	extended	73	no	no
2		basic	73	no	no
3		extended	73	no	no
4		extended	73	no	no
5		extended	73	no	no
1	Crowd Size	extended	121	no	no
2		extended	121	no	no
3		extended	121	no	no
4		extended	121	no	no
1	Process Design	extended	73	yes	yes
2		extended	73	yes	yes
3		extended	73	yes	yes
4		extended	73	yes	yes
5		extended	73	yes	yes
6		extended	73	yes	yes
7		extended	73	yes	yes
8		extended	73	yes	yes
9		extended	73	yes	yes

Table 6.2: Summary of trials settings: information level, crowd size, and whether consensus and different issue types were included.

### 6.3.4 Evaluation and Result Test

The evaluation of the Information Experiment results is done by comparing the legitimacy measures between the basic information trial and extended information trial. In addition, the legitimacy of the three remaining trials is used to ensure a consistence measurement over different actual effort time categories.

Similarly, the Crowd Size Experiment is evaluated by comparing the legitimacy measure from the Information Experiment (where small groups of workers are hired) with the legitimacy measures of the four trials in this experiment.

In the third experiment, the Process Design Experiment, the evaluation is based on the difference in accuracy between expert-estimated efforts and crowd-estimated efforts. Actual effort is used as a ground truth to measure the accuracy of both estimates.

A statistical significance test is used on the experiment results to ensure that the observed differences are statistically significant.

## 6.4 Results

This section gives the results of the three experiments concerning information level, crowd size, and process design. In the first two experiments, the goal is to evaluate whether crowd workers are able to produce a useful estimate regardless of accuracy measures. The point is to make sure that there are crowd workers willing to take such an estimation task and to make sure that CPP as a process (the core design with consensus disabled) guides the workers to produce a legitimate estimate. After ensuring crowd workers' capability and the working settings of CPP, the third experiment measures the accuracy of crowd estimates and compares it to that of expert estimates across different types of task and with the full design of CPP (i.e. considering consensus).

### 6.4.1 Information Experiment

The Information Experiment has five trials, and it considers the impact of additional information on the estimates produced by the crowd. Based on the results listed in Table 6.3, the crowd in the second trial, using only a basic level of information, eventually results in the lowest legitimacy score. Conversely, the crowd in Trial 1, estimating the same issue with extended information, produced a more accurate (if still incorrect) estimate of One Day, and did so with a better legitimacy score. Whilst preliminary, these results do suggest that a crowd benefits from additional contextual information when producing an estimate.



Issue Id	Information Level	RE	CE	Legitimacy Score	Crowd Estimate (MRE)	Actual Effort
PBR-413	Extended	76	25	33%	One day (100%)	Half-day
PBR-413	Basic	72	12	17%	Half-week (400%)	Half-day
GTNPORTAL-1606	Extended	66	21	32%	Half-week (0%)	Half-week
RF-11453	Extended	75	25	33%	Half-week (0%)	Half-week
JBBUILD-335	Extended	75	24	32%	Two weeks (0%)	Two weeks

Table 6.3: Summary of the five trials, including received (RE) and considered (CE) estimates, legitimacy score, and crowd outcome for each issue.

Issue Id	Number of Workers	RE	CE	Legitimacy Score	Crowd Estimate (MRE)	Actual Effort
EXOJCR-1259	112	208	60	29%	One Week (16.67%)	Two weeks
JBTM-1568	130	247	56	23%	Half-week (20%)	One week
IGNITE-10965	111	202	59	29%	Half-week (100%)	Half-day
AEROGEAR-5533	132	224	59	26%	One Week (150%)	Half-day

Table 6.4: Summary of the four trials of examining large size of crowd. It includes received (RE) and considered (CE) estimates, legitimacy score, and crowd outcome for each issue.

Trials 1, 3, 4, and 5 under the extended level of information allowed crowd workers to request additional information about the task or its context. This enabled us to track whether this information was actively sought by the crowd workers. The workers made 29, 9, 20, and 52 requests, respectively, during the trials with an extended level of information available. Further, a small subset of workers also used the provided search functionality to engage in open searches about the project.

These results indicate both that the crowd workers are willing to obtain additional information in order to complete their task and that they benefit from doing so. Thus, the extended level of information is the option that will be applied for the rest of this thesis's experiments.

## 6.4.2 Crowd Size Experiment

During the first experiment, an average number of 73 workers (24 workers per round) were hired to predict the estimates, and the average legitimacy score was 33%. In this experiment, another four issues were selected for estimation but with an average increment of 66% in the number of workers. According to Table 6.4, the average legitimacy score is 27%, which is slightly lower than the legitimacy score in the first experiment. Therefore, the increase in workers does not necessarily result in better legitimacy or better estimates. Therefore, the rest of the experiments in this thesis use the same number of workers as the first experiment.

### 6.4.3 Process Design Experiment

Enabling consensus and using a wider range of issues were part of the CPP process examination for producing expert-comparable estimates. The results of the trials reported in Table 6.5 show that the crowd workers correctly estimated the issue category in five of the nine trials (3, 4, 5, 6, 8) and also outperformed the expert estimation baseline (comparing MRE) in five of the nine trials (4, 6, 7, 8, 9). Therefore, overall, the crowd workers produced the same or better estimates than expert estimators in seven of the nine trials. The results also demonstrate that the CPP process can effectively discriminate between tasks of different orders of magnitude, ranging from half a day through to two weeks.

One caveat to these results is the outcome in trials 1 and 2, where the crowd workers dramatically overestimated the effort required, producing very large MRE scores (900% in both). As a consequence, the Mean MRE (MMRE) of the crowd workers across all nine trials is 214% compared to 122% for the expert estimate. Excluding these two outliers for the crowd workers reduces the MMRE to 18%. Chapter 9 gives some fine-grained information that offers insights on the questions of why and how the crowd workers under-performed in these two trials.

## 6.5 Discussion

This section addresses the three research questions mentioned earlier in Section 6.3, starting with the proper settings for amount and type of information, crowd team size, and the design of CPP process. It also addresses the feasibility of playing CPP using human computation and in a micro-task crowdsourcing environment. The questions' answers and related details will be discussed as follows.

### 6.5.1 PRQ1: What are the working settings of crowd size and amount of information that enable the crowd to predict an expert-comparable estimate?

The first experiment's results indicated that crowd workers performed poorly and achieved a lower legitimacy score at the basic information level, but it was improved by using an extended level of information. Such a result suggests the the extended level of information is the right option.

In contrast, the second experiment resulted in no improvement whatsoever when the number of crowd workers was increased by 66%. Thus, the number of workers used in the first experiment (73 workers per issue) is appropriate. Taking into consideration that not all of

Issue Id	RN	Consensus (%)	Issue Type	Crowd Estimate (MRE)	Expert Estimate (MRE)	Actual Effort
WFWIP-18	1	19%	Bug	One week (900%)	Half-week (400%)	Half-day
	2	0%				
	3	26%				
MSITE-68	1	0%	New Feature	One week (900%)	Half-day (100%)	Half-day
	2	19%				
	3	33%				
INT-541	1	26%	Impr.	Half-day (0%)	One hour (100%)	Half-day
AMQCPP-223	1	13%	Bug	Half-week (0%)	Two weeks (300%)	Half-week
	2	33%				
INT-2312	1	14%	New Feature	Half-week (0%)	Half-week (0%)	Half-week
	2	33%				
EXOJCR-420	1	0%	Impr.	Half-week (0%)	Half-day (100%)	Half-week
	2	20%				
	3	33%				
NETBEANS-905	1	19%	Bug	Half-week (75%)	Two weeks (0%)	Two Weeks
	2	20%				
	3	60%				
EXOJCR-1104	1	20%	New Feature	Two weeks (0%)	One week (100%)	Two Weeks
	2	33%				
HBASE-12128	1	20%	New Feature	One week (50%)	Two weeks (0%)	Two Weeks
	2	33%				

Table 6.5: Summary of the nine trials of examining CPP process after using consensus. It includes round consensus that is measured using Fleiss' Kappa, crowd and expert estimates along with their MREs for each issue. RN stands for Round Number and Impr. for Improvement.

the 73 workers were able to submit a legitimate estimate, the number of workers is expected to drop after developing quality controls. Moreover, each round had a distinct set of workers, meaning that each round needed an average of 24 crowd workers.

### **6.5.2 PRQ2: What is the proper process design for CPP that can imitate Planning Poker features in a micro-task crowd-sourcing environment?**

The proposed design of CPP as illustrated in Figure 6.1 features an iterative, yet asynchronous, estimation model. As in Planning Poker, the iterative loop continues until consensus, which enables using consensus as an aggregation method. If the crowd workers do not reach a consensus, a limit on the number of iterations is imposed, imitating the Planning Poker coordinator action of time-limiting the Planning Poker sessions.

Crowd workers provide their initial estimates individually to avoid bias, and this is similar to Planning Poker members making their initial estimations secretly. The asynchronous communication of CPP has not impacted the communication between crowd workers. In the next round, the crowd workers synchronise their understanding at the beginning of the next round by reading a summary outcome of the previous round and the boundary estimates highlighted along with their estimators' rationale.

This design of CPP shows evidence over the three experiments that it enabled crowd workers to imitate Planning Poker and produce legitimate estimates, although the legitimacy score is relatively low, with an average of 30%. Taking into consideration the high number of disengaged estimates may explain the lower legitimacy score as the existence of poor quality crowd assignments. Therefore, there is a need to design and implement a quality control in the CPP process to avoid unengaged work.

### **6.5.3 PRQ3: Given a software task that requires between one hour and two weeks of effort, can a crowd team produce a cost estimate that is of comparable accuracy to that of a project expert?**

Whilst preliminary, these results show crowd workers were able to predict expert-comparable estimates. However, a more rigorous study is essential to answer this question with confidence. Perhaps different kinds of software issue need to be considered and a larger number of crowd workers is required to come up with conclusions concerning the reliability of CPP.

The results presented in this chapter were drawn from pilot experiments to sufficiently justify the investment in a larger scale study, rather than conclusively answer the thesis research questions.

#### 6.5.4 Beyond Estimates – Crowd Insights

An additional benefit of requesting a rationale from crowd workers when they supply their estimates is that further insight and analysis of the task to be estimated can be obtained. Many of the workers provided useful information about how to approach the task. Such advice and guidance was often very detailed. For example, on a task concerning the creation of a preview mode for sites using the Apache Maven site plugin (MSITE-68), a crowd worker wrote:

“This seems like a good case for building at the DOM level, to ‘implement’ the changes in parallel for the previews. If that is in fact the case, it would probably take about a day to get a working prototype. If not... then a day would also probably be enough to know that this simply cannot be done.”

The crowd worker provides a suggestion that the resolution of the issue can be done by monitoring a page’s DOM for changes to create a preview. They also include a suggestion that a prototype should be created first to determine whether the feature is feasible.

For another issue, concerning the implementation of a new indexing mechanism for a JBoss workspace, the crowd worker provides a detailed breakdown of the work to be done:

- “1. How to determine, and what is the most efficient and accurate query for nodes and necessary information?
2. Initial testing for viability of indexing nodes (no lost data, consistency, etc.)
3. Deeper testing incl. stress testing at higher node counts, ensure all threads are deleted, etc.”

In particular, the crowd worker emphasises the importance of different types of testing, noting that non-functional testing should be treated separately from the design and functional testing of the feature.

These examples were intriguing, as we had not anticipated that crowd workers would provide insights with significant *domain-specific* knowledge. These suggestions and explanations have the potential to be of significant assistance to a team during the wider triage process for a software task that occurs alongside estimation. Chapter 9 offers fine-grained information to better understand why and how the crowd workers have performed.

Trial	Sign-ups	Estimates			
		Received	Paid	Minutes	Cost
<b>1</b>	96	35	30	15	\$4.50
<b>2</b>	78	32	23	16	\$3.45
<b>3</b>	72	22	13	4	\$1.95
<b>4</b>	83	33	14	6	\$2.10
<b>5</b>	93	34	15	7	\$2.25
<b>6</b>	89	40	25	10	\$3.75
<b>7</b>	83	37	26	19	\$3.90
<b>8</b>	88	31	20	9	\$3.00
<b>9</b>	117	37	25	12	\$3.75
<b>Total</b>	799	301	191	98	\$28.65
<b>Mean</b>	89	33	21	11	\$3.18

Table 6.6: Breakdown of trial costs of the third experiment (process design).

### 6.5.5 Evaluation of Costs in Pilot Studies

During the first two experiments, the costs of running CPP were neglected. Firstly, CPP was applied partially without enabling consensus, and thus additional rounds were played with no need for them to continue. Secondly, the experiment trials did not reflect the proposed CPP; in fact, they were used to figure out the proper settings for CPP. However, in the third experiment, the CPP process was fully applied and the information level and crowd size were fixed. Therefore, only the cost of the third experiment is considered. Table 6.6 summarises the costs and effort associated with the trials of the Process Design Experiment.

The table shows that the total amount of time to produce an estimate through CPP ranged from 4 to 19 minutes. Unsurprisingly, the number of rounds in a trial had a significant influence on the time taken, with the third trial requiring just a single round and lasting just four minutes, for example. These results suggest that producing an estimate from a crowd takes some additional time, compared with Planning Poker. Expert estimation may also be considerably faster when the expert group already has a good understanding of the task to be estimated and can rapidly achieve consensus without the need for discussion. Nevertheless, the results demonstrate that crowds can produce estimates relatively quickly and on demand.

The table also reports the cost for conducting the trials, showing an average cost of \$3.18 to produce a final estimate (again, this figure is influenced by the number of rounds taken in a trial). This cost appears to compare very favourably with the cost of running a Planning Poker session within a software team. Assuming a team of five developers with an average hourly salary of \$40 (excluding other costs) can estimate 10 tasks in hour, then the average cost per estimate would be \$20. Thus, the results of the trials demonstrate the potential for a significant cost saving.

### 6.5.6 Threat to Validity – Issue Availability

A limitation of the software development issues used is that they are collected from open source projects. This decision was necessary as the experiment required a source of software tasks that could be provided to anonymous crowd workers and that had been annotated with expert-estimated and actual work cost. This meant there was a risk that the crowd workers could access the issue trackers themselves and simply supply the actual reported cost, creating a threat to the validity of the reliability results.

This risk was mitigated in several ways. First, the issue identifiers were not supplied to the crowd workers and issues were selected from issue trackers that required user registration. This created an additional step to deter workers. Second, workers were asked for a categorical submission, rather than an absolute person-hour value, creating an additional step if the source issue was accessed. Finally, workers were encouraged to supply their estimate and it was clear that payment was not contingent on supplying the correct result. Consequently, there is no evidence in the behaviour logs that the workers accessed project issue trackers, although this may have occurred outside the CPP user interface.

Software development issues from open-source communities were used since they are publicly published and the concern of having sensitive data is avoided. However, using these issues introduces the risk of having a biased input that may not be applicable to other types of software development project, such as those from commercial development houses.

To mitigate the risk of bias, the issues were selected from different open-source communities, specifically, the JBoss, Apache, and Spring open-source communities. Further, the selected issues represent different kinds of development work, namely, bug issues, software improvement requests, and new feature requests. Randomness was also used in selecting the final issues that were used in the three pilot experiments.

The pilot experiments were conducted on a limited number of software development issues. Consequently, the experiment recruited a relatively small number of crowd workers. The limited number of issues and workers poses a challenge to using the experiment results to derive conclusions about the reliability of CPP.

Thus, the experiments and their results are not used to draw any conclusions about the CPP reliability. Instead, they are used to examine whether the crowd worker can produce estimates using the proposed CPP or whether the CPP design would not work in the micro-task crowdsourcing environment. In case the crowd workers were able to produce expert-comparable estimates using CPP, which they did, the experiments did help in identifying the proper size of the crowd, the amount and type of background information, and the specifics of the process design. In addition, by conducting those experiments at a low cost, the researcher learned what practical challenges may occur in playing CPP, one of which is handling the

quality of crowd assignments.

## 6.6 Summary

This chapter has presented the first study of applying crowdsourcing to Planning Poker for the production of software task estimates, answering Grenning [215]’s speculation from more than a decade ago.

Starting with general considerations of Planning Poker, the game context and goals, players, input and output, communication, and process were analysed with respect to playing Planning Poker in a micro-task crowdsourcing environment. Further, Planning Poker’s desirable characteristics (specifically, bias avoidance, synchronising team comprehension, consensus, and aggregation) and iterative estimation were also reviewed with the intention to imitate them in CPP.

Then, CPP was designed according to the considerations mentioned above. Both process and data flow are explained and illustrated using flowcharts. The four desirable Planning Poker characteristics mentioned above were imitated in the CPP design.

After that, three pilot experiments were designed to address the unknown parts of the CPP design, specifically, the type and amount of information, the size of the crowd estimator team, and the CPP process. Each experiment addressed one of these questions. The collective analysis of the three experiments answers the chapter’s main research question about the feasibility of playing Planning Poker in micro-task crowdsourcing environment.

The experiment results were presented and followed with a discussion about the three concerns mentioned above. The work demonstrates that crowd workers, organised in a CPP process, can produce software task estimates comparable with those produced by experts, and at a substantially reduced cost compared with small teams of domain experts. The crowd workers were able to discriminate between tasks of varying complexity and provide useful insights as to the resolution of the task. However, the quality of crowd assignments was a clear obstacle to playing CPP in a micro-task crowdsourcing environment.

Quality management in the crowd environment is an essential aspect that need to be addressed. The CPP process needs to be adjusted to consider different quality controls, as the experiment results showed that more than 70% of crowd assignments were not acceptable due to their poor quality. Moreover, the manual checking that is used in this chapter experiment may not be feasible, since it requires additional human resources and it might be not possible to meet crowd demand.



## Chapter 7

# Quality Assessment and Enhancement of Crowd Planning Poker

In the previous chapter, the feasibility of Crowd Planning Poker (CPP) for Software Effort Estimation (SEE) was assessed. The conclusion of three pilot experiments indicated that playing Planning Poker using human computation is feasible and can produce software estimates. At the same time, the experiments revealed the poor quality of a significant proportion of crowd assignments when run by micro-task crowdsourcing platforms such as Amazon Mechanical Turk (AMT).

Micro-task crowdsourcing platforms are attractive because they enable the systematic recruitment of workers to perform micro-tasks. However, a side-effect of this approach to recruitment is that workers are treated as disposable computing resources by requesters, and workers generally do not develop long-term relationships with requesters. In such an environment, trust-based relationships [223, 224, 225] are difficult to establish and this can be conducive to fraudulent behaviour by workers [226].

In general, the inability of buyers to discriminate offers on the basis of quality in a market is known to drive down the quality of all offers [227]. In the context of crowdsourcing, a worker is not incentivised to engage fully with a task and provide as high a quality submission as possible, because they lack feedback as to what actions would improve the quality of their submission; and in any case, the additional effort required cannot be rewarded relative to that of a low quality submission. Rather, workers are incentivised to complete a large number of submissions as quickly as possible in order to maximise their anticipated reward relative to their effort. Such an approach discourages workers from engaging fully with a task, and they will seek to minimise their effort and thus the quality of their submission.

In addition, the judgements performed by crowd workers in the context of CPP are inherently subjective. Rating the quality of submissions for such tasks is difficult, because there may not be ‘gold standards’ against which the quality can be measured or compared [228]. Workers on crowd platforms are effectively anonymous, which makes assessment of their qualifications to perform a task or attribution of their work difficult.

Given this challenge, this chapter proposes an approach to actively managing the quality of work by crowd workers, comprising: (a) a model of quality for the CPP task; and (b) active feedback to workers as to the quality of the current draft of their assignment prior to submission. To address the subjective nature of the quality of the assignment itself, we identify a set of *proxy* markers for task quality concerning artefacts associated with the assignment and worker behaviour, drawing on the existing literature of worker behaviour [229] and crowd rationales [230]. These markers are used to train a classifier to predict low quality assignments. Draft assignments that are identified as low quality are highlighted to the relevant worker, giving them the opportunity to improve their assignment and avoid potential rejection. The approach therefore relies on loss attention theory [27] to incentivise workers to improve their assignment and avoid the loss of income and reputation due to rejection.

The rest of the chapter is organised as follows. Section 7.1 details the problem of crowd quality, particularly focusing on effort estimation using CPP. Then, Section 7.2 describes the approach to measuring subjective crowd tasks, proposes a quality measuring model, and demonstrates its performance. Section 7.3 describes how to incorporate the measure of task quality into the crowd task to offer real-time feedback, while the crowds undertake the work. Finally, Section 7.5 sums up the results, limitations, and implications for crowdsourced work in general and considers future work on enhancing and integrating quality in a crowd marketplace for general subjective tasks.

## 7.1 Exploring the Quality of Crowd Assignments in the CPP Context

As explained above, not all the crowd submissions are qualified to be considered. More than half are of low quality. Manually evaluating every assignment and assessing its quality consumes a lot of time and energy. For instance, in the first experiment (Chapter 6), collecting crowd estimates needed a couple of days; however, it took a couple of weeks for a group of four researchers to review 364 estimates and classify them. Thus, the requester finds themselves in a trade-off between quality and resources. Such a challenge may invalidate the original promise of the crowd if not managed. Besides, the cost of hiring four researchers to review the 364 estimates is much more expensive than hiring expert estimators to estimate these issues in the first place.

Issue Id	Community	NoC	NoW	EEE (Hour)	AE (Hour)
PBR-413	JBoss	1	51	3	2.5
GTNPORTAL-1606	JBoss	2	64	8	6
RF-11453	JBoss	14	14	24	16
JBBUILD-335	JBoss	3	59	80	72
AMQCPP-223	Apache	1	39	60	20

Table 7.1: A summary table of characteristics for the selected issues from the preliminary study used in developing the quality measurement framework.

Another challenge is the nature of the estimation task, which is subjective. Thus, accuracy in the context of assignment quality is not applicable, since there is no reference for a prediction. Any reasonably justified estimate is a good estimate, even if it is far from the actual effort. CPP has been designed to limit crowd estimators' subjectivity using two components: limited estimate options and a summary of estimates from the previous round. Even with such a subjectivity-aware design, sometimes crowd workers select from the full range of estimation options available to them. This has happened when an estimation session consumes all three rounds allocated for it. For instance, in the second trial of the information experiment (the first experiment), the subjectivity of the crowd workers was high, so they did not reach a consensus.

Therefore, the initial goal was to define a measure of quality that could be used to evaluate worker assignments in Crowd Planning Poker without relying on the estimate accuracy itself. Most research in quality management in crowdsourcing concerns the correctness of worker assignments relative to some objective oracle, with relatively few studies addressing wider dimensions [231, 232]. However, the wider dimensions of *data* quality have been extensively researched, as demonstrated by the survey by Sidi et al. [233]. This literature suggests that there is no one definition of quality for a crowd task and that quality should be defined within the context of the specific task itself.

Drawing on this work, we therefore conducted an initial pilot study of our Crowd Planning Poker process to generate a dataset of worker-submitted assignments and associated worker behaviour during task completion, as detailed in Chapter 6. This work resulted in a dataset of 364 received assignments. Each assignment comprised a worker declaration of having software engineering experience, an explanation of that experience, a software task estimate for a proposed task, a rationale for the proposed estimate, and a log of worker interaction with the interface (worker behaviour). A summary of issues from the preliminary study used in the development of the quality model described in this thesis is given in Table 7.1.

Studying the assignments, we proposed four dimensions of assignment quality for crowd worker assignments in Crowd Planning Poker: completeness, consistency, uniqueness, and relevance. Each quality dimension has a different impact on the assignment's quality class.

For instance, to have an assignment classified as “Excellent”, all the quality dimensions have a role for that class, whereas the “Poor” class is affected by the uniqueness dimension only. The dimension definitions, specifics, and measures are detailed in the following subsections.

We ignored other concerns such as validity at this stage, since they were controlled by the crowd task user interface, even though the crowd behaviour logs show attempts to submit invalid assignments, e.g. to stop the JavaScript engine in the browser. However, the interface validation rules were able to control them and there were no invalid assignments.

**Completeness** Some workers cut themselves short and did not submit a complete assignment, which meant that at least one of the three assignment parts was not filled. While a software interface may help in forcing the worker to fill in a given field, it does not necessarily prevent workers from going around the interface restrictions and ignoring the field. Several crowd workers submitted “N/A” as their justification to bypass the interface restriction of not submitting an empty field. In another example, a crowd worker disabled the JavaScript feature of their browser, and thus, their behaviour (the fourth component) might not have been entirely recorded. Software errors and bugs also cause incomplete assignments.

Therefore, assessment of completeness was broken down into two stages. First, the completeness of the overall assignment was considered. All four elements of the assignment, as described above, were required to be present if the assignment was judged to be complete overall. If at least one element was missing, then a score of 0 was assigned for completeness. Otherwise, a score of at least 1 was assigned for completeness and the evaluation proceeded to the second stage, concerning estimate rationale completeness.

The second level relied on a structure supplied by the worker. Expert effort estimation best practices [6], effort estimation activities [2], and research on logs of expert estimation sessions [7] provided insights to developing the four elements. Four components that should exist in the worker’s justification were identified:

1. A task breakdown
2. A time assignment for each working block
3. A general discussion about the task topic, e.g. using similar issues from experience
4. An explanation of the estimation process applied, e.g. reference to peer estimates.

This scheme allowed for a maximum of five points to be scored for completeness, including one point for overall completeness. Any justification that achieved more than 3 points out of 5 was considered a complete justification.

**Consistency** CPP assignment consists of four parts, with three of them filled directly by the worker, and the fourth one collected automatically by the tool. Sometimes, when comparing these four elements, they contradict or do not comply with each other. This is evident when a crowd worker refers to a different estimate than the worker selected while writing the justification. For example, this worker selected half a week as an estimate, but the justification says a week: “In my opinion, it would take about a week to deploy and thoroughly test YARN on an Apache server.” Such inconsistency may happen when workers change their minds but forget to update the relative fields in their assignments, or more importantly, when the crowd behaviour (automatically collected as part of the crowd assignment) does not comply with the other part of the assignments — for instance, submitting work that it is impossible to have achieved in the time spent that was recorded in the crowd behaviour.

We considered the extent to which the different parts of an assignment were consistent with one another. Three separate relations were checked for consistency. First, the relationship between the worker’s declaration of experience and their described experience. For example, if a worker declared themselves to have software engineering experience, their description of their experience should relate to software engineering concepts, for example, experience in development using a particular programming language. Second, the submitted rationale should be consistent with the estimate itself. For example, a rationale that contained a task breakdown that comprised more work than the estimate would be considered inconsistent. Third, the worker’s behaviour needs to be consistent with the submitted assignment. For example, if an assignment has a rationale text of 50 words, the UI logs must show that the worker spent some time typing in the assignment rationale. An assignment was judged to be consistent overall if at least two of the three elements were judged to be consistent.

**Relevance** Irrelevant assignments were most obvious when a crowd worker submitted an assignment that was entirely out of the task context. For instance, during the third CPP experiment, a crowd worker fills the justification box with: “good survey” and answered the experience question: “5”. This would appear to be a deliberate attempt to secure compensation without properly engaging with the task. Not all irrelevant assignments are submitted with malicious intent. Some of them seem to be a misunderstanding. Some irrelevant assignments showed a greater level of relevance but were still far from being fully relevant to effort estimation. For example, a worker submitted “I justify the system is in [an] organised state; therefore the data is [in] good condition”, while the task was asking them to justify the estimate selected by the worker. This assignment is an example of low relevance but not complete irrelevance.

Further, the relevance dimension in CPP is limited to the worker’s justification. CPP assumes that there is a rationale behind selecting an estimate for the given issue. Suppose a worker goes off the issue topic and includes details not related to the issue in places the assignment

considers irrelevant. For instance, one worker wrote about Android IDE in his/her justification, where the issue topic is about user interface. Therefore, the issue topics and worker justifications are compared to measure assignment relevancy.

As a further measure, cosine similarity is used to assess the text-similarity between justification and issue description. An assignment that measures above 50% is considered relevant.

**Uniqueness** A contributor to low-quality assignments is copy-and-pasting content without effort. One reason for this may be that during the second and third rounds of CPP, crowd workers are shown previous estimates and their justifications, and thus, some workers just copied the justification and submitted their answer without spending time to think about it. Other workers benefit from the previous round by referring to it to justify and explain why they agree with a previous estimate. For example, a worker says: “This [is] the median estimate, a better measurement of frequency than the mean when outliers/large swings of data are present. Also, from my experience writing code this does not seem like too hard a problem.” While the worker has used previous estimates, the worker did create an original and unique assignment. Some workers just copy any text from the web and paste it even if it is irrelevant, and a quick Google search with the exact text reveals its origin.

It is unsurprising that crowd workers copy and paste text from the previous estimation round or even from the web to fill the justification or experience fields. However, the uniqueness dimension is limited to only the justification, since the worker’s experience will not change over a couple of days. If the provided assignment has a text used before in the previous estimate or there was an exact match for the justification text using the Google search engine, then the assignment was considered not unique.

Quality management is a well established problem of working with crowds [234]. The first step in tackling this challenge is to design a proper measurement framework and tools. Measurement will enable a better understanding of the problem and help in designing improved methods. The next section will explain the measurement framework and its tools.

## 7.2 Measuring Quality of Crowd Assignments

This section describes how to measure the quality of a crowd worker assignments within the Crowd Planning Poker task. The goal is to develop a means of automatically assessing the quality of a worker assignment for a crowd task in a specific context (Crowd Planning Poker). We developed an assessment strategy that uses automatic processing of both the worker’s behaviour and the justification. The crowd behaviour is used to draw a quality conclusion regarding crowd workers, and the crowd justification is used to assess the assignment quality. The strategy starts with a manual assessment of the worker’s justifications in the pilot study

Event	Target	Properties	Weight
Typing	Experience Field	15 words	1
Click	Extra Info Button	-	2
Click	Issue Comment Button	-	1
Click	Terms Definition Button	-	1
Click	Project Info Button	-	1
Spend	Extra info stage	25 second	1
Click	Google Search Button	-	1
Typing	Justification field	24 words	1
Spend	Whole task	3 min	1

Table 7.2: Summary of actions used for scoring crowd worker behaviour.

in Chapter 6. After collecting enough data, a machine learning (ML) model is trained to classify the assignment according to the quality classes as explained in the previous section.

### 7.2.1 User Behaviour Quality

Our next goal was to identify features that contribute to the observed quality criteria described in the previous section. To do so, we reviewed the user interface logs of worker behaviour from Chapter 6 in order to identify behaviours that contributed to behaviour quality. For example, log data that showed a worker moving the mouse pointer over a relevant part of the task, such as the software task description, was considered to contribute to quality.

We therefore identified a set of nine worker actions observable in the user logs that contributed to quality, as shown in Table 7.2. For each action, we defined the event type, the target user interface component, a metric for the event and the number of points to be added to the behaviour quality score if the event was observed. For example, if a user typed at least 15 words into the experience field, we added 1 point to the behaviour score. Similarly, if a user spent at least 25 seconds focused on the extra information page of the task (providing contextual information concerning the wider project), a further 1 point was added to the score. All the identified actions and weights were defined based on the researcher's assessment of the data from Chapter 6. The entire log of a worker's task assignment was scored against this table.

Tracking crowd interaction with the user interface for quality purposes is known as fingerprinting in the literature [229]. Unlike previous studies, this chapter extends the core fingerprinting concept. Firstly, it only includes logs for specific and relevant interface elements, such as action buttons and main text areas. Secondly, it uses different weights to calculate the behaviour score. Both the relevant interface elements and their weights are determined based on data from the pilot experiments for good and bad behaviours. To distinguish this

extension from the original fingerprinting, it is referred to in this thesis as Weighted Crowd Behaviour (WCB).

### 7.2.2 Manual Assessment of Issue Quality

Using these definitions, we proceeded to categorise assignments from the five issues listed in Table 7.5 according to their quality, using the following scale:

- A Assignments have a completeness score greater than 2, are relevant, consistent and unique, and the worker behaviour score is greater than 7.
- B Assignments have a completeness score of 2, are relevant, consistent and unique, and the worker behaviour score is greater than 4.
- C Assignments have a completeness score of 1, are relevant and unique, and the worker behaviour score is 4 or less.
- D Assignments are not complete, or not relevant, or not unique.

Classes A and B are considered acceptable quality, and thus they do not result in additional improvement work from the worker. However, classes C and D are considered unacceptable. Each crowd assignment went through a manual classification process of the four quality components by a group of four researchers. Most of the assignments with poor quality ended up in the ‘D’ category including those that were submitted by workers who were not engaged in process.

The manual classification of the five issues involved 364 crowd assignments. Thirty-seven assignments were classified as Class A and 70 as Class B. The majority of the assignments were classified as Class C (115) or Class D (142). The four researchers vote for a class for each issue, and the class with the majority of votes gets selected. In the case of non-majority votes, the team members discuss their opinions and vote again. If there are no majority votes after three voting rounds, the issue is considered invalid and not included. None of the issues has been invalidated because of researchers’ disagreements.

An assignment consists of four parts, including an estimate of a given software issue, a rationale supporting the selected estimate, the worker’s experience, and the worker’s behaviour (see Figure 7.1 for an illustration). While the estimate is what is needed for the estimation process, other parts are used as proxies for the assignment quality. The experience and behaviour parts are quality proxies of the worker, while the rationale is a proxy for the worker deliverable (the assignment).



Use of proxy information concerning the quality of a crowd assignment has been reported in the literature McDonnell et al. [235], Kutlu et al. [230]. Of particular relevance, Kutlu et al. [230] asked crowd workers to supply a rationale of their decision alongside the intended task answer. This work shows that obtaining rationales improves the quality of judgements without a substantial increase in task time. In addition, Dumitrache et al. [236] showed that rationales help uncover the reasons for subjective disagreement amongst crowd workers and thus help reach a subsequent consensus. Supplying a rationale is an original feature of Planning Poker, and is now part of CPP.

### 7.2.3 Machine Learning

Our next step was to test the viability of machine learning (ML) to automatically classify worker assignments into the four categories A–D that were specified for the manual categorisation, based only on the features in the assignment. The data of five issues from the pilot study were used, as shown in the first five rows of Table 7.5. A decision was taken to train an ML model on assignments across all five issues to ensure that variations between issues were considered, and to avoid overfitting the model.

The aim was to get a first impression of how good the classifier would be. Such an evaluation was essential to making a decision regarding the ethical and economical effects of the model on the crowdsourcing market. Thus, manually checked assignments from the pilot study were used for the purpose of evaluating the classifier only.

The four quality classes A, B, C, and D used for the manual assessment of the assignments described in Section 7.2.2 were grouped into two classes, ‘Accept’ and ‘Reject’, as the targets for the classifier. Assignments with classes A and B were considered good assignments and assigned ‘Accept’ as their new class, and classes C and D were considered bad assignments and assigned ‘Reject’ as their new class.

There are four possible outcomes based on the above grouping: accepted good assignments (True Positive), accepted bad assignments (False Positive), rejected bad assignments (True Negative) and rejected good assignments (False Negative). The classifier is correct when the outcomes are True Positive or True Negative, and incorrect otherwise. While the optimal goal of the classifier is to result in only True outcomes, in reality there is a percentage of False outcomes, which cost money and impact quality.

Rejecting good assignments (False Negatives) is ethically not fair, which basically violates the contract in the first place. It also has negative economical impacts on the market where there are no incentives for good workers to remain. On the other hand, accepting bad assignments (False Positives) economically impacts the market by incentivising bad workers. It also destroys the value that requesters (buyers) are attracted to and thus leaves no incentive

for them to remain. Thus, all false outcomes need to be handled in order to have an attractive value in the market for both workers and requesters.

The one value for requesters in the crowdsourcing market is doing the work at a lower cost than regular employees or contracted experts, and thus, the assumption here is that crowd workers' time is cheaper than non-crowd workers' time. Based on that, handling a false negative costs more than handling a false positive. False negatives cost the crowd worker's time plus the time to be inspected by the non-crowd workers (processing crowd appeals), whereas false positives cost the crowd worker's time and negatively impact the classifier quality.

While the cost of crowd workers' time and processing crowd appeals are direct, the negative impact on classifier quality is indirect. However, such a quality impact can be neglected as minimal on the final aggregated estimate given that the impact source is bad assignments, which are equivalent to the random error. In other words, the accepted bad assignments will be approximately distributed between the estimation categories, and hence, the estimator has no reason to select a particular category.

For that reason, the ML model should be tuned to minimise the number of false predictions in general, with the focus on false negatives. Thus, the recall score given by Equation 7.1 [213] is considered as an evaluation metric since it is sensitive to false negatives. However, recall alone can not illustrate the overall performance of the ML model, especially for imbalanced data. Therefore, ROC-AUC [212] is selected as a metric for the overall performance.

$$r = \frac{TP}{TP + FN} \quad (7.1)$$

where TP, FP, and FN are the true positive, false positive, and false negative scores, respectively.

As a feature extraction method, the TF-IDF matrix of the task description was used. In addition, the user behaviour score is used as-is and merged with the TF-IDF matrix. Random Forest was selected after outperforming other algorithms, including Support vector machine (SVM) and Naive Bayes, in initial trials.

As a final step before evaluating the ML model, hyperparameters need to be tuned. Given the priority of false negatives, hyperparameters are tuned to satisfy the highest recall score (i.e. to minimise false negatives). Seven parameters were identified for tuning, as listed in Table 7.3. Three of them are related to feature extraction, and the rest concern the classifier. Table 7.3 also shows the optimal values for these parameters based on the recall score.

A total of 417 assignments were received for the five issues, and they were all used in the first experiment. Table 7.4 shows the distribution of assignments across the two categories, 'Accept' and 'Reject'. The table shows that 87 assignments were correctly accepted and

Step	Parameter	Description	Optimal value
Feature extraction	max_df	Maximum document frequency of vocabulary terms to be considered	0.85
	max_features	Maximum number of feature for each document	3
	ngram_range	Minimum and maximum number of values for each word n-grams	1
Classification	n_estimators	Number of trees in the forest	90
	max_depth	Maximum depth of the tree	8
	min_samples_split	Minimum number of samples required to split an internal node	100
	min_samples_leaf	Minimum number of samples required to be at a leaf node	1

Table 7.3: Hyperparameters and their optimal values for CPP quality classification.

		Auto		Total
		Accept	Reject	
Manual	Accept	87	34	121
	Reject	33	263	296
Total		120	297	417

Table 7.4: Summary of classification results for assignments of the five issues used for classifier assessment.

263 were correctly rejected, so 84% of the classifier decisions were correct. On the other hand, 34 were incorrectly rejected and 33 were incorrectly accepted, so 16% of the classifier decisions were wrong.

The model performance, as measured by the overall metric ROC-AUC, was scored at 84.7%, whereas the recall score was 72.6%. That means the overall performance of the model is better than a random classification ( $>50\%$ ). In fact, it is close to the best quarter ( $\geq 87.5\%$ ). The recall score indicates that 27.4% of legitimate assignments will be rejected (34 assignments in the case of the pilot assignments). Since legitimate assignments represent the minority (29%), those false negative assignments represent 8.1% of all received assignments. While the classifier's overall score is good, the recall score suggests that the classifier is in general severe, rejecting assignments that would be otherwise accepted by a manual reviewer.

Given the result, it is clear there is a 27.4% chance of unfair rejection, and thus, a soft-reject process (equivalent to an appeal) was designed and introduced in the second experiment. The soft-reject process moves rejected assignments to a hold status for human inspection. Unlike the manual quality assessment, the soft-reject process provides additional information, in-

cluding the assigned class (C or D) and the classification confidence, to quickly help human inspection. For instance, Class D assignments were rejected during human inspection since they mostly came from unengaged workers, and they represent the majority of the rejects. While this may increase the overall headache, it ensures fair treatment for the crowd.

Looking at the classification outcomes (accept, reject) from a statistical point of view, a McNemar's test indicates no significant difference: McNemar's chi-squared = 0.056 and the p-value = 0.81. This is another indication that the RF classifier performance is similar to the manual process, and it can decide which assignment should be accepted or rejected with a relatively good accuracy.

## 7.3 Improving Crowd Quality

After measuring the quality of crowd assignments, the next step is improving it. This section describes how to enhance the efficiency of the Crowd Planning Poker task. The goal is to minimise waste at the worker's and requester's ends by improving the communication between them in the automated assessment process explained earlier in Section 7.2. Therefore, an improvement model with a continuous feedback loop was developed, and it is illustrated in Figure 7.1.

The model uses a crowd feedback loop to communicate assessment decisions with workers and enable them to respond. It encourages workers to enhance their assignments using Loss Attention theory, which emphasises possible losses a worker may face rather than possible gains. Apart from assignment quality enhancement, the model improves the chance of fair treatment of crowd workers by introducing a soft-reject procedure. The procedure is an adjusted appeal process that enables good rejected assignments to be considered by the requester.

### 7.3.1 Crowd Feedback Loop

A feedback loop is designed to communicate assessment decisions to crowd workers and deliver assignment improvements to requesters. It boosts the crowd worker's learning curve of assignment quality requirements. Similar work in the literature by [237, 238] suggests that engaging crowd workers in such a loop results in better outcome quality.

The Crowd Feedback Loop starts with a crowd worker submitting an assignment (Step 3 in Figure 7.1), then the CPP operator evaluates the assignment quality automatically (Step 4 in Figure 7.1). After that, the evaluation decision is communicated via a form of feedback to the worker, then the worker is given a chance to submit an improved assignment or with-

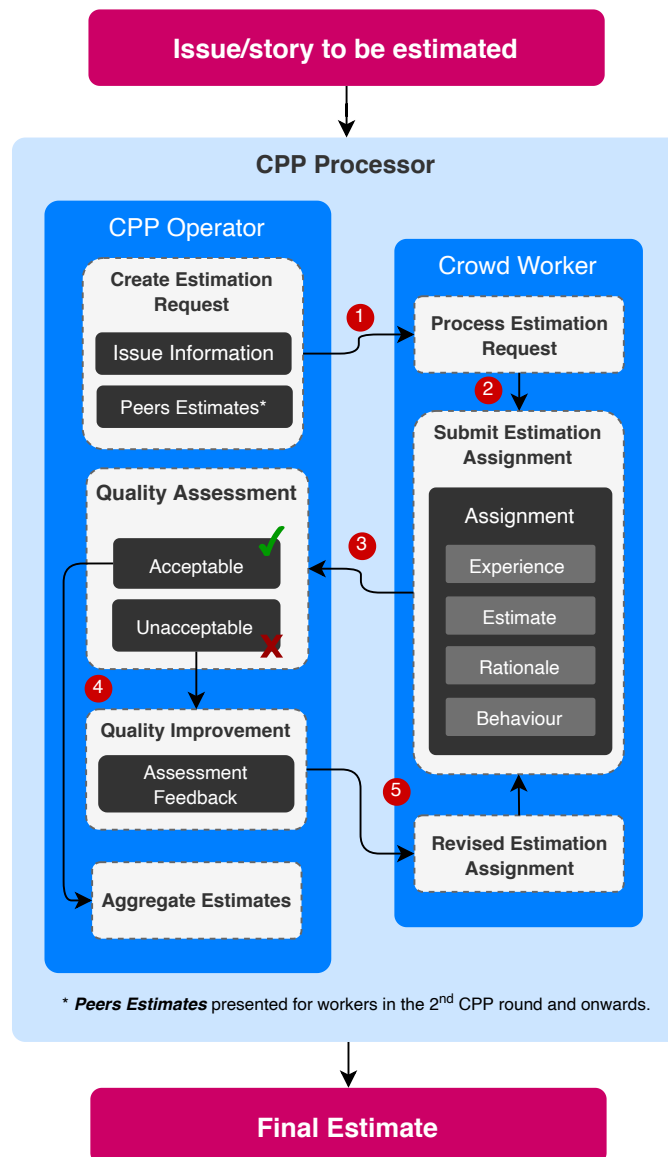


Figure 7.1: The overall real-time quality improvement framework. The framework comprises an operator component and a crowd worker component. White boxes represent the tasks within each component and grey boxes represent the data exchanged between the cores.

draw from the task (Step 5 in Figure 7.1). The improvement can be either an assignment improvement or a reduction of waste, depending on the worker's decision.

The gap between a requester and a worker regarding a given task's quality requirements can be narrowed by educating the worker through a feedback loop. It increases the crowd-sourcing efficiency if the requester is planning to hire the same workers for the same job, a common practice in crowdsourcing. It is possible to improve the assignment quality by asking the worker to revise the assignment and cover missing quality elements. For instance, an assignment with an insufficient justification can be returned to the worker to revise that justification and get to the next quality class.

Giving the crowd worker a chance before rejecting the submitted assignment achieves two results. First, honest workers who submit a borderline assignment (close to the next quality class) will not waste their time, and a bit of improvement will take their assignment to the next quality level and thus get accepted. Those workers already spent time to learn the CPP process and get themselves familiar with the issue in hand. They might spend extra time searching the web for additional information. Thus, giving them a chance to improve their assignment before rejection is vital to them.

The mechanism also deters workers who are unwilling to submit quality work or at least set their expectations to the right level. For those workers, the gain is to withdraw their assignment and thus save their reputation for not being rejected. AMT uses the "HIT approval rate" of each worker as a qualification to enable additional HITs. This is also a benefit for the requester since it saves time assessing the assignment, and in odd cases, by not polluting the ML model's learning process. Both results help improve CPP efficiency by reducing the number of rejected assignments, and thus, it shortens the time it takes to collect the required number of estimates before calculating a consensus.

However, asking for improvement from crowd workers requires additional effort which needs to be incentivised. Additional money incentives may not work, as stated by more than one study, such as Mason and Watts [239]'s work. It also represents a burden on the CPP process due to the additional costs. Fortunately, research in Behavioural Economics suggests that humans may be encouraged, without incentives, to avoid losses [240]. Thus, framing the improvement request as a form of loss avoidance may be more effective than framing it as a form of incentive, and it then does not require additional cost from the requester. Intuitively, the crowd feedback loop gives crowd workers an understanding of the discrimination between high and low quality work, so that high quality work can be appropriately rewarded.

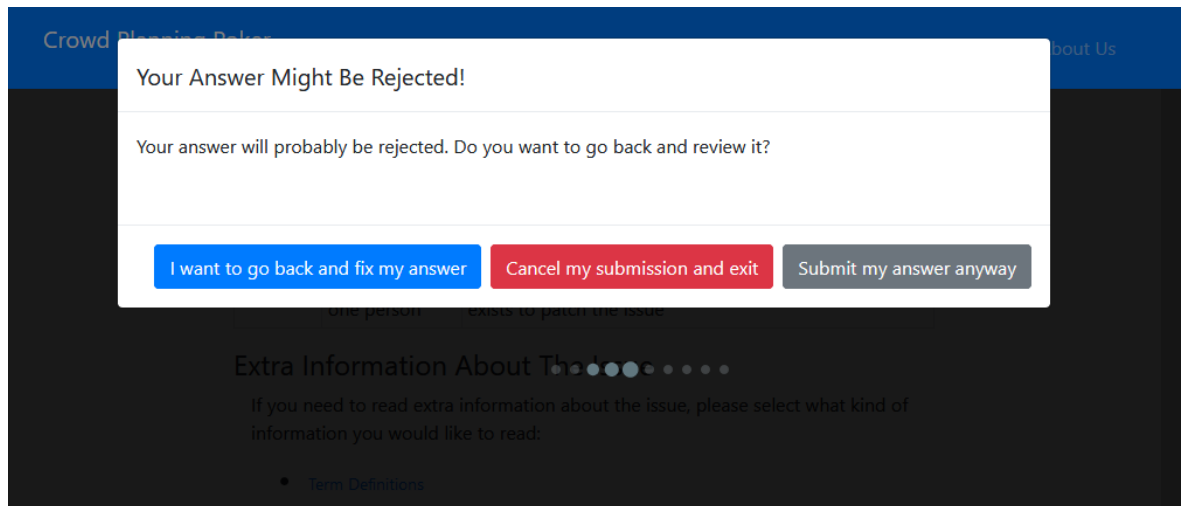


Figure 7.2: Low quality warning message presented to crowd workers who submitted low quality assignments, with three options: review, withdraw, or submit.

### 7.3.2 Encouraging Improvement Using Loss Attention

Recently, several studies [241, 242, 243] have examined loss aversion [240], a behavioural economic theory related to loss attention theory [27]. Loss aversion theory states that humans prefer to avoid losses than receive gains equivalent to the losses. For example, a person prefers not to lose £1 over gaining £1. While loss aversion theory is similar to loss attention, it differs in two points. First, loss attention does not assume equivalency between the losses and the gains. Secondly, loss attention concerns people’s attention, while loss aversion concerns benefits and gains, and thus, loss aversion treats losses as bias while loss attention looks at a loss as a factor to clear biases and increase attention.

Loss Attention theory [27] states that humans are willing to pay additional attention to tasks that involve losses. In the context of crowdsourcing, this section hypothesises that instead of offering an additional monetary incentive to encourage workers to fix their assignment, a warning message will be presented to those low quality assignments, see Figure 7.2. The intuition is to encourage workers to take on additional work (the improvement) to avoid two kinds of losses: the original financial incentive and their reputation (lower HIT approval rate). Yechiam and Hochman [27]’s work says the theory works even on losses as low as \$1.

The message says the assignment is of low quality and might be rejected if the worker does not enhance it. Such feedback is designed to convey two points: directing workers’ attention to the loss, and avoiding misuse and gaming of the ML classifier. First, the message’s language was written to warn the worker about the losses that might be incurred if the worker insists on submitting the assignment as-is. By doing so, the worker is reminded of the loss of the initial monetary incentive and the loss of reputation. Therefore, according to the loss attention theory, the worker should avoid submitting the assignment as-is.

The second point represents a trade-off between the benefits of educating the workers and the risks of exposing classifier weakness. For the case of CPP, the researcher avoided exposing the classifier by not giving more details on which part need to be enhanced. Instead, workers are encouraged to take a thorough review of the assignment and revise the part that feels weak. The reason behind this is that CPP has been designed to be played by distinct crowd workers, and thus, detailed education is not essential since the workers are not retained.

There are three possible responses to such feedback: improving assignment quality, withdrawing from the task, or insisting on submitting the assignment as-is. The first and second actions are equally favourable to the requester and the worker. For the requester, they either increase the number of high quality assignments or decrease the number of low quality assignments. For the worker, they either help in gaining the initial financial incentive or save the worker's reputation if they cannot make the necessary improvement. While the goal of the model is for workers to select one of these two actions, the third action (submit as-is) is enabled for workers since the assessment is ML-based and there is a chance of 27% misclassification with a lower quality class. While those assignments represent a minority, the soft-reject procedure is designed to process them.

### 7.3.3 Handling Rejected Cases Using Soft-Reject

Soft-reject works after an assignment is rejected by the ML classifier. It starts by holding the AMT acceptance decision. Then, the assignment is transferred to the rejection handling repository to be inspected by the researcher. If the assignment merits a rejection (i.e. the classifier decision was right), then the AMT is notified to reject the assignment. Otherwise, the AMT is notified to accept the assignment. Assignments accepted through the soft-reject procedure are not valid for any experimental work since they are assumed to be rejected. The only reason to accept them is to let AMT complete the transaction and the worker receive the financial incentive.

Soft-reject is similar to appeal procedures carried out by organisations in a real-world context. For example, auto loan eligibility software determines whether you can get a loan from a lender. In case of rejection, an appeal can be initiated by the requester, and the lender processes that request manually.

The purpose of soft-reject is to handle auto-rejected assignments. It ensures that workers are rejected for a genuine reason and not because of a flaw in the experimental tools. Soft-reject is not strictly part of the quality model. However, it has been applied since this work has been carried out as experimental work, and crowd workers should not be mistreated because of such a context. Research and experiment contexts have a higher risk factor due to the uncertainty of the applied methods. Further, CPP experiments include humans, and their rights should be reserved.



### 7.3.4 Quality Improvement Experiment Design

After designing the quality model comprising both measurement and improvement, the next step is to validate its assumptions and evaluate its performance. The aim is to investigate the improvement components and measure the reduced waste. Therefore, the experiment will answer the following quality research questions (QRQ):

- QRQ1: Can the model reduce waste (rejected assignments), and how much can be reduced?
- QRQ2: How do crowd workers respond to the improvement request?
- QRQ3: How many unfair treatments (false rejections) are handled by the soft-reject procedure?

The results of the CPP preliminary experiment detailed in Chapter 6 are used as the control for this experiment. Therefore, the design of this experiment is inherited from the CPP preliminary experiment. However, this experiment uses the proposed quality model as the treatment, replacing the manual quality evaluation in the CPP preliminary experiment.

Six issues were picked randomly from the JOSSE dataset (Chapter 4). Table 7.5 illustrates the characteristics of the selected data issues.

Issue Id	Community	NoC	NoW	EEE	AE
FUSED0C-2134	JBoss	7	79	24h	8h
HIVE-4460	Apache	21	64	72h	41h
JBLAB-278	JBoss	2	47	40h	62h
JBPM-153	JBoss	3	27	80h	80h
SLIDER-799	Apache	9	78	24h	24h
WFCORE-1495	JBoss	4	138	32h	28h

Table 7.5: A summary table of the selected issues, showing number of comments(NoC), number of words(NoW) of the issue description, expert-estimated effort (EEE), and actual effort (AE).

The selected issues were used in the similar experimentation settings explained in Chapter 6, Section 6.3. Instead of three trials, one trial was conducted using the six issues. Further, this quality experiment introduced the quality model explained earlier, and thus, the focus of measurement shifted to assignment quality rather than estimate accuracy.

Assignment quality in this experiment was measured using the quality score explained earlier in Section 7.2. In addition, two statistical tests, t-test and chi-square goodness of fit, are used to measure the result significance for waste reduction and quality improvement, respectively.

### 7.3.5 Experiment Results

516 submissions from 712 workers were collected for the six issues. A worker submission is a provisional assignment that is finally submitted. A worker attempt is a provisional assignment that is not submitted but is evaluated by the ML classifier. Table 7.6 details the submissions across the quality classes Excellent (A), Acceptable (B), Unacceptable (C), and Poor (D). The table also lists the attempts' distribution. On average, workers attempted 1.3 assignment revisions, with a maximum of 15 revisions per worker.

Issue ID	NoW	Number of submissions				Submission Attempts Distribution				
		A	B	C	D	Mean	Median	Max	Min	STD
FUSED0C-2134	155	1	60	16	36	1.34	1	7	1	0.93
HIVE-4460	68	0	20	4	20	1.09	1	4	1	0.41
JBLAB-278	99	0	40	7	19	1.25	1	6	1	0.71
JBPM-153	110	0	60	10	25	1.42	1	15	1	1.63
SLIDER-799	148	0	60	13	31	1.3	1	8	1	0.86
WFCORE-1495	132	0	60	18	16	1.22	1	4	1	0.56
Total	712	1	300	68	147					

Table 7.6: Summary of workers' submissions. NoW stands for number of workers.

From the angle of workers, an approval rate of 58.3% has been achieved, meaning that the auto classifier has rejected 41.6% of the workers based on the quality model that is described earlier in Section 7.3. As illustrated in Table 7.7, 43% of workers who submitted assignments below the acceptable quality threshold of CPP responded positively to the model (i.e. chose to either improve their assignments or withdraw from the task): 22% decided to improve their submissions, and 21% decided to exit the Crowd Feedback Loop (CFL).

**QRQ1: Waste Reduction** One of the model goals is to reduce the waste of time and effort for both requester and worker. It can be measured by the reduction in the rejected assignments as a result of workers either improving initial assignments or withdrawing from the task. According to Table 7.7, the total number of workers who attempted to improve their assignments is 157 (22%). However, only 25 (3%) workers were able to make it to the quality threshold; 87 (12%) were not able to pass the quality check but decided to withdraw and save their reputation, and 45 (6%) workers were not able to pass the quality check but insisted on submitting their assignments. That represents a 15% waste reduction, from those who were able to pass the quality check (3%) and those who attempted to improve but failed and withdrew (12%). Adding to that the 154 (22%) workers who decided to withdraw without any attempt to improve their assignments results in a total of 37% reduction in waste.

Issue ID	Number of Workers										
	All	Acc.	Rev.	PQT	Exit CFL				Insist to Submit		
					All	EnI	EaI	DaR	All	SnI	SaI
FUSED-2134	155	61	39	7	51	30	6	15	49	38	11
HIVE-4460	68	20	11	0	24	15	2	7	24	22	2
JBLAB-278	99	40	23	2	37	20	7	10	25	21	4
JBPM-153	110	60	20	3	29	20	2	7	34	26	8
SLIDER-799	148	60	33	6	52	36	5	11	43	32	11
WFCORE-1495	132	60	31	7	48	33	3	12	31	22	9
Total	712	301	157	25	241	154	25	62	206	161	45

Table 7.7: Number of workers across different crowd responses to the improvement component. Acc. stands for Accepted, Rev. for Reviewed, PQT for Passed Quality Threshold, CFL for Crowd Feedback Loop, EnI for Exit with No Improvement, EaI for Exit After Improvement, DaR for Drop after Review, SnI for Submit with No Improvement, and SaI for Submit After Improvement.

The total number of rejected assignments in this experiment is 215 (42%), as illustrated in Table 7.6, compared with 73% rejected assignments in the preliminary CPP experiment for the issues listed in Table 7.5. That represents a 31% reduction in rejected assignments between the two experiments. To examine the significance of the difference in rejected submissions' ratio between issues in preliminary CPP experiment (Table 7.5) and issues in this experiment (Table 7.5), a t-test of unpaired two-samples is applied. It results in  $t = 7.6982$  with a p-value  $< 0.0001$ . The test indicates a statistically significant difference between the two experiments in the percentage of rejected submissions.

**QRQ2: Crowd Response to Improvement Request** One of the model assumptions is that crowd workers can be encouraged by drawing their attention towards possible losses rather than promising additional incentives. As illustrated in Table 7.7, 157 (33%) workers out of 472 attempted to improve their assignments. Crowd workers were successfully encouraged to take the additional effort and improve their submissions up to 15 times, as shown in Table 7.6. However, not all workers were successful in their improvement attempts. Out of the 157 workers, 70 (44%) workers improved their assignments but did not reach the quality threshold; 45 (64%) of them decided to submit their assignments anyway and 25 (36%) of them decided to exit CFL. Out of the 157 workers who attempted to improve their assignments, 62 (38%) workers dropped out of the whole CPP process, and 25 (16%) successfully passed the quality threshold.

The total number of workers who positively responded to the quality model (i.e. attempted to review or withdraw) is 311, whereas only 161 did not respond positively to the model

(i.e. ignored the warning and insisted on submitting). To examine the difference between these two groups statistically, the chi-square goodness of fit test was applied on these two groups. The test resulted in  $X^2 = 47.67$  with a  $p\text{-value} < 0.0001$ . The test indicates a statistically significant difference between those who responded positively and those who did not. Applying the same test but with a further detail introduced by adding the workers who dropped out of the CPP process to those who did not respond positively resulted in  $X^2 = 14.502$  with a  $p\text{-value} = 0.00014$ . This test still indicates a statistically significant difference between those who responded positively from those who did not.

**QRQ3: Handling Unfair Treatment** As part of the soft-reject procedure, all submitted assignments of class C or D were held for a further review, as explained in Section 7.3.3. Those assignments were reviewed manually with the aim of ensuring fair treatment of crowd workers by accepting false negatives. 61 (8%) of submitted assignments were false negatives. These were accepted but not included in the experiment results.

The percentage of false negatives in this experiment and the CPP preliminary experiment are quite similar: 7.9% and 8.2%, respectively. While reviewing all the assignments and evaluating them manually is not feasible, this can be an indication of similar ML performance, as explained in Section 7.2.

## 7.4 Discussion

The first part of the chapter, under Section 7.2, measured crowd quality using a classifier. There was a difference in the classifier performance between finding good crowd assignments and inferior ones. The classifier performed better when classifying assignments with lower quality, and less so with those of higher quality. A similar trend was mentioned for [133]’s classifier. Their classifier was able to predict 96% of low-quality assignments, whereas it predicted 92% of good ones. One reason behind such a trend is that a large number of the bad assignments had the same feature and were easily distinguished. Since those submissions required no effort from crowd workers or were even automated, they came in large quantities. Taking a closer look at the four classes mentioned in Table 7.6, workers submitting class D assignments represented 68% of all rejected workers.

As a result, this trend may affect the classifier by making it greedy in assigning the first quality class (A), as was illustrated in Table 7.6. The fewer “A” class assignments, the less representation there is of them in the training set for the model compared with the other classes. Taking this point from another perspective, the “A” class requires workers to satisfy all the quality dimensions by guessing, which turns out to be quite tricky. In future experi-

ments, a general checklist [237] can be given to workers to help them identify the weak area of their assignment without exposing the classifier weaknesses.

The lack of information about what exactly the quality dimensions were impacted those who submitted low quality assignments and attempted to improve them, but eventually dropped the whole CPP process. This is clearly illustrated in Table 7.7, where 62 workers dropped out of the CPP process after attempting to improve their assignments, representing 39.5% of all workers who attempted to improve their assignments. While the workers are willing to improve, they will not be able to calculate the cost, e.g. how many iterations, until they get approved. That introduces a broader view of the cost of handling misclassification.

A cost model needs to be implemented in real-world scenarios to avoid unfair treatment of crowds and abuse of requester resources. Such a model should find a balance point, where the cost of appealing (treating a false negative) is less than the cost of submitting a high-quality assignment, and higher than the cost of submitting a low-quality assignment. If the cost of the appeal process is zero or near to zero, all rejected workers will appeal, and if the appeal cost is higher than the cost of submitting a high-quality assignment, no one will appeal, including those who were treated unfairly. Both extremes should be avoided, and a cost model needs to be designed to find a balanced cost point that allows workers who are treated unfairly by the classifier to appeal while deterring unengaged workers from abusing the appeal process.

**Assignment Quality vs Worker's Behaviour Score** This quality assessment does not assume that a good worker *always* produces a good outcome, and thus, workers and workers' assignments are assessed independently using user behaviour and estimate justification, respectively. This chapter speculates that good workers do not necessarily perform to their best ability all the time. Good workers may submit a low-quality assignment in some circumstances, e.g. under pressure. On the other hand, deliberate malice may be able to craft an assignment that is very hard to distinguish from the legitimate assignment using machines only. In such odd cases, methods that rely only on assessing workers' behaviour or assignment quality will not discover them. Table 7.8 shows the number of workers who submitted good quality assignments (classes A and B) and bad quality assignments (classes C and D) versus their behaviour scored as good behaviour ( $\geq 4$ ) or bad behaviour ( $< 4$ ).

As illustrated in Table 7.8, 27% of workers behave in contrast to their assignment quality, i.e. workers with good behaviour submitted bad assignments and vice versa. Judgement based on workers' behaviour only (293 workers) is more generous than judgement based on assignment quality only (274). Blending both mechanisms may give a better understanding of the overall work quality, and so the quality model in this chapter is designed accordingly.

	Assignment			
		Good	Bad	Total
Behaviour	Good	213	80	293
	Bad	61	162	223
	Total	274	242	516

Table 7.8: Number of workers who submitted good quality assignments (Class A and B) and bad quality assignments (Class C and D) versus good worker's behaviour score ( $\geq 4$ ) and bad worker's behaviour score ( $< 4$ )

**Threats to Validity** Possible threats to validity may come from the selected software development issues. They are all from open-source communities and their information is publicly available. However, this is mitigated by randomly selecting issues that vary in the required efforts (8h–80h), description length (27–138 words), developer interactions (2–21 comments), and source community (JBoss and Apache), see Table 7.5. At the same time, those issues demonstrate similar features to those selected for the CPP preliminary experiment where the classifier ground truth is developed. All the data are real-world data and thus minimise risk to the construct validity.

Workers who submitted low-quality assignments are presented with a warning message of possible losses with three options to select from. To minimise possible bias of the interface on workers' decisions, all options are presented equally in terms of their size and fonts. However, it was important to distinguish them using different colours. Thus, blue was used for the review option, red was used for the withdrawal option, and grey was used for submission. Figure 7.2 shows the CPP interface with the warning message along with the three options for crowd workers to select from.

Table 7.7 shows that 33% of workers were encouraged to review their assignments. This contradicts the initial assumption that says encouragement needs incentives. Applying the chi-square goodness-of-fit test with this assumed distribution, by allocating a minority of workers to review their assumptions (10%) and the rest of the workers equally divided between withdrawing and submitting their assignments, (10% for review, 45% for withdraw and 45% submit), resulted in  $X^2 = 283.92$  with a  $p\text{-value} < 0.0001$ . The test indicates a statistically significant difference between the original assumption (no one encouraged to review without incentives) and the actual results. However, it is not clear whether such an uptake is because of the message that is framed to attract workers' attention to possible losses or simply because the option is available. The workers were equally distributed across the three available options: review, withdraw, and submit. Thus, there is not enough evidence to decide whether the loss-attention strategy has worked in this situation.

Conclusions regarding crowd quality measurement, waste reduction, and assignment im-

provement are considered after applying corresponding statistical tests on unbiased measures including ROC-AUC and Recall for quality measurement, number of total rejected assignments for waste reduction, and the number of workers selecting improvement options (review or withdraw) for measuring the effect of encouraging workers to do better without additional incentives.

Threats to external validity is also mitigated by incorporating issues from different open-source projects. While the issues vary, they do not represent all kinds of software. For instance, commercial software projects, where the tightest time management is expected, are not included. Further investigation of such systems and projects is needed.

Now that the quality of crowd work can be measured and improved, the next step is to go back and investigate the crowd's ability in predicting reliable estimates for software development tasks in a more extended experiment, perhaps involving issues of a different nature, such as bug fixing and software enchantment tasks, to minimise threats to CPP's external validity.

## 7.5 Summary

While controlling the quality of work in a closed environment, e.g. a company with known contractors, is relatively manageable, it is challenging in an open crowdsourcing market such as AMT. Such an environment is filled with pitfalls, including quality assessment, subjectivity, and fraudulence, leaving crowd employers with mostly useless outcomes. However, crowdsourcing markets are the best at providing accessible, flexible, and cheap access to human capital around the clock.

A quality management suite consisting of three components is illustrated and detailed throughout this chapter. It starts with quality definition, where six quality dimensions are defined and specified for the CPP process. Then, an assessment component explains different assessment strategies and concepts suitable for the context of CPP. The third component deals with quality improvement for CPP assignments and details how waste can be reduced through a lean-inspired method.

As an application of the quality management suite, a quality model for CPP is proposed. It consists of five elements that reflect the concepts explained in the suite. It also highlights the ethical concern regarding using human crowds in experimental work, and explains how that is treated using a special handling scheme.

Moreover, two experiments were designed and conducted to evaluate the proposed CPP quality model. The first experiment investigated the performance of the assessment classifier. It found that the classifier performance is consistent with a 93%  $F_1$  score. The second experiment examined and observed crowd workers' response to the improvement element of the

model. Its results suggested that 70% of the rejected workers responded positively to the improvement element by improving their assignment (37%) or withdrawing from the task (32%).

While the two experiments suggest positive results, the next chapter will show more extended experimental work to ensure the generality of the quality model and its performance for a large number of issues and a wider range of issue types.



## Chapter 8

# Playing Planning Poker in Crowds: Human Computation of Software Effort Estimates

Chapter 6 introduced Crowd Planning Poker (CPP) as an estimation method based on the implementation of the Agile Planning Poker practice in a crowd, and Chapter 7 addressed the subsequent assignment quality issues that arose. This chapter contributes through an extended evaluation of the full design of Crowd Planning Poker on a diverse range of issues. Thirty of the issues had not previously been used in experiments. They comprise feature requests, software enhancements, and bug fixes, from three different open-source projects. In total, 80 Crowd Planning Poker rounds were executed and 807 estimates were received. Actual effort for task completion reported in the issue repositories ranged from half a day through to two weeks. In addition, this chapter details the application that was developed to execute the CPP process and orchestrate worker activity on the Amazon Mechanical Turk (AMT) platform.

This chapter is organised as follows. The next section reviews the involvement of human computation in playing CPP along with its challenges and the development of a system that can orchestrate CPP. Then Section 8.2 illustrates the experimental design to evaluate the automated version of CPP. It explains the dataset used and selected issues, the CPP workflow used, and how the quality model is implemented. Next, Section 8.3 lists the experiment results along with the result evaluations. It shows the performance of CPP for software effort estimation (SEE) and the scalability of the process. Then, Section 8.4 discusses the experiment results in the light of additional benefits of CPP and possible threats to validity. The last section summarises this chapter and traces the direction of the next step in this thesis.

## 8.1 Full Design of Crowd Planning Poker Using Human Computation

As explained in the thesis introduction (Chapter 1) and reviewed in the Literature Review (Chapter 3), human computation is an emerging discipline with a variety of promising applications. The proliferation of crowdsourcing platforms enabling access to human workers on platforms such as AMT has industrialised this mechanism. This section explains the details concerning human computation in the revised CPP. The introduction of human computation in CPP is to automate the auditing of crowd assignment quality and to help orchestrate CPP without requiring the researcher to intervene in the CPP process.

The notion of human computation is that machines in some computation processes are not capable of taking some decisions, and thus, they need a human to compute that piece of the process. Human Computation as a discipline studies the involvement of humans in a machine computation process without interrupting or changing the context or direction of the process. Since humans are the cornerstone of human computation, crowdsourcing works very well with it by providing easy and flexible access to human capital. Similarly, large-scale development efforts such as open-source communities will be able to use CPP in estimating their projects and issues. Organisations do not necessarily need to hire crowds outside their communities; CPP can be used within an organisation, using its own human capital. CPP also could help in slightly different applications such as task triage, which is currently challenging for open source communities, as Hooimeijer and Weimer [244] noticed in their overview:

“For software that is widely deployed, the number of bug reports typically outstrips the resources available to triage them.”

The CPP process presented in this chapter is similar in many aspects to the process described in Chapters 6 and 7. However, the revised process is described in full to present a single view of the whole process used for the full experiment, and to address the human computation aspects of the process.

Thinking of CPP as an automated process, a machine needs to take decisions about the crowd’s assignment quality and the estimation process along with the process orchestration tasks such as hiring qualified crowd workers. However, machines might not be as good as humans at linking previous experiences and providing estimates of a given task. Thus, the machine involves humans to compute what is necessary without handing the process to the humans or changing the overall process. Chapter 7 starts the process by merging human and machine computations in the quality model, and this chapter will draw a complete picture of the automated CPP using human computation in different CPP aspects after addressing possible challenges.

In the pilot experiments (Chapter 6), quality assessment was a manual process and there was no option for crowd workers to improve their assignment. Submitted assignments were assessed manually using a legitimacy score. The manual handling of quality prevented the experiment from engaging crowd workers in a feedback loop, and thus the quality improvement step needed to be conducted separately by posting additional Human Intelligence Tasks (HITs) to fill the gap created by rejected assignments. Such handling of quality results in extra financial costs and takes a prolonged period of time.

Additionally, running CPP with a large number of software development tasks, as intended in this chapter's experiment, demands an automated orchestration of all CPP activities, including the quality assessment. For instance, HIT configuration, recruiting crowd workers, and deciding when an estimation round is mature were done manually in previous experiments. Sometimes, a HIT got posted more than once when no one picked it or the hired worker submitted unacceptable assignment quality.

Therefore, quality evaluation in earlier experiments in Chapter 6 was conducted manually by the researcher and employed crowd workers for the purposes that are stated in their design. While it shows an encouraging result in that regard (using the crowd), the process was unpractical from the side of the requester (the researcher). Human computation can take over the requester role and start executing and administering the process automatically, including crowd management. Therefore, the earlier version of CPP was using a human-to-human process, whereas the revised version uses a machine-to-human process to support automation.

To implement the machine-to-human process, a CPP web application was developed to communicate with AMT and provide a customised interface for crowd workers to carry out the CPP process. The CPP application was implemented using Python. It implements CPP components including administration, quality, and AMT integration using AMT's Application Programming Interface (API). It also uses the Django framework to provide a web interface access for the researcher to enter issues, observe the process, and retrieve results. CPP is integrated with AMT to smoothly handle the workers' transitions between the application and AMT as if they are a single system. The development of the CPP application was essential in order to implement the quality model explained in Chapter 7, especially the improvement step where a crowd worker needs to be engaged in the Crowd Feedback Loop (CFL), which is beyond AMT's capabilities. Further, the CPP web application is designed to collect a log of the workers' interactions with the CPP user interface, which cannot be done using the AMT platform alone. Figure 8.1 illustrates the 15 activities of CPP using Business Process Model and Notation (BPMN), along with their operators, and the rest of this section will explain the process activities.

#### *HIT Configuration.*

Starting with the configuration before recruiting crowd workers, there are 15 configuration

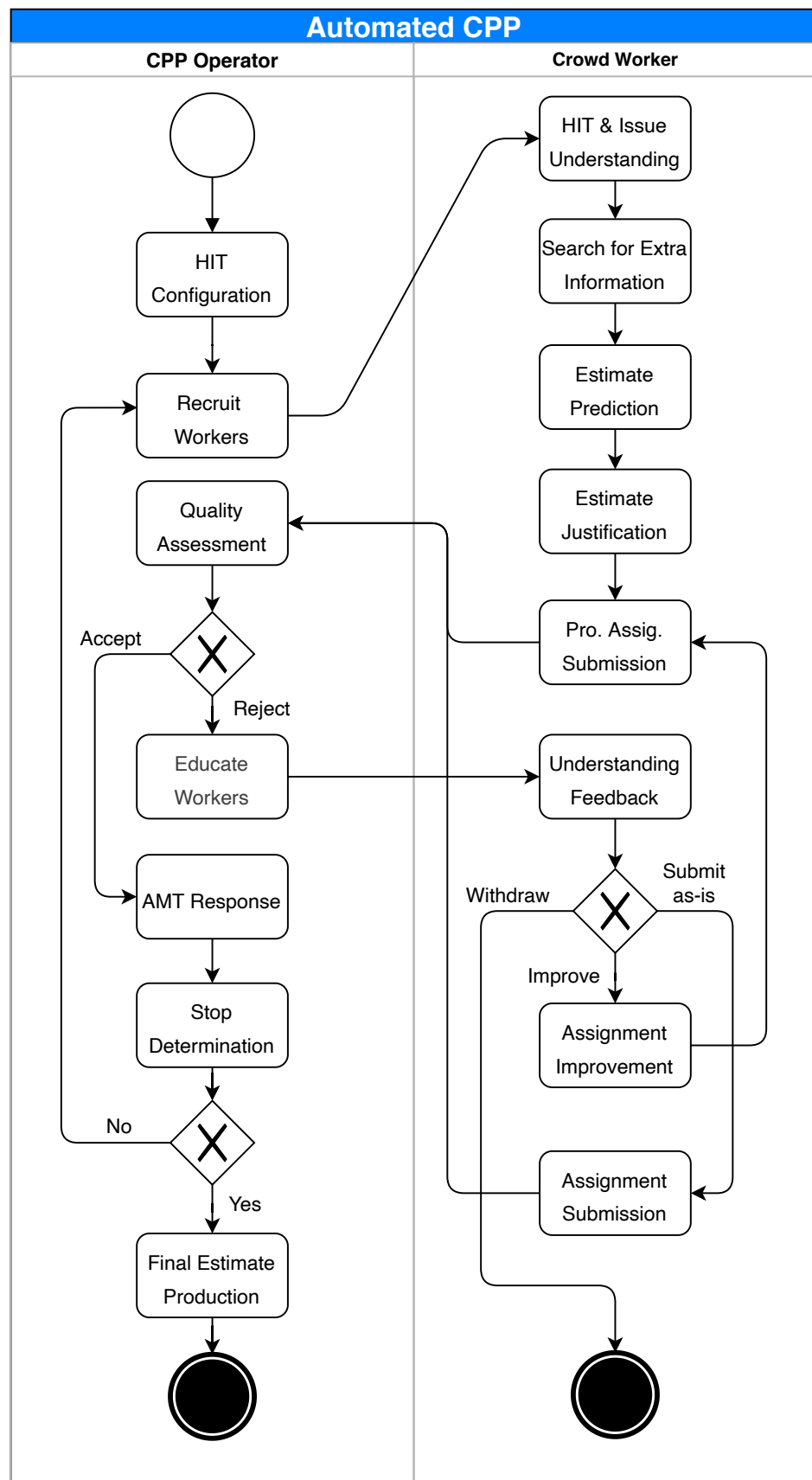


Figure 8.1: This is an illustration of a model process using Business Process Model and Notation (BPMN). Pro. Assig. stands for Provisional Assignment.

The screenshot displays the Amazon Mechanical Turk (AMT) interface. At the top, there's a navigation bar with 'amazonmturk', 'HITS', 'Dashboard', and 'Qualifications'. Below this, a search bar shows 'timing'. The main content area is titled 'HIT Groups' and shows a list of HITs. The first HIT is titled 'Timing a software development issue' by requester 'Mohammed Alhamed'. It has 17 HITs, a reward of \$0.23, and was created 1m ago. The description states: 'You need to estimate how much time a software development issue will take to be accomplished. You will be given the issue title and its description. Then, you will make an educated guess about how much person-hours it will take. Finally, you shall write a justification for the estimate you have chosen. You might be required to read other workers estimates and their reasons to guide your educated guess.' The time allotted is 7 Min 30 Sec, and it expires in 1d. Qualifications required include: 'Previous Participant has not been granted' (None), 'HIT approval rate (%) is greater than 70' (100), 'Research Study Consent for University of Glasgow is 1' (None), and 'Software Development Experience is 1' (None). Below this, there are three more HITs from 'Crowdsurf Support': 'Timing review - Earn up to \$0.18 per timed media minute' (6 HITs, \$0.18, 16s ago), 'Spanish: Timing review - \$0.18 per media minute' (3 HITs, \$0.18, 2d ago), and 'Special Handling - Entertainment Timing - Earn up to \$0.20 per media minute' (1 HIT, \$0.20, 1d ago). The bottom of the page shows a footer with 'Help', 'Contact', 'Legal', 'Service Health', 'Feedback', and '© 2008-2018 Amazon Mechanical Turk Inc. or its affiliates. All rights reserved.' and 'An amazon company'.

Figure 8.2: Screenshot of AMT showing CPP HIT specifications.

parameters [245] that need to be defined and specified. According to AMT, most of these do not need to change during the process and can use templates to set them, such as “Description” and “LifetimeInSeconds”. However, other parameters are dynamic and need to be managed dynamically during the execution of the CPP process, such as “MaxAssignments” where the number of required workers is defined.

#### *Crowd Worker Recruitment.*

Next, the requester posts the HITs and then the AMT lists the HITs on the platform, see Figure 8.2 for an example of a CPP HIT along with other HITs. After posting a HIT, the system waits for crowd workers to pick the HIT and accept the HIT terms. Only qualified workers can accept the HIT, and all the HIT qualifications are listed beside the HIT description, as shown in Figure 8.2

#### *HIT Understanding.*

After a qualified worker picks a HIT, the worker first reads the HIT instruction (Figure 8.3), and then initial information about the issue (title and description) is presented to the crowd worker via a web-based user interface. The worker reads and analyses the targeted software development issue. Ideally, the worker’s analysis of the issue description results in a breakdown of the issue into smaller components that can be easily estimated according to the worker’s knowledge.

#### *Search for Knowledge.*

The worker may need to look for additional information that can help in sizing the problem. CPP offers three additional pieces of information about the issue which the worker can opt to look at. CPP offers a dictionary, a project overview, and development team comments,

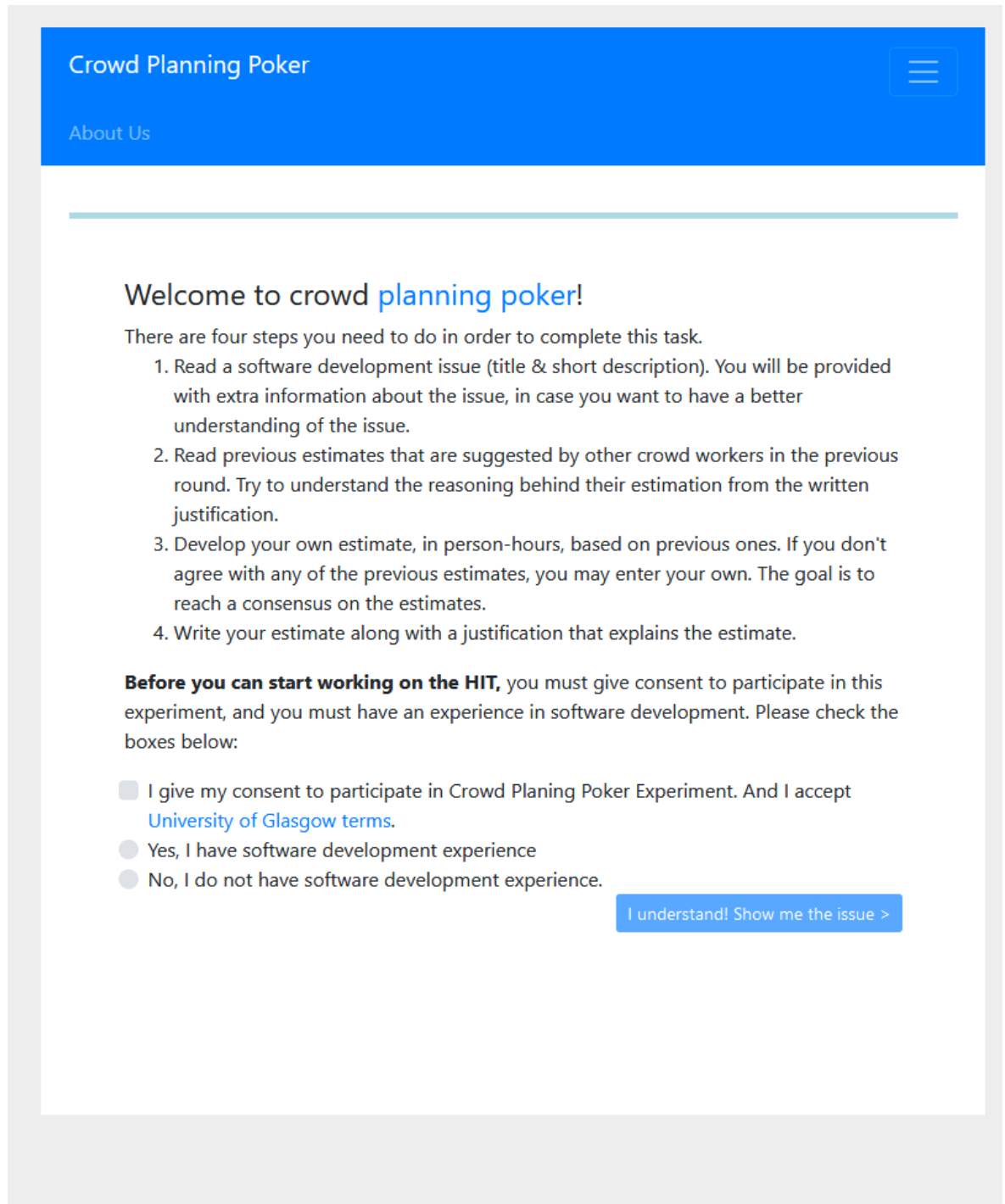


Figure 8.3: HIT instructions as they are listed in the CPP application.

see Figure 8.4. This information includes contextual project details, such as definitions of ambiguous terms and abbreviations, or more information about project-specific terms, such as the name of a software component that appears in the description. The crowd worker can also access comments that were posted on the issue. Further information can be searched by the developer using a search dialog provided on the user interface.

#### *Estimate Prediction.*

After having accessed the issue information and its background, the worker can move on to the next step by selecting an estimate category that fits the worker's estimate for effort, as shown in Figure 8.4.

#### *Explain The Rationale.*

Right after picking an estimate category, the worker is asked to justify the selected estimate. Ideally, the worker will list the sub-components from the analysis along with how much effort the worker predicts for those components. Additional comments about how to resolve the issue or what tools and technology might be helpful show that the worker is fully engaged. The worker can write the justification for selecting such an estimate in the text box shown in Figure 8.4.

#### *Assignment Submission.*

Now the worker can submit the assignment to the CPP system. This is different from the AMT assignment submission request. Before communicating with the AMT API, CPP receives the assignment as a provisional version that needs to be assessed, and it could be different from the final one.

#### *Quality Assessment.*

While the worker is waiting for a response from the web browser, the CPP process conducts a quality assessment on the worker assignment. The assessment happens in real time. After quality assessment, CPP automatically annotates the assignment with the proper quality class based on the submitted provisional version of the assignment. The quality assessment outcome is either to accept or reject the assignment, as detailed in Chapter 7

#### *Quality Feedback.*

If the assignment's quality is below a certain threshold, a feedback step is taken by communicating the evaluation result to the worker. By then, the worker is engaged in the CFL, as explained in Chapter 7.

#### *Understanding Feedback.*

The worker receives the feedback if the assignment is classified as low quality, as illustrated in Figure 8.5. After understanding the feedback, the worker is offered three options: 1) improve the assignment, 2) withdraw from the process, or 3) submit the assignment as-is.

Crowd Planning Poker

<b>Middle</b>	One day for one person	The change that probably is needed has already be done elsewhere. Testing is what will require time.
<b>Lowest</b>	One day for one person	Need to review and test the Pull Request that already exists to patch the issue

### Extra Information About The Issue

If you need to read extra information about the issue, please select what kind of information you would like to read:

- [Term Definitions](#)
- [Issue Comments](#)
- [About the issue project](#)

I want to Google it

---

Please, select one of the estimates below:

☐ A day

☐ Two weeks

☐ Half a week

☐ Half a day

☐ An hour or less

☐ A week

☐ ⋮

☐ More time

Write your justification and comments:

I can not estimate

Submit my estimate and finish

Figure 8.4: List of extra information that includes term definitions, development team comments, and development project brief. Crowd workers also can do a web search using Google. This screen also lists possible estimate options for the worker to pick from and a text box to write a justification for selecting such an estimate.



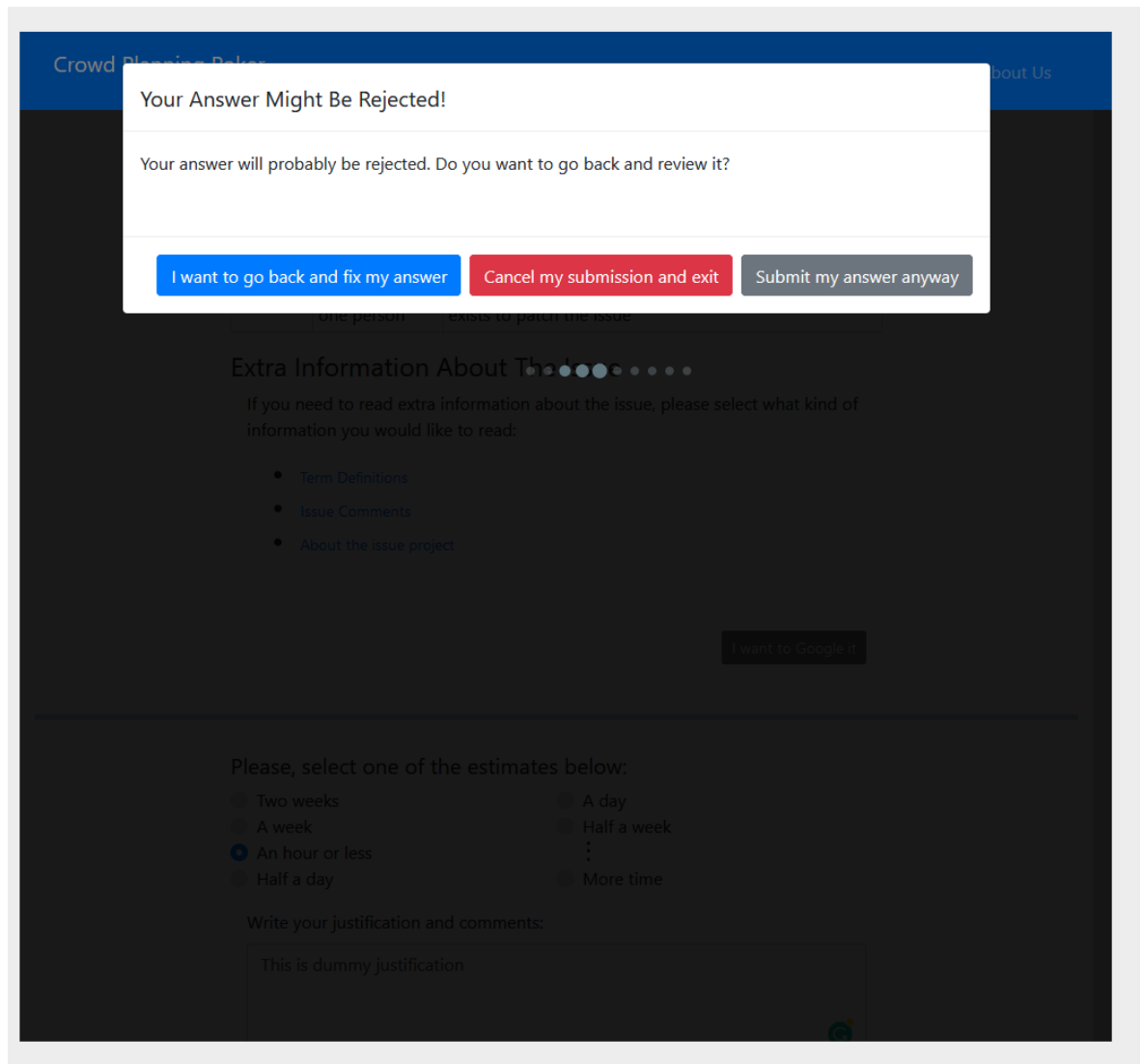


Figure 8.5: Low quality warning message presented to crowd workers who submitted low-quality assignments, with three options: review, withdraw, or submit.

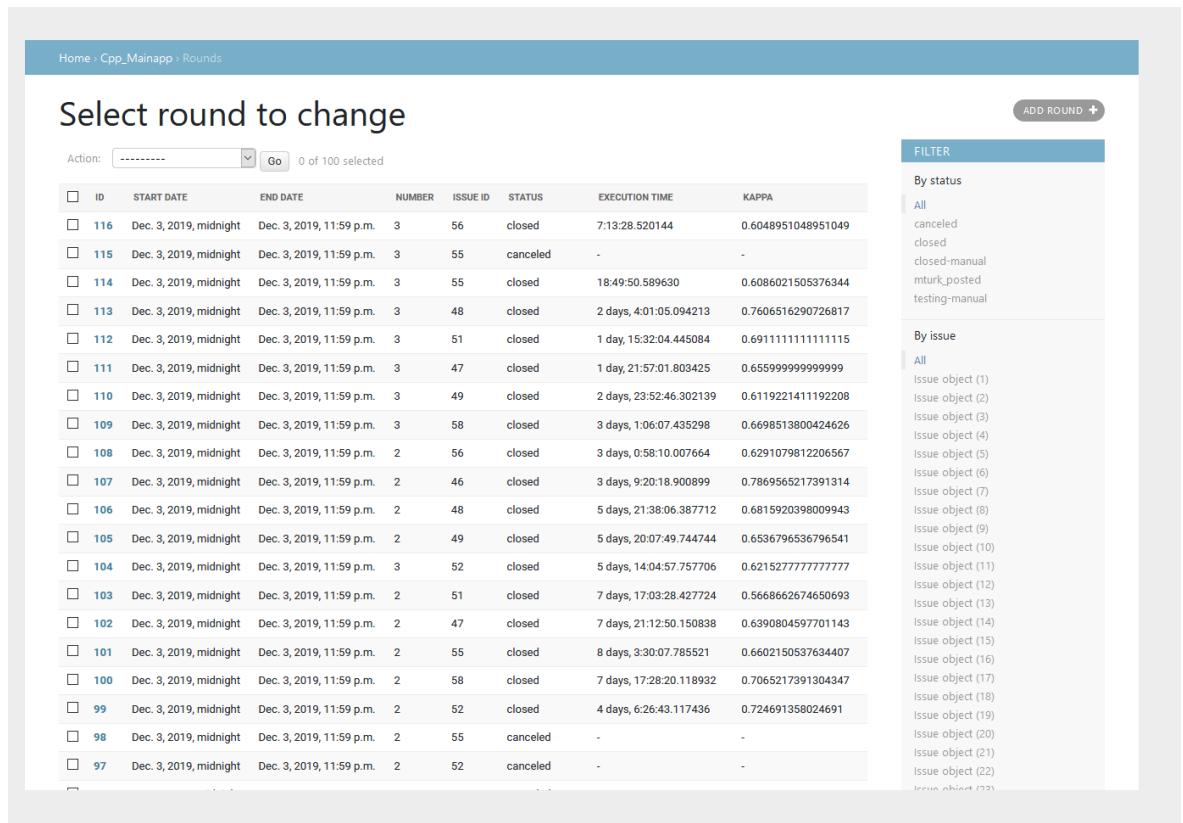
Withdrawing from the process or submitting the assignment as-is will terminate the process for the worker and mark the assignments accordingly.

#### *Assignment Improvement.*

If the worker decides to improve the assignment, the worker will be returned to the previous screen to add any modification to the estimation and/or justification. The worker can work in the CFL until the desired quality is reached.

#### *AMT Response.*

Either by submitting a provisional assignment with the required quality or by exiting CLF, an AMT response request is initiated to accept or reject the assignment on the AMT platform and CPP system. In case of approval, the process approves two aspects: a CPP approval to include the assignment in determining the final estimate, and an AMT approval to pay the



Home » Cpp\_Mainapp » Rounds

### Select round to change

Action:  Go 0 of 100 selected

<input type="checkbox"/>	ID	START DATE	END DATE	NUMBER	ISSUE ID	STATUS	EXECUTION TIME	KAPPA
<input type="checkbox"/>	116	Dec. 3, 2019, midnight	Dec. 3, 2019, 11:59 p.m.	3	56	closed	7:13:28.520144	0.6048951048951049
<input type="checkbox"/>	115	Dec. 3, 2019, midnight	Dec. 3, 2019, 11:59 p.m.	3	55	canceled	-	-
<input type="checkbox"/>	114	Dec. 3, 2019, midnight	Dec. 3, 2019, 11:59 p.m.	3	55	closed	18:49:50.589630	0.6086021505376344
<input type="checkbox"/>	113	Dec. 3, 2019, midnight	Dec. 3, 2019, 11:59 p.m.	3	48	closed	2 days, 4:01:05.094213	0.7606516290726817
<input type="checkbox"/>	112	Dec. 3, 2019, midnight	Dec. 3, 2019, 11:59 p.m.	3	51	closed	1 day, 15:32:04.445084	0.6911111111111115
<input type="checkbox"/>	111	Dec. 3, 2019, midnight	Dec. 3, 2019, 11:59 p.m.	3	47	closed	1 day, 21:57:01.803425	0.6559999999999999
<input type="checkbox"/>	110	Dec. 3, 2019, midnight	Dec. 3, 2019, 11:59 p.m.	3	49	closed	2 days, 23:52:46.302139	0.61192214111192208
<input type="checkbox"/>	109	Dec. 3, 2019, midnight	Dec. 3, 2019, 11:59 p.m.	3	58	closed	3 days, 1:06:07.435298	0.6698513800424626
<input type="checkbox"/>	108	Dec. 3, 2019, midnight	Dec. 3, 2019, 11:59 p.m.	2	56	closed	3 days, 0:58:10.007664	0.6291079812206567
<input type="checkbox"/>	107	Dec. 3, 2019, midnight	Dec. 3, 2019, 11:59 p.m.	2	46	closed	3 days, 9:20:18.900899	0.7869565217391314
<input type="checkbox"/>	106	Dec. 3, 2019, midnight	Dec. 3, 2019, 11:59 p.m.	2	48	closed	5 days, 21:38:06.387712	0.6815920398009943
<input type="checkbox"/>	105	Dec. 3, 2019, midnight	Dec. 3, 2019, 11:59 p.m.	2	49	closed	5 days, 20:07:49.744744	0.6536796536796541
<input type="checkbox"/>	104	Dec. 3, 2019, midnight	Dec. 3, 2019, 11:59 p.m.	3	52	closed	5 days, 14:04:57.757706	0.6215277777777777
<input type="checkbox"/>	103	Dec. 3, 2019, midnight	Dec. 3, 2019, 11:59 p.m.	2	51	closed	7 days, 17:03:28.427724	0.5668662674650693
<input type="checkbox"/>	102	Dec. 3, 2019, midnight	Dec. 3, 2019, 11:59 p.m.	2	47	closed	7 days, 21:12:50.150838	0.6390804597701143
<input type="checkbox"/>	101	Dec. 3, 2019, midnight	Dec. 3, 2019, 11:59 p.m.	2	55	closed	8 days, 3:30:07.785521	0.6602150537634407
<input type="checkbox"/>	100	Dec. 3, 2019, midnight	Dec. 3, 2019, 11:59 p.m.	2	58	closed	7 days, 17:28:20.118932	0.7065217391304347
<input type="checkbox"/>	99	Dec. 3, 2019, midnight	Dec. 3, 2019, 11:59 p.m.	2	52	closed	4 days, 6:26:43.117436	0.724691358024691
<input type="checkbox"/>	98	Dec. 3, 2019, midnight	Dec. 3, 2019, 11:59 p.m.	2	55	canceled	-	-
<input type="checkbox"/>	97	Dec. 3, 2019, midnight	Dec. 3, 2019, 11:59 p.m.	2	52	canceled	-	-

**FILTER**

**By status**

- All
- canceled
- closed
- closed-manual
- mturk\_posted
- testing-manual

**By issue**

- All
- Issue object (1)
- Issue object (2)
- Issue object (3)
- Issue object (4)
- Issue object (5)
- Issue object (6)
- Issue object (7)
- Issue object (8)
- Issue object (9)
- Issue object (10)
- Issue object (11)
- Issue object (12)
- Issue object (13)
- Issue object (14)
- Issue object (15)
- Issue object (16)
- Issue object (17)
- Issue object (18)
- Issue object (19)
- Issue object (20)
- Issue object (21)
- Issue object (22)

Figure 8.6: Screenshot of one of the CPP admin screens. It shows a list of rounds along with the round details, including how much time a round has taken, the crowd consensus (Fleiss' Kappa %), round number, and status.

worker for the assignment.

#### *Process Stop Determination.*

Then, CPP checks whether the workers have reached a consensus or the maximum number of iterations has been reached to determine whether another CPP round is necessary. If a further round is required, the crowd workers are offered a summary of the low, median, and high estimates from the previous round, along with the justifications provided. The provision of this supplementary information mimics the design of in-person Planning Poker. *Seed answer* refers to the summary of the previous round that is fed back to the crowd workers, in a similar way to the discussion that takes place in in-person Planning Poker. If the CPP limits are reached, the process stops recruiting more workers and deletes any HITs listed at AMT. Figure 8.6 shows a monitoring screen for CPP rounds. It lists detailed information for each CPP round, including how much time a round has taken, the crowd consensus (Fleiss' Kappa %), round number, and status.

#### *Final Estimate Production.*

After that, CPP aggregates the estimates from the final round which have already been narrowed by the workers during the CPP rounds. Figure 8.7 shows a screenshot of the dashboard

where the issues are listed after the final estimate is produced. The green rows represent the issues where crowd workers produced estimates more accurate than or similar to the experts' estimates, and the red rows are for issues with crowd estimates worse than the experts' estimates.

## 8.2 Experimental Design

Earlier experiments of CPP in Chapter 6 and Chapter 7 were designed and conducted to serve subtopics of the CPP design and experiment with its quality management options. Also, they were conducted using a limited number of issues. Although those experiments helped determine working settings for CPP and showed promising results for its efficiency and reliability, a more rigorous experiment with an extended number of issues needed to be undertaken. The aim was to confirm earlier conclusions and extend the method applicability to a wider range of issues. Therefore, this chapter aims to find answers to the following CPP research questions (CRQ):

1. CRQ1: Given a software task that required between half a day and two weeks of effort, are crowdsourced effort estimates, which are produced by the automated CPP, of comparable accuracy to those of experts?
2. CRQ2: How does the cost (money and time) of crowdsourced estimates compare with those produced by small groups of experts?

This section details an experimental design to compare the performance of a crowd in producing software task estimates with those produced by project expert estimation. It starts with a description of the software tasks that formed the objects of the experiment. It then describes the automated CPP adaptation of Planning Poker and our technique for filtering estimates provided by the crowd workers based on the quality of an associated justification and their behaviour. In addition, it reviews the outcome of the third preliminary experiment that assisted in the design of the CPP process.

The purpose of the experiment is to determine whether the CPP process performed by crowd workers and *orchestrated by a machine* can produce estimates comparable to those of experts. Therefore, it was necessary to obtain a set of software tasks that had been annotated with both an expert-estimated and an actual effort, providing an experimental baseline and a ground truth, respectively. The JOSSE dataset was found to satisfy these criteria.

After filtering the JOSSE dataset, 419 issues were found that had been annotated with both an expert time estimate and actual time spent in person-hours. Although the communities behind those issues have published their issue reporting documentation, the research was not

NUMBER OF THE TOTAL ISSUES	BETTER CROWD ESTIMATES	COMPARABLE CROWD ESTIMATES	WORST CROWD ESTIMATES	HOW BAD IS THE CROWD ESTIMATE	HOW BAD IS THE EXPERT ESTIMATE
10	2	4	4	130.0%	80.0%

ISSUE #	ROUNDS	EST. RECEIVED	EST. CONSIDERED	EST. DISCARDED	AGGREGATED CROWD ESTIMATE (MRE%)	EXPERT ESTIMATE (MRE%)	ACTUAL EFFORT	CROWD AGREEMENT
46	2 Round/s	65	39	26	[20.0] Half a week(100%)	[40.0] One week(0%)	[80]Two weeks	72.04%, 78.7%
47	3 Round/s	97	60	37	[20.0] Half a week(100%)	[24.0] Half a week(100%)	[80]Two weeks	71.32%, 63.91%, 65.6%
48	3 Round/s	95	60	35	[20.0] Half a week(100%)	[80.0] Two weeks(0%)	[80]Two weeks	70.89%, 68.16%, 76.07%
49	3 Round/s	102	56	46	[40.0] One week(0%)	[80.0] Two weeks(100%)	[40]One week	66.99%, 65.37%, 61.19%
50	1 Round/s	44	19	25	[20.0] Half a week(0%)	[72.0] Two weeks(100%)	[40]One week	76.64%
51	3 Round/s	97	59	38	[8.0] A day(100%)	[8.0] A day(100%)	[4]Half a day	61.94%, 56.69%, 69.11%
52	3 Round/s	101	59	42	[20.0] Half a week(400%)	[8.0] A day(100%)	[4]Half a day	65.51%, 72.47%, 62.15%
55	3 Round/s	104	51	53	[40.0] One week(400%)	[24.0] Half a week(200%)	[8]A day	64.08%, 66.02%, 60.86%
56	3 Round/s	85	41	44	[20.0] Half a week(0%)	[24.0] Half a week(0%)	[20]Half a week	68.22%, 62.91%, 60.49%

Figure 8.7: The screenshot shows a list of software development issues (a row for each issue) along with expert and crowd estimates. The green rows represent the issues where crowd workers produced estimates more accurate than or similar to the experts' estimates, and the red rows are for issues with crowd estimates worse than the experts' estimates.

able to determine exactly how the estimated or actual times were produced. After reaching out to several community team leaders to determine the exact estimation process, the researcher realised that the estimation process depends on their experience and there is no structured process to follow for producing software estimates.

Moreover, the issue history log shows that estimated effort had been determined by one or more of the issue assignees. Therefore, the estimated effort reported by the development team on the issue is referred to as an expert estimate in this study (see Section 4.3 in Chapter 4).

This experiment follows the previous design of using a *relative* unit for effort estimation instead of using literal person-hours; extra details have been illustrated in Chapter 6. The units of effort were labelled as one hour, half a day, one day, half a week, one week, two weeks, and more than two weeks. Therefore, the reported efforts in the issues were translated into those categories, based on an 8-hour day and 40-hour working week. This enabled a comparison between the CPP estimates and the person-hour costs reported on the issues.

Further narrowing filtered down the population to 126 issues, to avoid issues that:

- required less than 30 minutes or more than two weeks to complete;
- contained less than 20 words in the description and were assumed to be too vague; or
- had received no comments and so were assumed to not be of interest to the community.

Thirty (30) issues were randomly selected from the filtered data set for use in the experiment. To ensure that a diverse range of effort magnitudes were included, issues were first organised into effort categories ranging from one hour through to two weeks, as described above. Issues were then selected randomly from these categories for inclusion in the sample. URLs for the selected issues were obtained from the JOSSE dataset. Issues selected were found to comprise a mixture of bugs, feature requests, and enhancements.

Before proceeding, it was necessary to check whether the expert estimates for the selected issues were representative of the whole data set. The selected sample might represent an artificially low baseline if the estimates they contain are less accurate than those for the population of issues as a whole. To do this, the mean absolute error, median magnitude of relative error, and mean magnitude of relative error were calculated for the three sets of issues (all estimated issues, filtered issues, random sample), as shown in Table 8.1. The results show that the average estimation performance by experts in the sample is slightly better than for the whole or filtered set of issues. This assessment demonstrates that the selected baseline (the expert estimates in the sample) is suitable for use in the study.

Crowd workers (the study subjects) were recruited from the Amazon Mechanical Turk platform [246]. Only workers with a self-declared experience of at least two years of software

	#	Mean Absolute Error (hours)	Median Magnitude of Relative Error	Mean Magnitude of Relative Error
All	419	$\pm 29.3$	128%	2475%
Filtered	129	$\pm 12.0$	100%	773%
Sample	39	$\pm 10.5$	90%	440%

Table 8.1: Comparison of estimation error metrics of the whole population (419 issues), filtered set of issues (126), and selected sample (30).

engineering were permitted to participate. Each estimation session employed a group of between 5 and 15 workers.

## 8.3 Result and Evaluation

Thirty trials were conducted (one per selected issue), as summarised in Table 8.2. All trials proceeded until an ‘Almost Perfect’ level ( $>0.75$ ) [247] of agreement was reached amongst the crowd workers, measured using Fleiss’ Kappa [221]. The crowd workers reached a consensus within three rounds in all trials, with nine trials ending after a single round, ten trials ending after two rounds and eleven trials requiring three rounds of estimation.

Each round of CPP received between 10 and 5 estimates, with an average of 8 estimates received in each round, resulting in a total of between 5 and 30 estimates for each trial. Each round was kept open until a minimum of five estimates of sufficient quality had been received. Unlike the earlier pilot study, the proportion of rejected estimates was much lower, averaging 39% across all trials and reaching 50% in Trial 2 and Trial 9. The reduction in low-quality submissions is likely due to the automatic quality assessment and feedback process explained in Chapter 7.

The table also shows a comparison between the final aggregate estimate produced by the crowd for each trial and the expert (baseline) estimate and actual effort (ground truth) for the issue as reported in the source project’s issue tracker. The category (one hour, half a day, etc.) of the final estimate and actual effort is reported in all cases.

Further, the Magnitude of Relative Error (MRE) is shown for both the crowd and expert estimates, relative to the actual effort. The MRE for expert estimates was calculated directly from the effort estimates reported in the respective project’s issue tracker. For crowd workers, the categorical estimates from individual estimates were translated back to person-hours, from which a mean estimate was calculated. The next two sections review these results with respect to the original research questions.

Trial	Estimates				Crowd Agreement		Actual Effort Category	Expert Estimate		Crowd Estimate	
	NoR	All	AC	RE		(Fleiss' Kappa %)		Category	MRE	Category	MRE
1	1	6	5	1	APrA	76.19%	A day	Half a week	100%	<b>Half a day</b>	50%
2	2	20	10	10	APrA	76.19%	One Hour	<b>A day</b>	500%	<b>A day</b>	700%
3	3	26	15	11	APrA	76.19%	One Hour	<u>One Hour</u>	0%	A day	700%
4	2	12	10	2	APrA	80.95%	Half a day	<u>Half a day</u>	0%	A day	100%
5	1	10	5	5	APrA	79.17%	One Hour	Half a day	100%	<u>One Hour</u>	0%
6	2	17	10	7	APrA	76.19%	One Hour	<u>One Hour</u>	0%	Half a day	300%
7	1	7	5	2	APrA	79.17%	Half a week	<u>Half a week</u>	20%	Half a day	80%
8	1	7	5	2	APrA	79.17%	One Hour	A day	700%	<b>Half a day</b>	300%
9	3	30	15	15	SuA	66.67%	Half a day	<u>Half a day</u>	25%	<b>Half a day</b>	0%
10	1	7	5	2	APrA	76.19%	Half a day	<b>A day</b>	50%	<b>A day</b>	100%
11	3	26	15	11	SuA	66.67%	One Hour	Half a week	2300%	<b>Half a day</b>	300%
12	1	8	5	3	APrA	79.17%	One Hour	<u>One Hour</u>	0%	Half a day	300%
13	2	16	10	6	APrA	83.33%	Half a week	<u>Half a week</u>	20%	A day	60%
14	2	18	10	8	APrA	79.17%	Half a day	<u>Half a day</u>	25%	A day	100%
15	2	16	10	6	APrA	83.33%	Half a week	<u>Half a week</u>	20%	Half a day	80%
16	2	22	10	12	APrA	79.17%	One Hour	Half a week	2300%	<u>One Hour</u>	0%
17	2	24	10	14	APrA	76.19%	Half a day	One week	1100%	<b>Half a week</b>	400%
18	2	14	10	4	APrA	83.33%	Half a day	Two weeks	1700%	<b>A day</b>	100%
19	2	17	10	7	APrA	75.0%	Half a day	Half a week	300%	<b>A day</b>	100%
20	3	21	15	6	APrA	79.17%	One Hour	<b>Half a day</b>	300%	Half a week	1900%
21	3	19	15	4	SuA	70.83%	Half a day	<u>Half a day</u>	0%	<b>Half a day</b>	0%
22	3	24	15	9	APrA	76.19%	One Hour	Half a day	200%	<u>One Hour</u>	0%
23	1	9	5	4	APrA	83.33%	Half a day	A day	100%	<u>Half a day</u>	0%
24	1	5	5	0	APrA	79.17%	A day	<b>Half a week</b>	200%	<b>Half a week</b>	150%
25	3	26	15	11	SuA	71.43%	One Hour	<u>One Hour</u>	0%	Half a day	300%
26	3	17	15	2	APrA	80.95%	A day	<b>Half a week</b>	100%	<b>Half a week</b>	150%
27	3	23	15	8	SuA	66.67%	Half a day	<u>Half a day</u>	0%	<b>Half a day</b>	0%
28	3	22	15	7	SuA	52.38%	A day	<b>Half a week</b>	100%	<b>Half a week</b>	150%
29	3	27	15	12	SuA	66.67%	One Hour	<u>One Hour</u>	0%	A day	700%
30	1	10	5	5	APrA	83.33%	Half a day	<u>Half a day</u>	0%	One Hour	75%
Total	<b>62</b>	<b>506</b>	<b>310</b>	<b>196</b>							

Table 8.2: Summary of trial results, including number of estimates received, accepted, and rejected, outcome for each round and overall trial, and level of agreement achieved within the crowd. The abbreviations APrA and SuA in the agreement column stand for Almost Perfect Agreement and Substantial Agreement, according to Munoz and Bangdiwala [247]. NoR stands for Number of Round, AC for Accepted, and RE for Rejected.

### 8.3.1 Crowd Performance Compared with Experts

The CRQ1 research question concerns the reliability of the crowd estimate compared with the expert estimate and actual effort. The results of the 30 trials conducted are reported in Table 8.2. The table reports the total number of rounds for each trial, along with the number of accepted and rejected submissions. The table also shows the category of actual effort required for the task concerned, the expert's estimate, and the estimate produced by the crowd. Estimates in bold are the estimates closest to the reported effort (either expert or crowd, or both if the error was equal). Estimates are also underlined if the correct category

was estimated.

As can be seen from the table, the crowd workers correctly predicted the effort category for 7 of the 30 trials (5, 9, 16, 21, 22, 23, 27), as compared with 13 trials by the expert estimator (3, 6, 7, 9, 12, 13, 14, 15, 21, 25, 27, 29, 30). This indicates that expert estimators significantly outperform crowds when considering only correct predictions.

However, when considering reliability more broadly, it can be seen that the crowd workers produced the same estimates as experts in 8 trials, crowd workers were more accurate in 10 trials, and experts more accurate in 12 trials. In addition, comparing the Mean MRE of crowd estimates (239.83%) with the Mean MRE of expert estimates across all the issues indicates that the crowd workers' error is less than the experts' by 102.19%. This suggests that crowd workers are more likely to *underestimate by a category*, while experts are more likely to *overestimate using person-hours*.

This comparison was checked by investigating whether a statistically significant difference existed between the distributions of the MREs for the crowd and expert produced estimates. First, the Shapiro–Wilk test was applied to both MRE distributions to determine if either were normal. The results of the test for crowds ( $W=0.60984$ ,  $p=9.645e-08$ ) and experts ( $W=0.57812$ ,  $p=3.987e-08$ ) indicate that both were non-normal. Therefore, the Mann–Whitney U Test was applied, since both distributions were assumed to be independent. Applying this test to the two distributions resulted in a rejection of the null hypothesis ( $W=497$ ,  $p=0.4861$ ), indicating that there is no statistically significant difference between the MRE distributions, and thus, that the two effort estimation techniques have similar accuracy.

### 8.3.2 CPP Scalability

The CRQ2 research question addresses the scalability of Crowd Planning Poker, compared with a Planning Poker estimation activity conducted by a team of experts. Table 8.3 summarises the costs and effort associated with the trial.

The table shows that the total amount of time that crowd workers took to produce an estimate through CPP ranged from 17 to 76 minutes, including idle time. Unsurprisingly, the number of rounds in a trial had a significant influence on the time taken, with Trial 8 requiring just a single round and lasting just four minutes, for example. These results suggest that producing an estimate from a crowd takes some additional time compared with a Planning Poker process conducted by a group of experts, as described in Chapter 3.

Expert estimation may also be considerably faster when the expert group already has a good understanding of the task to be estimated and can rapidly achieve consensus without the need for discussion. Nevertheless, the results demonstrate that crowds can produce estimates relatively quickly and on demand. In addition, the work demonstrates that CPP can estimate



Trial	Sign-ups	Estimates		Minutes	Cost
		Received	Paid		
<b>10</b>	105	6	6	35	\$0.9
<b>11</b>	253	20	15	51	\$2.25
<b>12</b>	366	26	20	70	\$3.0
<b>13</b>	230	12	11	45	\$1.65
<b>14</b>	118	10	7	32	\$1.05
<b>15</b>	302	17	13	62	\$1.95
<b>16</b>	163	7	7	21	\$1.05
<b>17</b>	202	7	5	17	\$0.75
<b>18</b>	643	30	23	74	\$3.45
<b>19</b>	88	7	5	28	\$0.75
<b>20</b>	474	26	17	59	\$2.55
<b>21</b>	131	8	7	33	\$1.05
<b>22</b>	348	16	12	54	\$1.8
<b>23</b>	476	18	16	63	\$2.4
<b>24</b>	285	16	10	54	\$1.5
<b>25</b>	393	22	15	71	\$2.25
<b>26</b>	323	24	16	68	\$2.4
<b>27</b>	307	14	12	48	\$1.8
<b>28</b>	342	17	13	76	\$1.95
<b>29</b>	425	21	16	71	\$2.4
<b>30</b>	553	19	16	67	\$2.4
<b>31</b>	324	24	20	71	\$3.0
<b>32</b>	215	9	7	28	\$1.05
<b>33</b>	52	5	5	35	\$0.75
<b>34</b>	544	26	18	61	\$2.7
<b>35</b>	258	17	16	66	\$2.4
<b>36</b>	449	23	22	60	\$3.3
<b>37</b>	303	22	18	57	\$2.7
<b>38</b>	408	27	20	57	\$3.0
<b>39</b>	241	10	9	19	\$1.35
<hr/>					
<b>Total</b>	9321	506	397	1553(26Hrs)	\$59.55
<b>Mean</b>	311	17	13	52	\$1.99

Table 8.3: Breakdown of trial costs.

multiple tasks in parallel, as compared with in-person Planning Poker, where only one issue can be considered at a time.

Table 8.3 also reports the cost for conducting the trials, showing an average cost of \$1.99 to produce a final estimate (again, this figure is influenced by the number of rounds taken in a trial). This cost would appear to compare very favourably with the cost of running a Planning Poker session with a software team. Assuming that a team of five developers with an average hourly salary of \$40 (excluding other costs) can estimate 10 tasks in hour, the average cost per estimate would be \$20. Thus, the results of the trials demonstrate the potential for a significant cost saving.

## 8.4 Discussion

Beyond estimates, an additional benefit of requesting a rationale from crowd workers when they supply their estimate is that further insight and analysis of the task to be estimated can be obtained. This phenomenon was first noted in the pilot studies in Chapter 6. Many of the workers provided useful information about how to approach the task. Such advice and guidance was often very detailed, for example, on a task concerning the creation of a preview mode for sites using the Apache Maven site plug-in (MSITE-68), a crowd worker wrote:

“This seems like a good case for building at the DOM level, to ‘implement’ the changes in parallel for the previews. If that is in fact the case, it would probably take about a day to get a working prototype. If not... then a day would also probably be enough to know that this simply cannot be done.”

The crowd worker provides a suggestion that the resolution of the issue can be done by monitoring a page’s DOM for changes to create a preview. They also include a suggestion that a prototype should be created first to determine whether the feature is feasible.

For another issue, concerning the implementation of a new indexing mechanism for a JBoss workspace, the crowd worker provides a detailed breakdown of the work to be done:

- “1. How to determine, and what is the most efficient and accurate query for nodes and necessary information?
2. Initial testing for viability of indexing nodes (no lost data, consistency, etc.)
3. Deeper testing including stress testing at higher node counts, ensure all threads are deleted, etc.”

In particular, the crowd worker emphasises the importance of different types of testing, noting that non-functional testing should be treated separately from the design and functional testing of the feature.

These examples were intriguing, as the researcher had not anticipated that crowd workers would provide insights with significant *domain specific* knowledge. These suggestions and explanations have the potential to be of significant assistance to a team during the wider triage process for a software task that occurs alongside estimation. Overall, the results also demonstrate that the CPP process can effectively discriminate between tasks of different orders of magnitude, ranging from half a day through to two weeks.

### 8.4.1 Threats to Validity – Issue Availability

A limitation of the study was employing issues created for open-source projects. This decision was necessary as the experiment required a source of software tasks that could be provided to anonymous crowd workers and that had been annotated with expert-estimated and actual work costs. This meant there was a risk that the crowd workers could access the issue trackers themselves and simply supply the actual reported cost, creating a threat to the validity of the reliability results.

This risk was mitigated in several ways. First, the issue identifiers were not supplied to the crowd workers, and issues were selected from issue trackers that required user registration. This created an additional step to deter workers. Second, workers were asked for a categorical submission, rather than an absolute person-hour value, creating an additional step if the source issue was accessed. Finally, workers were encouraged to supply their own estimate and it was clear that payment was not contingent on supplying the correct result. Consequently, there is no evidence in the behaviour logs that the workers accessed project issue trackers, although this may have occurred outside the CPP user interface.

Since the software development issues were selected from open-source projects, the expert effort estimations may have been changed in a later stage and the actual reported effort may not reflect what a task actually took. As explained above, data confidentiality limited source options to open-source projects. In addition, the variety of software development projects and the abundance of information may not be easily collected from different sources such as commercial software development houses.

The risk of tampering with the expert estimate at a later stage was mitigated by reviewing the change log of the issues. None of the issues' estimates were found to be changed after its initial entry. In addition, the issue change logs were reviewed with respect to the reporting of actual effort. The issues' change logs show that the actual efforts were updated at the same

time as a major change of the issue properties, e.g. issue status. This is an indication that the actual efforts were kept by the issue assignee.

Further, crowd workers were asked to self-assess their experience in software development. Not all workers would necessarily have such experience, and thus there was a risk of having workers without software experience. Another option to assess the workers' experience is to ask them to take an exam. While this is a viable option, it increases the burden on the workers and thus may result in extra money incentives being required. Similarly, for extra fees, AMT offers the option to hire workers who have undertaken an assessment of their software development experience by AMT. Both options increase the overall cost of CPP, which limits its scalability.

The risk of having workers with no software development experience is mitigated by asking the worker to explain their experience and quantify it in a declaration form as part of the HIT instruction, see Figure 8.3. In addition, workers with no experience in software development may struggle to provide a quality justification for their estimates. That was clear during the soft-reject process, where the researcher reviewed the auto-rejected assignments and returned to the experience declaration to find that most workers with no or unclear experience declarations had submitted low-quality justifications and thus they were rejected.

## 8.5 Summary

While the literature suggests that most of the SEE academic research is in the area of ML SEE methods, it also illustrates how the industry is relying on expert-based SEE methods such as Planning Poker. Narrowing the literature gap by investigating and developing an expert-based SEE is one of this chapter's outcomes. It adds to the earlier development of CPP by automating the process using a human computation orchestration methodology.

Human computation can boost machine artificial intelligence computation by delegating part of its computation to a human to process. Such a part might be impossible for a machine to compute, or it might produce unreliable outcomes without human intervention. For CPP, that part comprises the individual estimates that work as seeds which are grown by the machine in order to produce a final estimate. The machine uses CPP, a process inspired by Planning Poker, to collect individual estimates from crowd workers and then aggregate them to produce the final estimate.

In order to examine the process efficiency and practicality, an experiment was designed to estimate prior software development issues that are annotated with actual and expert-estimated efforts. Since the process is automated, the budget is the only limit on the number of issues to be estimated. Thirty issues were randomly selected from a filtered list of JOSSE dataset issues.

Then, an experiment was conducted to estimate those issues. The aim of the experiment was to see whether the automated CPP process is able to produce estimates that are comparable with expert-estimated efforts. The results indicate that CPP successfully produced estimates that are comparable with expert estimates. These results therefore present several opportunities for future research directions. First, an observed benefit of CPP compared with in-person Planning Poker is the ability to obtain results on demand, rather than needing to wait for a team's regular planning session. In addition, it was noted that the crowd workers often provided useful insights as to the best approach to take to resolve the issue and the sub-tasks that this might involve. Therefore, CPP could be used by a software team to obtain an initial estimate for a task along with some initial guidance, prior to the task being triaged by a team member.

While crowd workers were able to produce expert-comparable estimates using CPP, more details are needed about how the workers were able to do that and which part of CPP participated in enabling the worker to produce such reliable estimates. Therefore, the next chapter will investigate the workers' behaviour and draw insights into how the workers interact with the CPP process.

## Chapter 9

# Crowd Estimator Personas: an Ethnographic Study of Crowd Behaviour

The preceding chapters of the thesis have demonstrated the feasibility and reliability of Crowd Planning Poker (CPP). However, fine-grained details about how crowd workers produce software effort estimates and which circumstances help crowd workers to produce such estimates are not addressed. Such details are essential to understanding the mechanics and dynamics of CPP and thus continuing to improve CPP and software effort estimation in general. They will also help future studies in assigning different kinds of work for suitable workers, providing rationales for challenges, and illustrating a framework for similar future investigations.

This chapter investigates and describes different behavioural profiles of the crowd workers who participated in the CPP experiments described in earlier chapters. It extracts fine-grained details of the estimation process followed by workers and uses these details to develop a set of worker personas. The chapter provides insights into the reasons for the differing quality of crowd worker assignments and their use of inputs from prior rounds. It also pinpoints which resource was most helpful for the workers while estimating the software development tasks. The chapter contributes by developing crowd estimator *personas* and persona descriptors using systematic log scanning, and then provides a qualitative analysis of CPP components, specifically, peer discussion and the crowd feedback loop. It also contributes by confirming the outcomes of the systematic scanning and qualitative analysis by asking participating crowd workers about the outcomes.

A large amount of the worker behaviour (User Interface (UI) interaction logs) dataset was collected while conducting the CPP experiments; this was possible because an interface logger was enabled. The logger captured all interaction between crowd workers and the CPP

software UI. Just from the last experiment, which is detailed in Chapter 8, over five million UI interaction events were recorded for 1,449 crowd workers. Analysing such a wealth of information about participants helps draw a better understanding of how and why CPP has worked.

The next section reviews ethnography studies in software engineering and explains how ethnographic studies have advanced to enable investigation of online communities such as crowdsourcing platforms. Then, Section 9.2 explains the research method of systematic behaviour scanning and the process of developing crowd personas. After that, Section 9.3 presents a systematised analysis of the UI logs to develop the personas. It provides details about accurate and inaccurate estimates across different quality classes of crowd assignments, crowd personas, and crowd workers' perception of peer discussion and the feedback loop. Then, Section 9.4 explains the crowd survey design and presents the survey results. Section 9.5 discusses the analysis of the customised scanning of UI logs in the light of the survey results. Finally, the last section provides a summary of this chapter followed by insights into future research directions.

## 9.1 Related Work

This section reviews the literature at the intersection of crowd worker behaviour, assignment quality, and UI data-based ethnography. In particular, the review identifies aspects of previous work that informs the design of the present study.

Ethnography is a research discipline that traces and studies human behaviour. Ethnography as a research method has been used in several software engineering areas, including software development [248], maintenance [249], architecture [250], and software testing [251, 252]. In a typical ethnographic study, a researcher immerses themselves physically in the targeted study. The researcher spends prolonged time observing and recording different behavioural aspects of the group members. Spending a prolonged period of time is a key challenge in applying ethnography to software practices, as explained in Passos et al. [253]'s work. However, in rapidly changing disciplines such as software engineering, short ethnographic studies are also possible, as done by Sharp and Robinson [248] while studying XP practices. Sharp and Robinson [248] spent a week with the XP team, tracing and logging their behaviour against XP practices. Furthermore, attending and observing the study group in person is a challenge that can be resolved by inferring human behaviour from software UI interaction logs. For instance, Kim et al. [47]'s ethnographic study employed UI interaction logs to understand the behaviour of copy-and-paste programming.

Ethnographic studies investigate real-world practices, which may deviate from the theoretical or descriptive concept. For instance, Martin et al. [252] examined software testing and found

that testing has social aspects as well as technical ones. Thus, involving social aspects as well as technical advancements will help improve software testing. Two more comprehensive studies of the application of ethnographic methods in software engineering research were presented by Sharp et al. [254] and Rönkkö [255]; the reader is directed to these works for more information.

Applying ethnography to software engineering practices using traditional ethnography techniques has several challenges, such as the need to have a successful relationship between the researcher and research participants, and the impact of the researcher's presence on the reality of software development practice, among other challenges as pointed out by Passos et al. [253]. For instance, the attendance of the researcher at the working environment and the fact that the researcher is observing the development team may disrupt the team's behaviour and thus compromise the distinctive feature of ethnographic study, which is the recording of real-life behaviour.

Another aspect that calls for innovation of ethnographic methods is the emergence of digital and virtual life [46]. These contexts can be as important for software engineering practice as the physical world. Therefore, a sub-discipline called Digital Ethnography has emerged [46], which concerns topics related to digital innovations and their impact on humans. It also provides tools and methods for gathering data with the features of the digital world. For example, attending in-person might not be viable in such a world. Alternatively, video recording and digital logs can be used to infer human behaviour. Pink et al. [46] detail the emergence of Digital Ethnography and suggest several practices to conduct such research.

There have been three ethnographic studies of crowdsourcing reported in the literature [256, 257, 258]. O'Neill and Martin used ethnography research methods to highlight the challenge of unbalanced treatment of crowd workers and their employers. Furthermore, the ethnographic study illustrates the complexities of the relationships between workers and employers and states design aspects that can be implemented in crowdsourcing platforms to help in balancing the treatment of both parties.

Relevant to the present study, Gupta [258]'s thesis also uses ethnographic research methods. It identified different motives of the workers, features of the crowdsourcing platform, i.e. Amazon Mechanical Turk, the social side of crowd work, effort that goes unseen/unpaid, and workers' risk plans. Gupta [258] concluded that money is not the only motive of crowd workers, and they can take on collaborative work which may result in more unpaid work that is ignored by the crowdsourcing platforms. As a recommendation, Gupta [258] emphasises the importance of communication between crowd workers to enable more collaborative work between them and to help reduce the unseen effort of workers who do such jobs.

Moreover, Gupta et al. [259] describe how ethnography can be applied to crowdsourcing research. Their paper explained, using evidential examples, how crowd workers in India



work, their perspective of being crowd workers, and who they are. Gupta et al. [259] aim to influence the design of such platforms for workers in terms of how it impacts their life. Another example that illustrates how ethnographic study can differentiate between theoretical conception and real-world practice is Gray et al. [260]’s work. The study demonstrated that crowd workers can handle independent work and they can also work collaboratively. Using ethnographic methods, Gray et al. [260] examined that assumption and found that crowd workers are depending on each other and collaborating to enhance their crowdworking environments.

The concept of *personas* has been used in a variety of different areas to provide archetypes of subsets of a system’s users. These personas can then be used to understand how and why a system is interacted with in different ways. Personas have been used extensively in software requirements [261] as well as other areas of software engineering practice, [262, 263, 264]. Ford et al. [262]’s work, for instance, explains how a persona was successfully used to precisely allocate human resources to the right job. Their work tries to fix the misallocation of general software engineering skills by specifying contextual skills for specific jobs. Moreover, personas are also used in requirements engineering in several ways. One example is to represent archetypical users of a software system and limit software developers from stretching the assumptions of what makes users happy. Faily and Lyle [263] suggest guidelines to help keep these personas prominent in software engineering activities and integrate them into different software development tools. Aoyama [264] detailed more about different usage of personas in software development. In addition, they explained several associated challenges, such as a lack of knowledge and resources in creating and adopting them.

From the crowdsourcing perspective, a few studies have tried to implement the concept of personas for crowd workers. For example, Bernstein et al. [265] suggested two personas: “Lazy Turker” and “Eager Beaver” based on the workers’ contributions. By developing these two personas, Bernstein et al. [265] were able to clearly communicate the effort spectrum of crowd workers, define effort patterns and then resolve effort issues that were identified as part of the persona’s behaviour. More recently, Ayalon and Toch [266] used personas to examine their effects in inspiring empathy in crowd workers while criticising privacy designs. As a result, Ayalon and Toch [266] demonstrated how personas encourage workers to give useful critiques and how that enhances privacy-by-design processes. Moreover, Stergiadis [267] modelled crowd workers using personas and the repertory grid technique (RGT). This thesis found that RGT can offer empirically grounded data which can be helpful in different applications including user profiling and information graphics.

User interface (UI) log data have been used in several user experience studies and crowdsourcing research. In particular, UI logs have been used to model workers’ behaviour and therefore measure assignment quality [229, 268]. Instead of assessing the quality of workers’ assignments, Rzeszotarski and Kittur [229] focused on how workers produce the assignment,

and used the captured UI interaction logs to infer the assignment's quality and the likelihood of the worker's disengagement. They found that the UI logs can be used to build predictive models of task performance. Similarly, Kazai and Zitouni [268] adopted the same technique and enhanced the model's productivity by including trusted users' behaviour as a gold standard; they called them Gold Judges. The model's accuracy almost doubled using the Gold Judges' behaviour data.

Using UI interaction logs as a worker behaviour proxy has been done in a couple of ethnographic studies such as Kim et al. [47]'s work of investigating the copy-and-paste programming practice. With the emergence of digital ethnography, there will be more reliance on such logs, as pointed out by Pink et al. [46]. The point is that some traditional practices of ethnography are not viable, such as attending the research group in person. For instance, crowd workers in the CPP research group came from several countries (over 20 countries based on their internet connection IP), and attending in person would raise a substantial financial burden.

Moreover, tracking user interaction with the interface is more subtle, and thus less disruptive to the nature of the worker than someone attending the worker's workplace. As pointed out by Passos et al. [253], attending in person is challenging because it changes the reality of the work that is being studied. However, such logs cannot convey other human aspects, such as feelings and body expressions, which may leave an opportunity for research assumptions that need to be addressed. One way to reduce that gap is by using surveys to confirm study findings and assumptions with the workers before rendering them as outcomes.

Drawing on the explored literature, this chapter uses the logs of the crowd workers' interaction with the CPP UI as a proxy for their behaviour and thus systematically scans and analyses the logs, creates crowd personas, and confirms the analysis findings with a survey of participating crowd workers. Therefore, this chapter uncovers details about how crowd workers, using CPP, are able to produce expert-comparable estimates and what resources are most useful for predicting such estimates.

## 9.2 Design of Systematic Behaviour Scanning Study

In this study, a systematised scanning of the UI interaction log and analysis of selected topics of the CPP UI logs are used to investigate the behaviour of participating workers and develop crowd personas. The UI interaction log has been used as a proxy for crowd behaviour. Later, crowd workers were invited to participate in a survey to confirm the study findings, as explained in the next section 9.4. Thus, in this study, the outcomes are better interpreted alongside the post-CPP survey results.

The aim is to investigate what activities of CPP a crowd worker, as an estimator, has followed, abandoned, repeated, or even invented, and why. Different worker personas can be identified by such behaviour. Therefore, the analysis of UI log data drawn from the CPP crowd workers' behaviour is used to answer the following research questions about crowd behaviour (RQB).

- RQB1: What workflows do crowd workers follow when performing effort estimation assignments?
- RQB2: What are the different personas of crowd estimators and their descriptors?
- RQB3: What information artefacts do different personas of crowd workers value as a basis for making effort estimation decisions?

The following subsection explains more details about the systematised scanning of the UI interaction log, along with the persona descriptors. Then, it illustrates the process that is used in the selective analysis.

### 9.2.1 UI Interaction Log Systematic Scanning

The UI interaction log of crowd workers will be investigated systematically as explained in this section. The goal of the systematic investigation is to consider all logs to understand how a crowd worker applies the CPP process in the real world. For comparison purposes, Figure 9.1 illustrates an ideal state machine for crowd assignments. A crowd assignment starts with an *Estimating* state, where a crowd worker comprehends the information about a given software development task, picks an estimate category, and writes a corresponding rationale to support their selection. Then the assignment moves to an *Evaluating* state, where the quality evaluation of the crowd assignment takes place. If the evaluation classifies the assignment as good quality, then the assignment moves to an *Approved* state, which takes it to the end. However, if the assignment quality is below the quality threshold, the assignment moves to a *Rejected* state. After communicating the feedback to the worker, if the worker decides to improve the assignment, it moves to a *Revising* state, which is similar to the *Estimating* state, but populated with the previous worker's inputs. Otherwise, if the worker withdraws from the process or the improvement loop, the assignment moves to a *Dropped* state.

Further, the *Estimating* and *Revising* states encapsulate the state machines of the workers as illustrated in Figure 9.2. The crowd worker state machine starts with a *Reading* state for the CPP instructions and terms and conditions. Then, the worker moves to a *Declaring* state, where the worker's experience is declared and described. After that, the worker moves

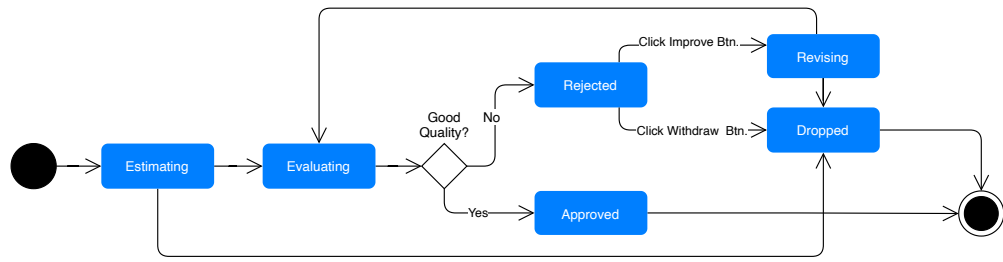


Figure 9.1: State machine for crowd assignments. Blue boxes represent the assignment state. The Estimating state is extended further in Figure 9.2.

to a *Reading* state again but with different content. The new content consists of several types of information about the software development task that needs to be estimated. If the worker chooses to search the web using Google, the worker moves into a *Searching* state. Then, the worker moves into an *Estimating* state, where the estimate category is selected. After that, the worker moves into a *Justifying* state, where the rationale behind the estimate category selection is explained. After the assignment quality evaluation, the worker moves into another *Reading* state, where the feedback of the quality assessment is communicated. If the worker decides to improve the assignment, then the worker stays in the *Reading* state but with different content, i.e. the information about the software development task. If the worker decides to withdraw, then the worker moves into a *Leaving* state where the worker is reminded to confirm the exiting choice. Otherwise, the worker terminates the state machine by submitting the assignment as-is.

During the CPP experiments detailed in Chapter 7 and Chapter 8, the CPP application recorded crowd workers' interaction with the application UI. It used JavaScript to collect the mouse and keyboard events and send them to the back-end server for each worker and assignment. Each record had information about the UI events, specifically, the event names as listed in Table 9.1, event content detailing associate event information such as click coordinates, event time, and event target, which represents the UI element as listed in Table 9.1, and reference pointers to the worker, assignment, and CPP round records. The UI log traces were captured for 1,449 crowd workers, resulting in over five million UI interaction events. Those records comprise the Crowd Planning Poker Behaviour (CPPB) dataset. The dataset is stored in an SQL database that is accessible from the a public repository<sup>1</sup>. Besides the UI events listed in Table 9.1, the CPPB dataset includes additional UI events such as scrolling and window resizing. While the additional events are not included in this analysis, they may benefit future research. The included UI events were selected since they can provide information about targeted crowd actions, as illustrated in Table 9.1.

Each log trace belongs to a single crowd assignment. Those traces are associated with the as-

<sup>1</sup><https://github.com/crowd-planning-poker>

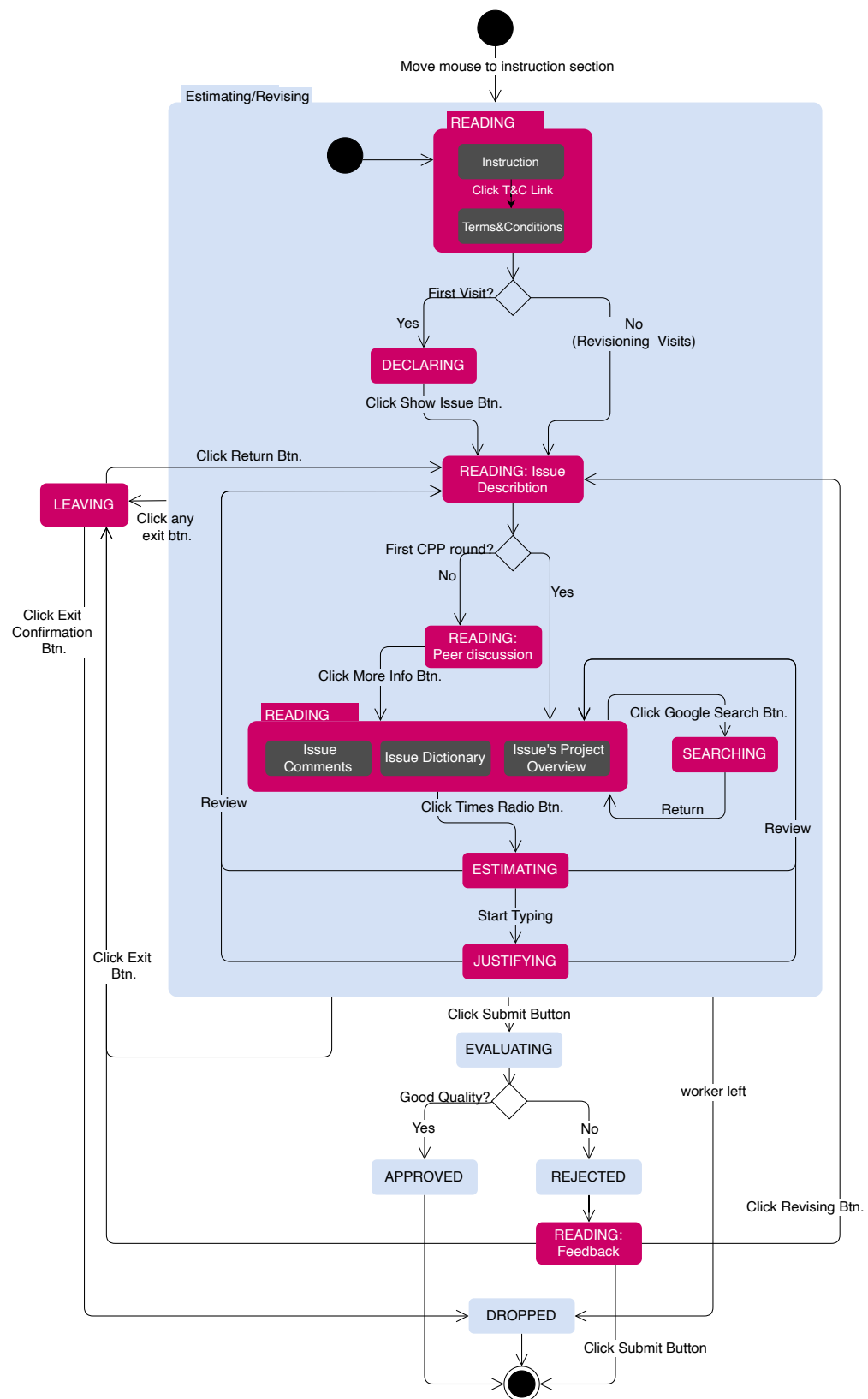


Figure 9.2: State machine of a crowd worker navigating through the CPP process. Grey boxes represent the crowd assignment states, as illustrated in Figure 9.1. Red boxes represent the crowd worker states that are encapsulated into the *Estimating* and *Revising* assignment states

Crowd Action/ UI Element	read	declare	search	estimate	justify	leave	idle
instruction	•						
terms-Conditions	•						
experience		•					
basic-info	•						
pair-discussion	•						
issue-dictionary	•						
issue-project	•						
issue-comments	•						
err-msg	•						
feedback	•						
google-search			•				
time-options				•			
estimating-step				•			
justification					•		
exit-dialog						•	
experience-exit						•	
task-desc-exit						•	
detail-info-exit						•	
estimate-exit						•	
feedback-exit						•	
waiting							
justify-submit					•		
feedback-submit							
irrelevant							•

Crowd Action/ UI Event	read	declare	search	estimate	justify	leave	idle
highlight	•	•	•		•		
copy	•	•	•		•		
paste		•	•		•		
mousemove	•			•		•	•
click	•	•	•	•	•	•	
type		•	•		•		

Table 9.1: The first table shows which crowd action is associated with each UI element. The second table shows which UI atomic events contribute to each crowd action.

Accuracy/ Quality	Excellent and Acceptable (A&B Classes)	Unacceptable (Class C)	Poor (Class D)
Accurate	222	196	
Inaccurate	593	308	669

Table 9.2: Number of assignments (cases) that are considered from each class. Each assignment has a stream of activities (Crowd Actions). Poor Class assignments are those assignments that are useless and show a clear sign of a quality problem, e.g. no justification. Thus, they are studied as a whole to identify the behaviour behind such assignments.

signment accuracy (crowd estimate compared with a ground truth), as well as the assignment quality, as described in Chapter 7. Therefore, the dataset of log traces from the UI is divided into five parts using two factors: the quality of workers' assignments and the accuracy of their estimates. Table 9.2 shows the log size of each portion. One reason behind selecting these two factors is the overall thesis objective of investigating the reliability of crowd estimates. Each portion is scanned to filter out duplication or events that are not considered in this investigation. The quality class is determined using the quality model explained in Chapter 7. Accuracy determination follows Moores and Edwards [109]'s suggestion. Assignments with an estimate that is within 20% of the actual effort are grouped in the accurate category. Otherwise, they are grouped in the inaccurate category.

Six UI events are associated with 23 CPP UI elements to represent eight crowd CPP actions. UI events are the worker's atomic actions which interact with the software UI, such as mouse click and highlighting. CPP UI elements are a group of HTML tags that build up an HTML page. Each HTML tag in the CPP HTML page is associated with an ID, and then these HTML tags are grouped to represent one CPP UI element. For example, "Instruction" is the CPP UI element that contains the CPP task instruction, and it consists of fourteen active HTML tags. Not every HTML tag in the CPP HTML page is considered, and thus, there are passive HTML tags that do not represent any value to the CPP process but are important to have on the page to implement the CPP process, for instance, HTML hidden input. By associating CPP UI elements with UI events, crowd CPP actions may be inferred. For instance, when a crowd worker *highlights* text from the "basic-info" CPP UI element, the worker has probably started a searching activity. Likewise, when the worker retrieves extra information about the issue by *clicking* on either the "issue-comments" or "issue-project" CPP UI element, the worker probably starts a reading activity. Table 9.1 shows the different kind of crowd CPP actions and CPP UI elements.

After processing the UI interaction log as described above, the logs are transformed into extensible event stream files to be analysed using DISCO and ProM, which are process mining applications. Section 9.3 gives more detail about the outcomes of the analysis and possible personas associated with such behaviour. Moreover, different persona descriptors are

extracted from the UI log and other experimental work components. These descriptors are explained in the following subsection.

### 9.2.2 Behaviour Descriptors for Crowd Personas

A persona is a way to express a group of merits that can be grouped into one character. Each persona has two kinds of *descriptor*: a public feature of that character, such as the character demographic, and a topic-related feature, such as spending more time reading an issue [261]. In this chapter, personas are used to describe different crowd estimator behaviours. In order to develop a persona, the features that a persona can assemble need to be identified and detailed. The goal of the descriptors is to represent the archetypical pattern that characterises the persona from different perspectives, including the persona's activity, intention, practice, effort, accuracy, knowledge, interaction, and reputation, as explained in Reinhardt et al. [269]'s research while describing knowledge workers. Adopting Reinhardt et al. [269]'s knowledge-worker actions, and based on what information is available in the CPPB dataset, eight descriptors were identified, specifically, *Activity*, *Advocacy*, *Repetition*, *Timings*, *Effort*, *Knowledge Seeking*, *Peer Interaction*, and *Assurance*. Additionally, two features were inherited from crowd assignments that are associated with workers' behaviour, specifically, estimate accuracy, measured by Mean Magnitude Relevant Error (MMRE), and crowd assignment quality class, as explained in Chapter 5 and Chapter 7, respectively. Table 9.3 states a statistical summary (mean) of the descriptors across the logs categories shown in Table 9.2. A crowd assignment has one or more UI interaction log records that reflect what the crowd worker has done before submitting an assignment. The eight descriptors are explained as follows.

#### *Activity.*

This descriptor illustrates how active the persona is in terms of the generated number of UI events as a result of the persona interacting with the CPP application UI. The persona's activity was measured by the average number of total event traces. Further, the average number of crowd actions (Table 9.1) is also used to measure the activity. Crowd actions are extracted from the logs, and sequence actions, e.g. typing text, are consolidated as one action.

#### *Advocacy.*

This descriptor shows how engaged the persona is in the CPP process. CPP-specific crowd actions are the actions that a crowd estimator takes to estimate a software development task. Each action is associated with relevant UI events, e.g. mouse clicks, and identified with targeted CPP-specific UI elements, e.g. the more info button. The persona's advocacy of the CPP process was measured by four measurements, specifically, the number of CPP crowd



	Number of Assignments	Activity		Number of Crowd Action	CPP Advocacy				Ratio of repeated Crowd Actions to Total Crowd Actions
		Number of UI Interaction Log Records	Number of UI Interaction Log Records		Number of Crowd CPP Actions	Ratio of Crowd CPP Actions to Total Crowd Actions	Duration of Crowd CPP Actions Seconds	Duration Ratio of Crowd CPP Actions to Total Spent Time	
A-B Accurate	222	1420	1420	19	15	0.77	138	0.7	0.43
A-B Inaccurate	593	1528	1528	20	16	0.76	152	0.71	0.44
C Accurate	196	1716	1716	27	21	0.77	135	0.66	0.47
C Inaccurate	308	1686	1686	29	22	0.77	152	0.68	0.5
D	669	1257	1257	24	19	0.76	97	0.58	0.43

	Average Time Spent on a Single Crowd Action	Crowd Action Timings		Time Spent on Peer Discussion	Assurance			Number of Knowledge Request	Assignment Size (Length of Total Submitted Text)
		Maximum Time Spent on a Single Crowd Action	Maximum Time Spent on a Single Crowd Action		Ratio of Submissions with Justification Change	Ratio of Submissions with Estimate Change	Ratio of Submissions with Estimate Change		
A-B Accurate	11	78	78	6	0.05	0.02	0.02	0.68	355
A-B Inaccurate	11	82	82	9	0.07	0.03	0.03	0.96	341
C Accurate	8	67	67	3	0.18	0.1	0.1	0.88	284
C Inaccurate	8	68	68	7	0.19	0.16	0.16	0.88	262
D	6.5	58.5	58.5	2.5	0.1	0.1	0.1	0.56	198.5

Table 9.3: A statistical summary (averages across logs per category) of the descriptors.

actions, the ratio of CPP actions to the rest crowd events, time spent while performing CPP actions, and the ratio of CPP action time to the total time spent.

#### *Repetition.*

This descriptor illustrates the repetitive patterns in a persona's behaviour, and how many times an action is performed during an estimation session. The repetition action descriptor is measured by the ratio of repeated crowd actions to total actions. The repetition pattern might offer insight on workers' familiarity with the estimation activity in general.

#### *Timings.*

This descriptor records how much time has been invested by a persona while estimating a software development task using CPP. Based on a single crowd action, effort is measured by two measurements: the average time and maximum time spent on a single action.

#### *Effort.*

This descriptor shows the persona's productivity, and how much effort has been invested to predict an estimate for a given software development task. As a proxy for the amount of thinking and analysing that a persona may do, the effort descriptor is measured by the length of the crowd justification and experience text, represented by the total number of words in the text corpus.

#### *Knowledge seeking.*

This descriptor gives an idea about how far the worker will go to obtain additional information in order to predict an estimate for the software development task in hand. It also illustrates the importance of the supplied additional information about the development task and whether the worker will be willing to go and spend more time on reading and comprehending the written materials. This descriptor is measured by the number of requests for additional information, including Google enquires.

#### *Peer interaction.*

This descriptor casts light on the CPP-specific feature of asynchronous communication of estimation rounds between crowd workers. As explained in Chapter 6 and Chapter 8, the workers' estimates were aggregated using crowd consensus on an estimation category for a given software development task. While it was not feasible to have synchronous communication between the workers, CPP added an additional area with summarised information about the previous estimation round for workers estimating the development task in later rounds. This descriptor is measured by time spent reading the summarised information from peers who estimated the previous round.

#### *Assurance*

This descriptor gives an idea about the willingness of a persona to take on additional work and invest additional time to improve an assignment with low quality. This takes place after communicating the quality assessment result with the worker. The assurance descriptor quantifies the additional effort using three measures: the number of submissions for each assignment (number of provisional assignments), the ratio of submissions with a change in estimate category to the total number of submissions, and the ratio of submissions with a change in estimate justification to the total number of submissions.

DISCO [270] was used to draw the event stream that represents the flow of crowd CPP actions. The process mining discipline uses its terminology to convey the meaning of the analysed process. In particular, “case” is used to refer to a single worker’s attempt at the process. It is equivalent to an assignment in the CPP context. “Activity” refers to a single activity (crowd CPP action) done by a “resource”. “Resource” means the subject who performs the activity, a crowd worker in the CPP case. “Path” is the flow between two activities. DISCO gives the user the ability to select which activities and activity flow paths need to be illustrated in the event stream diagram based on the frequency of those activities and flow paths. In this chapter, the activity frequency was set to show the 50% most frequent activities (crowd CPP actions), and only the most dominant process flows will be shown with 0% of path branches. The goal of these settings is to simplify the flow map.

## 9.3 Results of Behaviour Analysis and Crowd Personas

In this section, the systematised scanning of the CPPB dataset is reflected onto a descriptive observation of crowd behaviour, and based on that, four personas are developed. Because few assignments were classified as Class A (19 assignments), and classes A and B are both considered acceptable from a quality perspective, assignments in classes A and B were combined in this scanning. The following subsection will go through each category describing and comparing workers’ behaviour.

### 9.3.1 Combined Class A and B

Accepted assignments are classified as either A or B. These represent the best quality assignments that are collected by the CPP process depending on the automatic quality model described in Chapter 7.

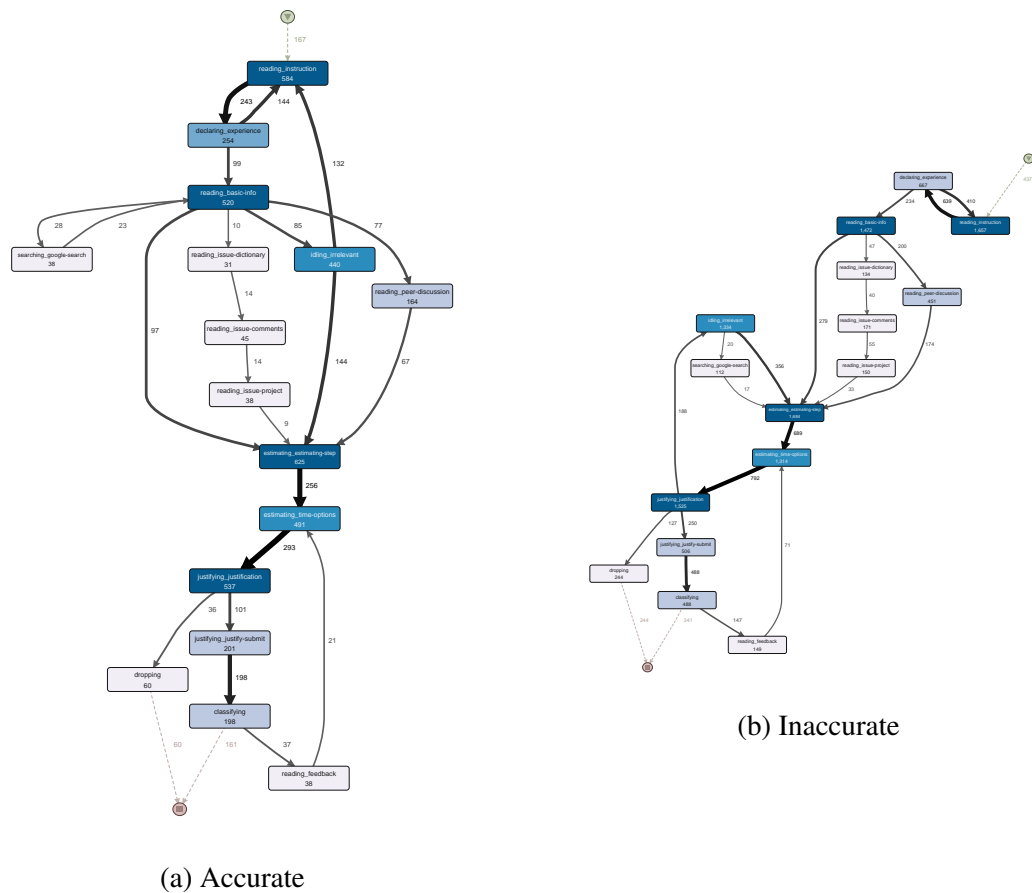


Figure 9.3: The activity stream maps for accurate and inaccurate assignments of classes A and B.

### Accurate Group Activities

Analysis of the Accurate group, that is, the group of workers whose assignments were accurate, shows that four frequent CPP Actions (CPPA) were followed as expected, including reading instructions, reading basic information, estimating, and writing justifications. Surprisingly, another popular Crowd Action (CA) was an idle event between reading basic information and estimating, although it was less popular than the four frequent CPPAs. Figure 9.3 shows the event stream map of these cases. Reading peer discussion and idling are the two activities performed most frequently after reading basic information. Mostly, workers preferred reading peer discussion to going over extra details of the issues, including issue dictionaries, comments, and issue project. However, issue comments were the most requested type of extra information for those who selected to go over the extra information. The majority of workers in this group considered revision after reading the classifier feedback.

### Accurate Group Stream Flow

The most dominant flows illustrated in Figure 9.3 are from reading instructions to declaring experience, from starting estimation to selecting an estimate, and from selecting an estimate to writing a justification. These flows are expected, and they followed the designed model. Unexpected behaviour is that several workers (16% of cases) dropped the CPP process after spending time (an average of 30 seconds) writing their justification and before submitting their assignment. The flow of crowd workers looks systematic and smooth, meaning that workers follow the process steps without jumping from one place to another. As shown in Figure 9.3, there are four flow loops in the flow for this group, as follows:

- Reading Instruction → Declaring Experience
- Reading Instruction → Declaring Experience → Reading Basic Information → Idling
- Reading Basic Information → Searching Google
- Selecting an Estimate → Writing a Justification → Submitting a Provisional Assignment → Reading Feedback

The looping behaviour illustrated by the four loops above indicates that the crowd worker is engaged and considering whether to go back and forth between CPP components, reflecting critical thinking behaviour while comprehending the given software development task. As a result, this group gave the best assignments in terms of accuracy and quality.

### Inaccurate Group Activities

The Inaccurate group had the same four frequent CPPAs as the Accurate group. Unlike the Accurate group, this group tended to idle while they were estimating. In addition, they searched Google while they were estimating. Figure 9.3 shows the event stream map of this group. Reading peer discussion was the activity workers did most frequently after reading basic information. Similarly, workers preferred peer discussion to going over extra issue information, and issue comments were the most requested of the three extra information components. About half of the workers who got the classifier feedback revised their assignments.

### Inaccurate Group Stream Flow

The most dominant flows illustrated in Figure 9.3 are the same as for the Accurate group. Similarly, about a quarter of the workers (21% of the cases) dropped the process some time

(about 34 seconds on average) after justifying their assignment and before submitting it. The flow of crowd workers is less systematic. It was systematic in the early stages of the process and only became chaotic after starting the estimation. Neither idling nor searching Google was expected during justification. As shown in Figure 9.3, there are three flow loops in the flow of this group:

- Reading Instruction → Declaring Experience
- Start Estimating → Selecting an Estimate → Writing a Justification → Idling
- Selecting an Estimate → Writing a Justification → Submitting a Provisional Assignment → Reading Feedback

The looping behaviour here is missing the loop for seeking more information about the issue. Unlike the Accurate group, this group rushes to estimation before taking their time to read and perhaps search the web for extra information. However, this group still shows good looping behaviour around estimation and justification, giving an indication that they are engaged while they think and write the rationale behind their estimate selection.

## Descriptors

By taking a look at the descriptor summary statistics relating to classes A and B, as shown in Table 9.3, it can be seen that Accurate and Inaccurate are alike except in the time that is spent estimating. The Inaccurate group is likely to spend more time on the overall estimation process, including reading peer discussion. In addition, the Inaccurate group tends to change their estimate selection during the revision of their assignments more than the Accurate group.

### 9.3.2 Class C

Class C assignments are not accepted because they are in a grey area between good quality and poor assignments. These represent assignments that show some confusion and irrelevancy to the topic of the issue. In addition, the assignments' rationales are not as complete as the ones in classes A and B.

#### Accurate Group Activities

Reading basic information is not among the frequent CPPAs of the Accurate group. The most frequent activities are reading instructions, selecting an estimate, and writing justifications.

Similarly to the Accurate group for classes A and B, this group idled between reading basic information and estimating. Figure 9.4 shows the event stream map for Class C. Workers in this group prefer neither reading peer discussion nor requesting extra details about the issue. Instead, the majority went from reading basic information to estimating. Only a few workers read comments about the issue. Some Google searches were done during the estimating part of the process. Few of the workers in this group considered revision after reading the classifier feedback; they either exited the process or insisted on submitting low quality work.

### Accurate Group Stream Flow

The most dominant flows illustrated in Figure 9.4 are from reading instructions to declaring experience, from starting estimation to selecting an estimate, from selecting an estimate to writing a justification, and from submitting a provisional assignment to reading the classifier feedback. The flow of crowd workers shows chaos from the beginning to the end. For instance, several workers started with reading issue comments and then exited the process. As shown in Figure 9.4, two flow loops were identified for this group:

- Reading Instruction → Declaring Experience
- Reading Instruction → Declaring Experience → Reading Basic Information → Idling

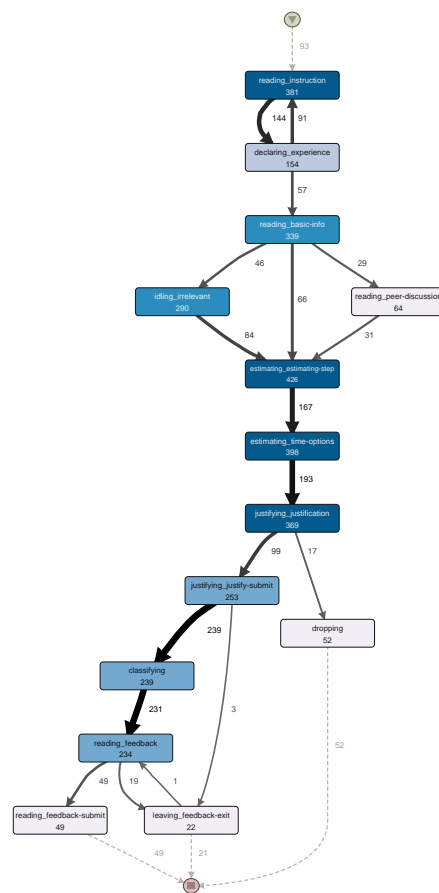
Unsurprisingly, the looping behaviour here is weaker than in assignments with classes A and B. The lack of looping behaviour suggests that less critical analysis was done by crowd workers in this group, and thus, they submitted assignments with lower quality.

### Inaccurate Group Activities

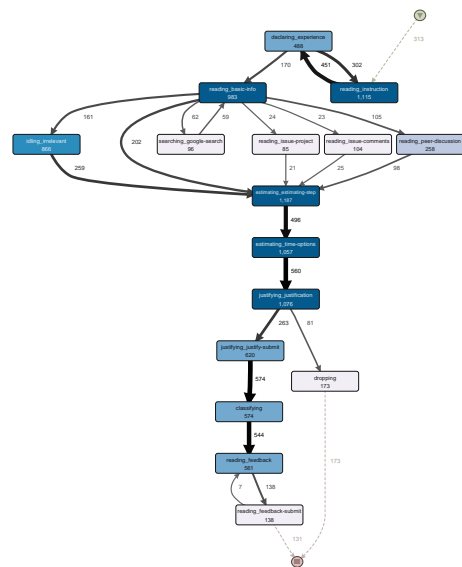
The Inaccurate group exhibits four frequent CPPAs: reading instructions, reading basic information, estimating, and writing a justification. In addition, workers search Google and idle after reading basic information (similar to the Accurate group of classes A and B). Figure 9.4 shows the event stream map of this group. Idling, reading peer discussion, and reading issue comments are the most frequent activities workers did after reading basic information. As was the case in the Accurate group, only a few workers from those who got the classifier feedback revised their assignments.

### Inaccurate Group Stream Flow

The most dominant flows illustrated in Figure 9.4 are the same as for the Accurate group. More workers (20% of the cases) have dropped the process after some time (about 19 seconds



(a) Accurate



(b) Inaccurate

Figure 9.4: The two activity stream maps for accurate and inaccurate assignments of Class C.



on average) justifying their assignments and before submitting them. The flow of crowd workers shows a lot of chaos, especially in the first half of the process (while the workers were reading the issue information). As shown in Figure 9.4, there are two flow loops for this group:

- Reading Instruction → Declaring Experience
- Idling → Estimating → Writing a Justification

This group has the least looping behaviour, with no looping on the basic information. Instead, workers tend to use Google to find ready justifications for their random selection of estimates. The groups that did not show iterative behaviour around the core information during the estimation process submitted inaccurate estimates. This gives insight into why machine-based SEE lacks reliability. As illustrated here, good estimators are taking in a lot of context and considering a lot of related information.

## Descriptors

Table 9.3 suggests the same pattern as between the Accurate and Inaccurate groups for classes A and B. However, the Inaccurate group tends to spend more time and changes their estimate selection during the revision of their assignments more than the Accurate group. Moreover, workers in both groups of Class C submitted smaller assignments than those in the classes A and B. Unsurprisingly, since workers are more rejected in this class, they spend more time and show more activities.

### 9.3.3 Class D

Poor assignments are classified as Class D. They have clear signs of poor work, e.g. submit no justification. This class includes unengaged and unqualified workers who submit useless assignments. The analysis of this group is to understand the behaviour of unwanted workers. Accuracy here is not a concern, and thus, the class is analysed as one group. In fact, an earlier attempt to distinguish the accurate and inaccurate assignments of this class failed, as both groups exhibit similar features.

## Activities

Reading instructions and estimating were the two most frequent CPPAs. Figure 9.5 shows the event stream map of Class D. The majority went from reading basic information to estimating. The second majority did the same but with an idling state between reading basic

information and estimating. Only a few workers read comments about the issue. Some Google searches were done during the first part of the process (the reading part). A couple of workers considered revising their assignments; most insisted on submitting low quality work.

### Stream Flow

The most dominant flows illustrated in Figure 9.5 are from reading instructions to declaring experience, from starting estimation to selecting an estimate, from selecting an estimate to writing a justification, and from submitting a provisional assignment to reading the classifier feedback. The flow of crowd workers shows chaos in the first part but is smooth in the second part. As shown in Figure 9.5, no significant loop has been found, except the usual one between reading instructions and declaring experience.

### Descriptors

Unsurprisingly, Table 9.3 suggests that this group made the lowest effort even though the auto classifier warned them, unlike the groups of Class C assignments, where workers showed extra activity and working time as a response to the warning. This group of Class D assignments showed the lowest level of activity across all the classes. Moreover, workers in this group submitted the smallest assignments, and they showed less demand for extra information.

## 9.3.4 Identified Personas

Four personas, John, Johanna, Sarah, and Smith, were derived from the analysis above and matched to the four quality classes A, B, C, and D, respectively. The personas' characteristics were assembled from the descriptors, behaviour activities, and behaviour loops detailed in previous subsections. The collective analysis description of the personas can be stated as the following:

- John: Goes systematically through the CPP process steps. He prefers to think and search while reading and before taking any decision. John prefers reading what people say about the issue. He is most likely to predict an accurate estimate. John will mostly take on extra work to improve his assignment quality.
- Johanna: Follows the process systematically until an action needs to be done, then she shows some chaos. She prefers to shift between thinking and searching at the point where she is stuck. While Johanna will probably submit a high-quality assignment, she is less accurate than John and tends to be more active than John.

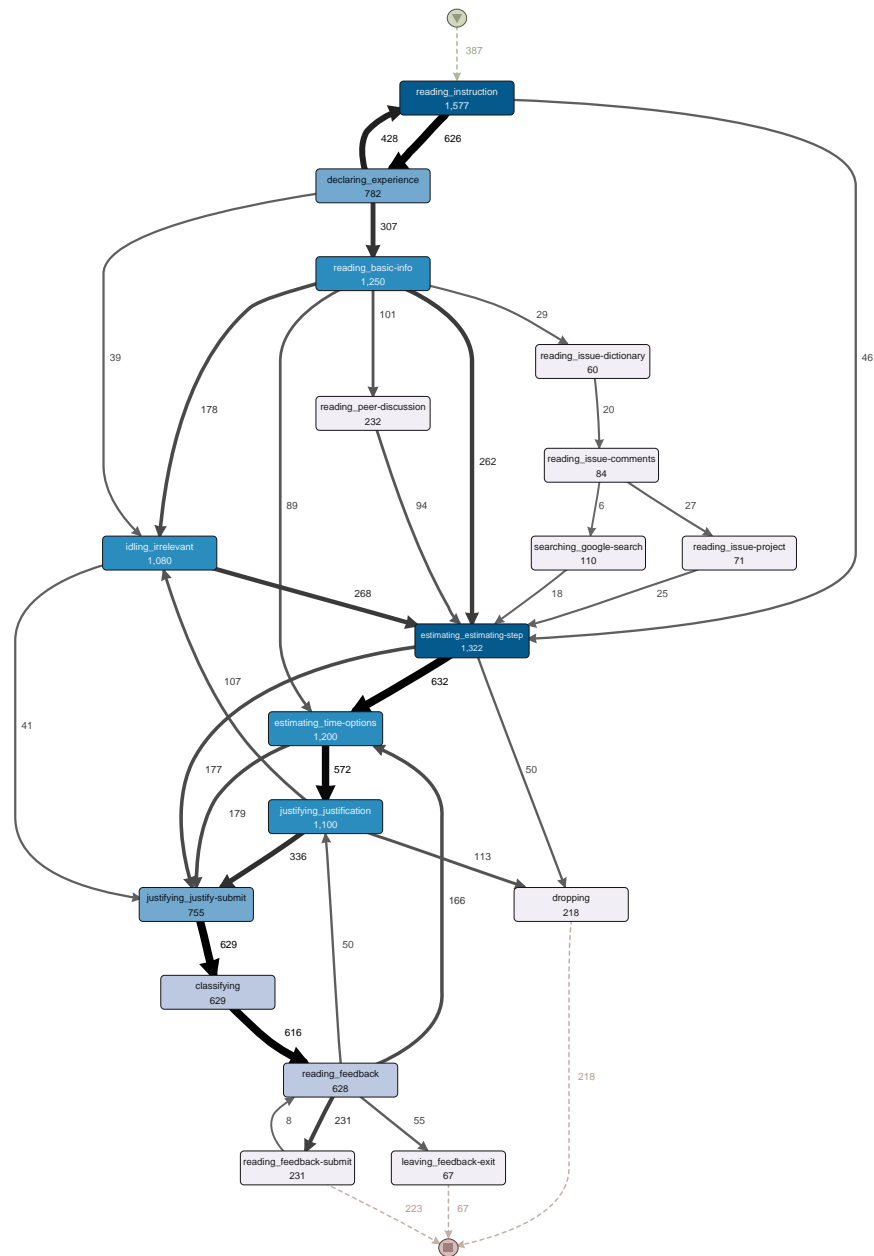


Figure 9.5: Activity stream map for poor assignments (Class D).

- Sarah: A good person who wants to deliver good quality. However, she is probably not qualified. She goes randomly between the process activities and shows signs of chaos and hyperactivity, e.g. a lot of clicks and several attempts to try most of the components of a website. Sarah will try to improve her work. However, she will give up quickly.
- Smith: Probably only after the money. He does not care about quality. Sometimes, he becomes an unengaged worker. Time is his concern, so he will probably allocate the least time to work on the task. Improving assignments is not his priority. Although his flow through the process seems systematic and smooth, he does not spend any useful time and clicks like a robot to obscure his behaviour. Smith will never submit a useful assignment.

Smith is the worker that you need to avoid. On the other hand, Sarah is a worker who might be worth taking on with extra training and explanation of the estimation task.

John is precise, and it is better to have him for the first round of CPP. Johanna can join in the next rounds of CPP to build upon John's estimates, with fewer interface options offered to her. She is easily distracted.

## 9.4 Post CPP Survey Study

Since crowd behaviour is instrumented using a proxy, the UI interaction log, the second part of this chapter is a survey study that investigates the crowd's opinion about the outcomes of the systematic scanning of their UI interaction log. The objective of the survey is to confirm the scanning outcomes regarding the crowd's familiarity with Planning Poker and software development issue, the most helpful part of CPP, and the crowd's perspective of the CPP quality model.

As discussed earlier in the persona descriptor sections, these three topics distinguish between accurate and inaccurate crowd workers who submitted good quality assignments. The following section will detail the survey design. The section after that will illustrate and discuss the survey results.

### 9.4.1 Survey Design

The targeted population of the survey is the 350 crowd workers who participated in a CPP trial and submitted an acceptable assignment for those trials explained in chapters 6, 7, and 8. Since the total population is relatively small (only 350) and consists of only one group

(accepted assignment), a quota sampling of 100 workers is selected. At the same time, the invitation is sent to the whole population. Such a sample will result in a confidence level of 95% with a 10% interval. The other group of crowd workers who submitted unacceptable assignments are excluded because they are not trusted to provide valid input in the first place.

The survey consists of nine questions that address three topics based on patterns discovered during systematic scanning of the UI interaction log: familiarity with Planning Poker, the most helpful part of the CPP process, and the crowd's perspective of the CPP quality model. Table 9.4 lists the questions along with their topics and the rationale behind each question.

Table 9.4: List of questions along with their topic and the rationale behind selecting them

Q#	Topic	Question	Rational
1	Planning Poker familiarity	How familiar you are with Planning Poker?	This is a direct question to assess Planning Poker familiarity and its role in impacting the worker behaviour and the accuracy of the submitted estimate.
2	Most helpful CPP component	<p>When you were estimating an issue, how useful did you find each of the following sources of information:</p> <ul style="list-style-type: none"> <li>• The specific issue description</li> <li>• General project information and context</li> <li>• Comments made by the issue's contributors</li> <li>• Peer estimates and justifications from previous Crowd Planning Poker rounds</li> </ul>	In this question, the worker specifies in general what source of information is most helpful.
3	Software issue familiarity	Thinking about the <b>LAST</b> issue you estimated, how familiar were you with the nature of the issue to be estimated?	This is a direct question to assess worker familiarity with the issue domain and its role in impacting worker behaviour and the accuracy of the submitted estimate.
4	Most helpful CPP component	<p>Thinking about the <b>LAST</b> issue you estimated, how much time did you allocate to each of these estimation activities?</p> <ul style="list-style-type: none"> <li>• Reading the present issue</li> <li>• Thinking about similar issues</li> <li>• Analysing and breaking down the issue sub-tasks</li> <li>• Searching the web for supplementary information</li> </ul>	This question investigates where workers are most likely to invest their time.

Table 9.4: continued from previous page

5	Software issue familiarity	<p>Thinking about an issue you were asked to estimate that did not seem familiar (the nature of the project or the work involved was unfamiliar), which of the following activities would you do?</p> <ul style="list-style-type: none"> <li>• Look at every bit of information on the page and probably click on every button to reveal as much information as possible</li> <li>• Pick any estimate and write any justification</li> <li>• Withdraw from the estimation page and exit</li> <li>• Ask a friend or colleague about it</li> <li>• Search for similar issues</li> </ul>	This is a follow-up question to the previous unfamiliarity question to understand the expected next action under uncertainty.
6	CPP quality model	When working on a Crowd Planning Poker issue, did you receive a warning about the quality of any of your estimates, saying that your assignment might be rejected?	This is a direct question to differentiate between workers who experienced a full cycle through the quality model from those who did not need to go through the quality enhancement cycle.
7	CPP quality model	What did you do after reading the warning feedback ?	This is a direct question to understand what is the next action taken by the worker in the case of submitting low quality work.

Table 9.4: continued from previous page

8	CPP quality model	<p>When I'm revising my estimate, I review the following sources of information:</p> <ul style="list-style-type: none"> <li>• The estimate that I have selected</li> <li>• The justification and comments that I have written</li> <li>• The specific issue description</li> <li>• General project information and context</li> <li>• Comments made by the issue's contributors</li> <li>• Peer estimates and justifications from previous Crowd Planning Poker rounds</li> </ul>	In this question, the worker details what source of information they are most likely to return to for clarifying and fixing the quality issue.
9	CPP quality model	<p>What do you think makes for a good justification for an estimate?</p> <ul style="list-style-type: none"> <li>• The justification is composed of descriptions of smaller tasks</li> <li>• The justification is composed of estimates for a series of smaller sub-tasks</li> <li>• A reference to a similar issue</li> <li>• A reference to a previous estimate for the same issue done in the previous round</li> </ul>	In this question, the worker gives their perspective and understanding of what makes a quality estimate.



Google Forms was used to create the nine survey questions. Some questions were supplemented with an additional free-text question in case the options listed as answers were insufficient or not valid for some workers. To have a look at the exact survey as implemented in Google Forms, refer to Appendix C, where all questions are listed as they were given to crowd workers.

Since the crowd workers had already participated in the CPP trials, AMT provided the option to communicate with them through their API. All the workers were invited to participate in the survey, and the survey was open until it reached the sample quota (100 participants).

Before running the survey, three researchers evaluated the survey to measure how much time it might take. The researchers took 3.5, 4, and 8 minutes to complete the mock survey. A decision was made to select the median (4 minutes) and pay workers \$7.50 per hour, since that was the minimum wage in the United States, where AMT is operating.

### 9.4.2 Survey Results

The survey results can be grouped into three aspects: the crowd's familiarity of CPP as a process and the software issues that were given to be estimated, the most useful components of the CPP process and most helpful issue data, and the crowd's perspective of the quality model and their response to the quality feedback loop.

The survey took seven days. On average, 35 answers were collected daily from the workers who participated in the CPP process. Questions 1, 3, and 5 address the crowd's familiarity with CPP process as well as the presented software development tasks they were estimating. Questions 2 and 4 address the most useful components of CPP from the workers' perspective. Questions 6, 7, 8 and 9 address the details around the CPP quality model and how the workers perceived the quality feedback and what reactions they had.

The survey consists of two familiarity questions as listed in Table 9.5. The first one assesses the crowd's familiarity with Planning Poker as an estimation method on a scale from 1 (not at all familiar) to 5 (very familiar). The second question measures the crowd's familiarity with the presented software issue for which the worker is about to estimate the effort.

The results suggest that most of the crowd (64%) is familiar with Planning Poker as a process. They also acknowledged that the issue is something part of their experience. However, the crowd is less knowledgeable about the issue than Planning Poker as a process. Thus, only 58% of workers indicated that they know the issue or its project. Please refer to Figure 9.6 for a histogram of both topics.

Moreover, when crowd workers are not familiar with the presented issue (see Table 9.6), the majority (65%) of the crowd start looking for additional information and clicking on every

#	Question	Familiarity Scale (1 is not familiar 5 is very familiar)				
		1	2	3	4	5
Q1*	How familiar you are with Planning Poker?	3%	15%	19%	<b>37%</b>	26%
Q2*	How familiar were you with the nature of the issue to be estimated ?	2%	13%	27%	<b>38%</b>	20%

Table 9.5: Survey results for crowd familiarity of Planning Poker and the software development task in percentages of participating workers (100 participants divided between the answer columns).

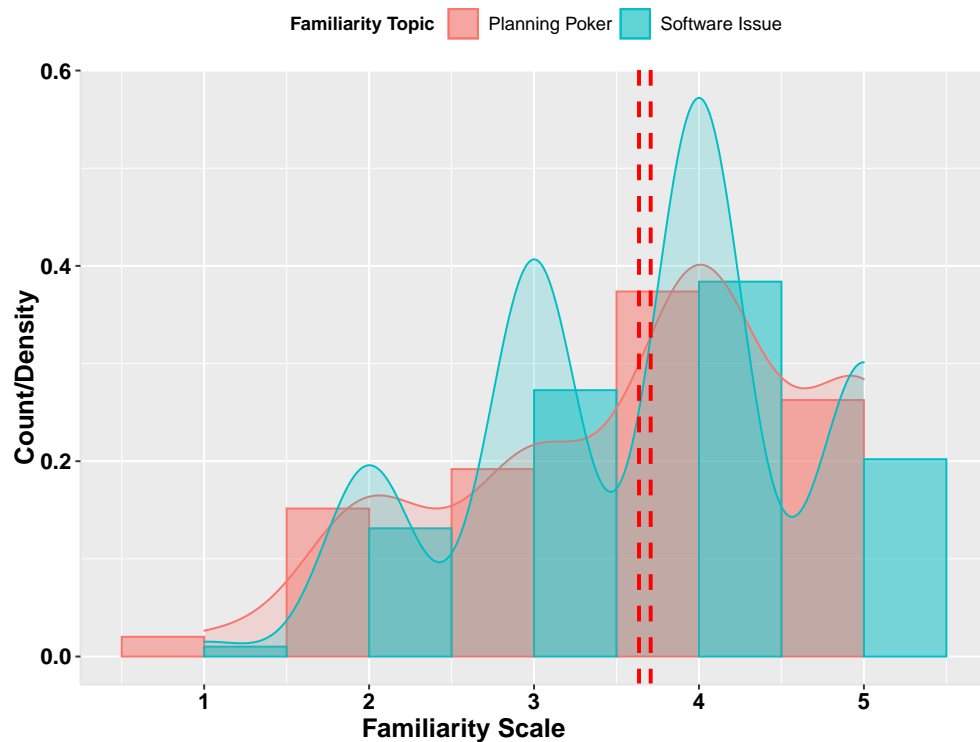


Figure 9.6: Histogram illustrating crowd familiarity with Planning Poker as a process and with the selected software issue on a scale from one to five, where one indicates “Not at all familiar” and five indicates “Very familiar”. The two red dashed lines indicate the mean.

Answer Options	How Likely Scale (1 is rare 5 is often)				
	1	2	3	4	5
Look at every information in the page & click every button. . .	4%	9%	24%	<b>36%</b>	24%
Pick any estimate and write any justification	30%	<b>21%</b>	18%	<b>21%</b>	9%
Withdraw from the estimation page and exit	<b>27%</b>	26%	18%	19%	9%
Ask a friend or a colleagues about it	17%	17%	18%	<b>25%</b>	21%
Search for similar issues	4%	8%	20%	35%	32%

Table 9.6: Q5: Thinking about an issue you were asked to estimate that did not seem familiar, which of the following activities would you do?

Answer Options	Usefulness Scale (1 is not useful 5 is very useful)				
	1	2	3	4	5
The specific issue description	4%	6%	27%	<b>32%</b>	31%
General project information and context	2%	12%	22%	<b>44%</b>	20%
Comments made by issues contributors	6%	13%	26%	<b>36%</b>	19%
Peer estimates and justifications from previous crowd planning poker rounds	5%	8%	28%	<b>45%</b>	14%

Table 9.7: Q3: When you were estimating an issue, how useful did you find each of the following sources of information?

button on the web page. They also search the web looking for similar issues. However, the workers rarely (28%) withdraw from the process or just pick any estimate.

Starting with which information the crowd found most useful among the offered information, the result generally suggests that the crowd found all the information very useful, see Table 9.7. The crowd scaled four types of information about the issue from 1 (not at all useful) to 5 (very useful). Most of the crowd (> 55%) indicated that all the information was useful. Besides task description (63%), project information (64%) and peer discussion (60%) were the most useful information.

From the CPP process activities perspective (Table 9.8), crowd workers suggested that thinking about similar issues or finding an analogy for the given issue was the estimation activity they spent the least time on. Only 19% of workers said they spent most of the time thinking about similar issues, and 42% of them spent little time on this activity. However, the majority of the workers (61%) suggested that they spent most of the time analysing (35%) and reading (26%) the presented issue. Searching the web for extra information was an activity that crowd workers spent some time on.

Answer Options	Little	Some	Most
Thinking about similar issues	<b>42%</b>	24%	19%
Searching the web for supplementary information	<b>25%</b>	18%	19%
Reading the present issue (issue description, project information and comments)	19%	17%	<b>26%</b>
Analysing and breaking down the issue sub tasks	13%	<b>40%</b>	35%

Table 9.8: Q4: How much time do you allocate to each of these estimation activities?

Answer Options	How Likely Scale (1 is rare 5 is often)				
	1	2	3	4	5
The justification is composed of descriptions of smaller tasks	2%	7%	33%	<b>36%</b>	20%
The justification is composed of estimates for a series of smaller sub-tasks	2%	10%	26%	<b>41%</b>	20%
A reference to a similar issue	4%	6%	28%	<b>39%</b>	21%
A reference to previous estimation on the same issue done in the previous round	4%	12%	32%	<b>34%</b>	15%

Table 9.9: Q9: What do you think makes for a good justification for an estimate?

When the crowd was asked what makes a high-quality assignment (Table 9.9), the majority (61%) responded that a justification explaining possible sub-tasks along with their sub-estimates can reflect a high-quality assignment. Similarly (60%), responded that reference to previous experience or a peer estimate also reflects a high-quality assignment.

Moreover, when the workers were asked whether they received a warning about their assignment quality (Q6 in the survey), 38% answered Yes, 43% answered No, and 18% answered Maybe. If the Maybe group (18%) is considered as noise and excluded, that gives us around 47% of crowd workers who submitted acceptable assignments after having received a quality warning. As illustrated in Table 9.10, the majority of them (61%) revised their assignments. Only 17% decided to withdraw from the process after the feedback.

In addition, when workers revised their assignments (Table 9.11), mostly (38%), they re-

Answer Options	Count(%)
Revise my justification	<b>27(32%)</b>
Withdraw from the feedback page and exit	9(11%)
Revise my estimate selection	24(29%)
Closed the webpage (exit the CPP task)	5(6%)
Submit it without revising it	14(17%)

Table 9.10: Q7: What did you do after reading the warning feedback?

Answer Options	Count(%)
The specific issue description	24(18%)
Peer estimates and justifications from previous crowd planning poker rounds	13(10%)
General project information and context	<b>25(19%)</b>
The justification and comments that I have written	<b>25(19%)</b>
The estimate that I have selected	<b>25(19%)</b>
Comments made by issues contributors	19(14%)

Table 9.11: Q8: When I'm revising my estimate, I review the following sources of information.

viewed their estimate selection and the justification that they had written. Some (19%) also retake a look at the issue description and project overview.

## 9.5 Discussion

The crowd survey results suggest similar conclusions to those are drawn from the behaviour analysis, starting with the chaos patterns that are spotted in the Johanna and Smith personas. As per the analysis above, the exploratory behaviour exhibited can result from a crowd worker not being familiar with the issue. The survey says that workers unfamiliar with the issue start looking for additional information by clicking everywhere, looking for additional information. In addition, since all crowd workers who were surveyed had also successfully submitted acceptable assignments, the survey suggests that the majority were familiar with the process and the issues in question. Both the survey results and the behaviour analysis suggest that a minority of the crowd workers withdrew from the CPP process before submitting any provisional assignments.

Moreover, the survey results confirm what has been noticed in the crowd behaviour in terms of data component usefulness. The analysis observed a higher request rate on project information and peer discussion. That also confirms earlier findings in Chapter 6, where the types of information are evaluated. Surprisingly, the survey suggests that the crowd does not look for analogies at the beginning of their estimation process. Instead, they prefer to read the available information and start analysing the issue by breaking it down into smaller sub-tasks.

From the perspective of the quality model, the crowd proposal for quality criteria is very similar to the implemented version, and that lends additional credibility to Chapter 6's experimental work. Moreover, the survey results and behavioural analysis agree that the majority of crowd workers consider a revision path. Again, another similarity is that assignment

justification and estimate selection are the most reviewed component of the assignment for those workers who decided to revise their assignments.

While the systematic method of investigating the UI interaction log considers all the log events without exception, a focused investigation on peer discussion and the crowd feedback loop is also considered by filtering the logs to include only related trace entries.

By filtering the activity stream flow to show only flows that considered peer discussion, an interesting correlation between peer discussion and extra information shows up. Workers who submitted accurate estimates were relying only on peer discussion. They did not ask for or read additional information, including the issue dictionary, issue project, and issue comments. That phenomenon appeared across quality classes (A, B, and C).

On the other hand, workers who ask for or read extra information after peer discussion submit inaccurate estimates across quality classes.

In other words, if Johanna or Sarah stuck to the peer discussion without distracting themselves with information in comments or Google searches, they would probably reach a more accurate estimate. The original Planning Poker showed that peer discussion is beneficial [5], and it has similar effects here in CPP.

Moreover, such a phenomenon suggests that crowd workers can adopt a handover mechanism to process large tasks instead of breaking the tasks into smaller independent parts.

Shifting the focus towards the reading feedback activity, the usual behaviour in all classes is that workers who consider reviewing their assignment go over estimate options then the justification. Fewer workers idle then go on to read the issue instructions and basic information, and this behaviour was not exhibited by workers who submitted accurate estimates in classes A and B.

The aim of using Loss Attention is to encourage workers to take on additional work and review their assignments. In addition, the selected feedback message does not specify a particular component to review, to not expose the classifier weakness. This was also reflected by workers' behaviour in going over most of their inputs, and a smaller number went further and reviewed the instruction and issue description. Interestingly, workers who submitted accurate estimates behaved more systematically, and those who submitted inaccurate estimates showed chaos in their behaviour.

Workers who considered revision after getting Class D (Poor Quality) showed less review activity. However, the majority reviewed the time options and submitted the assignment again.

## 9.6 Summary

While the preceding chapters of the thesis have demonstrated the feasibility and reliability of CPP, fine-grained details about how crowd workers interacted within CPP process are still uncovered. Such details are essential to understanding the mechanics and dynamics of CPP and thus continuing to improve CPP.

The extended experiment in Chapter 8 has recorded a large number of UI interaction logs. Such logs offered an opportunity to investigate crowd behaviour, from which several observations can be derived. As a next step, an ethnographic methodology is used to investigate the logs.

In the literature, several studies have used ethnographic methods to understand the research group. While the traditional method is by immersing the researcher into that group, new methods using UI logs and surveys are also used in more than one study, for example Sharp and Robinson [248].

The method designed for the ethnographic study consists of three components: a systematic scanning of all UI interaction logs, a selective topic analysis, and a crowd survey. The systematised scanning aimed to provide a general understanding of different crowd behaviours, from which four persona archetypes were derived. The personas that were developed are John, Johanna, Sarah, and Smith.

Smith shows the behaviour that needs to be avoided. The poor assignments are what can be expected from Smith. He will ignore any improvement opportunity. On the other hand, Sarah does her best to submit a good quality assignment, and she considers improving her chances. However, good quality is not going to come from her. Hence, additional training and detailed feedback may benefit her. Johanna and John are the best workers, and good quality assignments are their game. Moreover, John is more accurate than Johanna, and his behaviour suggests he is focused on the analysis of the issue while reading the issue details, unlike Johanna, who is a bit distracted and acts only when needed.

The selective analysis of peer discussion shows that crowd workers rely on such information. Sarah and Johanna returned to peer discussion only after getting their provisional assignment rejected. This may explain the correlation between accurate assignments and reliance on peer discussion that is identified in the analysis.

The other selective analysis was about the crowd feedback loop. It appears that when crowd workers considered a revision, they mostly looked at the justification and their estimate selection. It also confirms the effectiveness of using loss attention as part of the improving element of the quality model. It helped to encourage the workers to review their assignments. However, Smith is not among those workers.

A hundred crowd workers were asked to take a survey to confirm the analysis findings and understand their perspectives. The results confirm similar conclusions to the systematic and selective analysis. However, looking for issue analogies was not viewed by the crowd as the best option during their estimation process, contradicting what was assumed before. The survey also doubly confirms the earlier findings of Chapter 6's experiments.



# Chapter 10

## Conclusions

Planning Poker as an expert-based Software Effort Estimation (SEE) method is investigated in this thesis with the aim of automating its process. Besides covering the literature gap in studying expert-based SEE methods, the thesis investigated several aspects of SEE using machine learning (ML) algorithms. While ML algorithms are efficient and automated, they require careful tuning and access to a large amount of context-related data. The thesis has shown that, unlike current ML-based methods, expert-based methods are more flexible and easier to adopt and that the Planning Poker method can be scaled using human computation techniques deployed on crowdsourcing platforms..

The next section summarises the research activities that took place in this thesis. Then, Section 10.2 discusses the thesis questions along with their answers. It starts with the evaluation of ML-based SEE methods and then covers questions related to CPP design, quality, automation, and crowd worker behaviour. After that, Section 10.3 explains the thesis contribution to the software engineering and human computation research disciplines. It also illustrates the learned lessons from the thesis's experimental work. After that, Section 10.4 states the thesis scope and assesses overall threats to validity, beyond those discussed for specific experiments. Finally, this chapter concludes with future research directions, including the investigation of using CPP in industry and the effect of obfuscating sensitive information on estimate reliability.

### 10.1 Summary of Research Activity

The overall work presented in this thesis concerns the development of a Planning Poker-based effort estimation method using crowds of expert workers. There were several steps in the research, starting with assessing contemporary natural language processing (NLP) techniques in building an ML model for software effort estimation based on the description

text of software development issues.

After evaluating ML algorithms, a series of research and development phases were followed to develop a semi-automated version of Planning Poker. This work began by investigating the feasibility of crowdsourcing for the process and studying what working settings could help crowd workers to play Planning Poker and give an expert-comparable estimate. In that phase, three pilot experiments were conducted to inspect different crowd settings, resulting in the Crowd Planning Poker (CPP) method. CPP is an adaptation of Planning Poker for a crowd environment. One major challenge hindering CPP from being useful was managing the crowd outcome quality.

The next phase was developing an approach to address the problem of poor quality submissions by workers in the CPP process. The problem is that the majority of workers' output was of unacceptable quality, and thus, useless. A quality model with five components was developed to resolve this quality issue of CPP. The model was able to merge ML algorithms for inspecting assignment quality and human intelligence (crowd workers) to improve their assignments. Resolving the quality issue led to an extended experimental work on CPP to confirm its performance over a large set of software development issues.

The third phase was extending the human computation implementation in CPP to reach a semi-automated CPP process that is autonomously played by crowd workers. An extended experiment with thirty issues was conducted to confirm CPP's ability in producing expert-comparable estimates. The results suggest that crowd workers using CPP are capable of predicting estimates that are similar to the expert ones. The extended CPP experiment generated a large number of crowd worker UI interaction traces.

Finally, fine-grained details about how crowd workers produced software effort estimates and which CPP component helped them most were still uncovered. Such details are essential to understanding the dynamics of CPP and thus continuing to improve CPP and software effort estimation in general. Therefore, an ethnographic-inspired method was used to follow and study crowd workers' traces. Several observations and insights into workers' behaviour were made by inspecting the UI log. In addition, four personas were developed to represent four distinguished crowd behaviours. Such personas are helpful in distributing CPP work to suitable workers while at the same time avoiding unwanted behaviour. To confirm the observations, crowd workers who took place in previous CPP experiments and had been subjects of the ethnographic study were questioned about the study findings using a survey.

## 10.2 Questions and Findings

The thesis's broad question concerns the ability to combine crowdsourcing and human computation to predict an expert-comparable estimate using a Planning-Poker-style process.

However, to answer this question, a list of seven questions need to be addressed first. These questions are grouped into five categories including: ML algorithms, CPP design, CPP quality, CPP automation, and crowd behaviour. The following subsections will address these five categories which in turn will provide a detailed answer for the thesis's broad inquiry.

### 10.2.1 ML Algorithms

Before automating an expert-based SEE method, it is wise to study previous automated methods, such as those that rely on ML algorithms. Therefore, the first question addressed in this thesis is:

*Can recent NLP advancements help in building an ML model that predicts more reliable estimates than experts?*

Chapter 5 demonstrated that current generation algorithms were not able to reliably generate effort estimates compared with experts. Despite the complication and data demands of ML methods, two of the state-of-the-art ML algorithms (BERT and RF) failed to provide reliable estimates that compared with those of experts. This conclusion conforms with the results of existing studies in the literature. However, the ensemble ML algorithm RF, performed better and it is probable that it will be helpful in inspecting crowd assignments to determine their quality.

Most likely, the lack of reliability in ML-based SEE methods originates from the fact that the estimation process needs much more information about the context. While BERT is able to extract the semantics from a text corpus, the corpus may not reflect all information about the software development task. Further, the semantic delta between two text excerpts may not be representative of the effort delta between the software development tasks that those excerpts belong to. For instance, the delta between “Develop a web application using JavaScript” and “Develop a web application using COBOL” from a semantic perspective may not reflect the delta in effort required to complete each task. That is also clearly demonstrated in the behaviour of crowd estimators, with accurate estimates being associated with estimators who exhibited a looping behaviour around task information and demanded extra background information about the software development task, as explained in Chapter 9. Therefore, a structured effort-based data model may help in enhancing the predictability of effort for software development issues.

### 10.2.2 CPP Design

Chapter 6 illustrated the design of CPP and the working settings of the crowd environment for CPP. Originally, Planning Poker was designed for in-person meetings, where participants

can discuss and exchange their views. However, in a crowd environment, that becomes infeasible. Therefore, the question that needs to be addressed here is:

*What is the proper process design that can enable crowd workers to produce software estimates using Planning Poker?*

Three main aspects of Planning Poker need to be imitated: the iterative estimation model, team discussion, and team consensus. A process that recruits crowd workers in rounds is designed to represent the iterative estimation model. Team discussion is replaced with a summary of a preceding round, asynchronously conveyed to the current round. The summary contains estimates and the justifications behind them. Consensus is calculated using Fleiss' kappa at the end of each round to determine if there is a need for an additional round or not. A level of fair agreement is used to indicate that the crowd workers have reached to a consensus. The adopted process is called Crowd Planning Poker (CPP).

Before going on to examine the CPP process design, the size of the crowd and type of background information about the software development task needs to be determined. Therefore, the following question also needs to be addressed:

*What are the proper size of the crowd team and the types and amount of information required to play Planning Poker in a crowd environment?*

Issue title and description were found to be not sufficient for crowd workers to predict an effort estimate. The workers needed extra contextual information such as information about the issue's project, developers' comments about the issue, and an explanation of abbreviations and technical terms in the issue description. A crowd of 70 workers was found to be large enough to produce a final expert-comparable estimate.

### 10.2.3 CPP Quality

Chapter 7 demonstrated that crowd assignment quality is a legendary challenge and it was found to be sensitive in the context of CPP. There are two main components of crowd quality in the context of CPP: quality assessment and improvement. What makes quality challenging in crowd settings is that there are a large number of assignments that need to be inspected. The estimate being subjective is another quality challenge. Manual processing of assignments eliminates the crowdsourcing benefits of being cheap and fast. Therefore, the first quality question is:

*Can the quality of crowd estimates be measured automatically?*

Instead of asking the crowd for estimates only, the crowd is asked to answer other associated questions, such as their experience and a justification for the estimate. Another helpful association is the trace of the worker's interaction with the CPP software UI. By collecting such

information, a classifier can be trained on earlier CPP experiments and then used to classify new assignments based on the associated information. The classifier's  $F_1$  score was 93%. After identifying a crowd assignment's quality level, there may be assignments that almost qualify for consideration, and revision may boost their quality to the next level. Thus, CPP needs a quality improvement component to help enhance those assignments, and the next quality question is:

*How can low-quality crowd estimates be improved automatically?*

Based on the crowd quality definition in Chapter 7, improving an assignment's quality will require additional human resources. Thus, the same crowd worker is asked to improve their assignment quality after it is automatically assessed. However, time is critical for crowd workers in general, and to ask a crowd worker to do additional work, they also need encouragement or additional incentives. CPP avoided additional financial incentives since this has been proved in the literature to not be effective. In addition, it increases the financial burden.

Instead, a behavioural economic theory, Loss Attention, is used to encourage the workers to do the additional work. Loss Attention suggests that individuals increase their attention and resources when losses are involved in their task. Thus, CPP used a rejection warning for assignments that were below the quality threshold before rejecting them. Many of the targeted crowd workers responded positively to the warning and either improved their assignment or withdrew from the task.

#### 10.2.4 CPP Automation

Chapter 7 demonstrated an automated crowd quality assessment in the CPP context, which opened the door to examining CPP performance in a more extended experiment, as illustrated in Chapter 8. Additional development was also added to CPP by using human computation in the administration process, meaning that the latest development made CPP autonomous. However, CPP still needed to be tested against a wide range of software development issues to address the following question about the latest version of CPP performance:

*Does the semi-automated CPP process enable the crowd to produce expert-comparable estimates?*

To answer the question, an experiment that recruited over 700 workers and estimated effort for 30 different software issues was designed and conducted. The experiment confirms CPP performance and the crowd workers' capability of producing expert-comparable estimates. Crowd workers produced the same estimates as experts in 8 trials, crowd workers were more accurate in 10 trials, and experts more accurate in 12 trials.

Furthermore, the delta of crowd to expert estimates MMRE is 102.19% across all software development issues. This is another indication of how good the crowd estimates are. It also

suggests that crowd workers are more likely to *underestimate by a category* as compared to experts who are more likely to *overestimate using person hours*.

### 10.2.5 Crowd Behaviour

A deeper understanding of crowd behaviour was essential to undertaking further CPP development. In addition, the extended experiment left a large amount of UI interaction traces, and crowd behaviour could be derived from such logs as had been done before in more than one ethnographic study. Therefore, a question about the crowd behaviours and their characterisation can be posed:

*What are the personas of crowd estimators?*

A systematised investigation of the UI log and a post-CPP crowd survey was carried out to identify four crowd personas and their behavioural archetypes. John is a calm persona who prefers to read, think, and search before acting. Most accurate and high-quality assignments will come from him. His behaviour followed a systematic pattern. Johanna is also expected to submit high-quality work but probably not accurate estimates. She prefers to act first, then, when she is stuck, she starts thinking, reading, and searching. Some chaos can be spotted in her behaviour pattern. Sarah shows unpredictable behaviour from the beginning. She seems to not understand the task and jumps from one place to another. In general, she does not produce good-quality output. However, she might be accurate sometimes. Sarah is a person that needs extra training, and revision chances helped her in enhancing her work quality. Finally, Smith shows a robotic behaviour who spends as little time as possible on producing his small assignment. He never submits good work and is not willing to accept calls for revision. Sometimes, Smith acts as a scammer.

## 10.3 Contributions and Learned Lessons

More broadly, the thesis contribution is within the context of software engineering automation. In particular, it focuses on automating effort estimation, a task that begins early in a software project, but continues throughout the lifecycle. The aim is to design more efficient effort estimation that can lift the heavy load on software developers and reduce the time spent on manual effort estimation which can be unbearable in large open-source projects.

The thesis assessed SEE by using state-of-the-art NLP and ML algorithms. The research demonstrated that these methods were inadequate due to a lack of consideration of the wider context of a task. Next, it selected one of the popular and effective expert-based methods to automate, Planning Poker. Since all expert-based SEE methods demand human resources for prediction, the automation was done by employing human computation and using a

crowd platform to orchestrate the SEE process. Finally, the thesis reviewed crowd estimator characteristics, in order to understand how crowd estimators were able to produce expert-comparable estimates. Such understanding is important to continue to improve research into SEE.

The contribution can be grouped under two umbrellas, *Knowledge* and *Artifact* contributions. The knowledge contributions are:

- Evaluation of NLP/ML-based software estimation, specifically, Random Forest and BERT, using textual features of software development tasks. These offered no better estimates than experts.
- The design and implementation of Crowd Planning Poker as a software effort estimation method for large-scale open-source development projects.
- The design and implementation of automatic quality assessment and quality improvement of CPP crowd assignments.
- Providing empirical evidence on CPP reliability compared with expert-based SEE methods.
- Identification and validation of four crowd worker effort assignment personas, providing insights into how different workers produce estimates and the artefacts that they focus on.

The thesis also contributed four artefacts that are publicly available and can be retrieved from the thesis repository<sup>1</sup>:

- JIRA Open Source Software Effort (JOSSE) dataset that consists of 16,979 data points annotated with descriptive text, actual effort, and expert estimates (only 15% of the data points are annotated with expert estimates).
- Crowd Planning Poker Behaviour (CPPB) dataset that consists of 8,338,021 data points for over 10,316 crowd workers.
- Crowd Software Effort Estimate (CSEE) dataset that consists of 507 estimates for 30 software development tasks annotated with expert estimates and actual effort.
- Crowd Planning Poker web application that is integrated with Amazon Mechanical Turk and implements the full process of CPP.

---

<sup>1</sup><https://github.com/crowd-planning-poker>

The rest of this section will cast light on more details about the thesis contributions in their chronological order.

Chapter 4 proposed the new dataset, *JOSSE*. The dataset can be used to train ML models. Most of the available data for ML SEE were old and describing whole software projects rather than smaller software development tasks such as stories and software issues. Although there are a few datasets with stories, they contain actual effort only, without expert estimation. To the researcher’s knowledge, there were no publicly accessible datasets that represent small software tasks and contain task descriptions, actual effort, and expert-based effort estimates. Thus, *JOSSE* was collected and cleaned using an extensive cleansing process. The *JOSSE* dataset is publicly available in the thesis repository<sup>2</sup>.

Then, in Chapter 5, BERT was used to extract word embeddings as a context-aware vectorisation of a text corpus and compared with the popular TF–IDF vectorisation method in a comparative study of BERT and RF models. BERT and RF research in the literature used datasets with no textual attributes, or did not use TF–IDF in comparison with a context-aware vectorisation such as BERT. However, BERT is added into the comparison matrix to reflect its impact on the accuracy of estimating the effort of software development tasks.

In the comparative study between different ML models demonstrated in Chapter 5, the *JOSSE* dataset was used with six other popular and recent datasets of software effort estimates. The comparisons were done across datasets, vectorisation methods, and ML classification algorithms. The *experimental work* of Chapter 5 led to work on developing CPP.

Two key learnings from the ML experiment are that the ensemble ML algorithm, namely RF, generally performs better than a single ML algorithm. In addition, BERT as a factorisation algorithm might not be the best option, as was assumed, especially when the targeted training model is an RF model.

Chapter 6 was the first to demonstrate the *playing of Planning Poker in a crowd environment*, as proposed by Grenning [215]. The literature, as reviewed by the researcher, has no prior work of recruiting crowd workers for estimation using a process that is inspired by Planning Poker. In that chapter, a model process of Crowd Planning Poker was designed. The model imitated three main features of the original Planning Poker. including iterative estimation, team discussion, and estimate consensus. In addition, Chapter 6 examined a variety of crowd settings that make CPP work. In particular, it evaluates what kind of information crowd workers may need to understand the software issue, and how much information should be presented to the workers. The other CPP crowd variable was the suitable size of a crowd to form a reliable estimate. By determining the settings for those variables in two different experiments, a third experiment was used to perform a preliminary assessment of

---

<sup>2</sup><https://github.com/crowd-planning-poker>



the proposed design. This experiment revealed crowd outcome quality as a challenging issue of playing Planning Poker in a crowd environment.

The learning outcome of the preliminary course of CPP experiments is that crowd workers can produce useful output, but it will be hidden by an excess of unwanted work. Thus, quality management is a major component in any crowd work and especially in subjective work such as estimating effort. In addition, navigating the crowd environment is not as easy as had been assumed. Rather, it requires a lot of administration work to reach to the final effort estimate.

To deal with the quality issue of crowd work, Chapter 7 designed a *quality model* that is built on three quality suite components: quality definition, assessment, and improvement. The model automatically assessed and improved workers' assignment quality. It used human computation to merge the ML classifier and human analysis to dramatically improve the quality of the crowd outcome. It is the first to use the behavioural economic theory of loss attention in the improving element of the quality model. Having resolved the quality issue, work was done to confirm earlier CPP findings by conducting an experiment with a wider range of software issues.

This then demonstrates that while the crowd produces a large amount of low-quality output, machines can identify this fairly clearly in the context of SEE. Despite the promotion of the crowd as mechanical work, crowd workers are still human, and they can respond to behavioural theories such as loss attention. This was a major learning point in how to deal with crowd workers. It also helps in improving the CPP design to make an autonomous process that can be run by a machine and a crowd.

Chapter 8 also used human computation to deal with the CPP administration effort and delegate the effort estimation process to the machine. The result is an *autonomous CPP process* that takes an issue and outputs its expert-comparable estimate. Moreover, the concept of using a hand-over procedure between rounds is revisited in this chapter. Instead of breaking down tasks into independent micro-tasks so that workers can work on them, a hand-over procedure is suggested to crowdsource prolonged dependent tasks. In the same chapter, an extended CPP experiment that employed over 1,449 crowd workers to estimate 30 software issues was conducted. The collected crowd estimates, along with the expert estimates and actual effort, were used to annotate the 30 software issues which were stored in the CSEE dataset.

The experimental work detailed in Chapter 8 required the development and implementation of a custom web application to reflect the logic and process design of CPP. The developed CPP web application uses the AMT API and was designed to ensure a smooth transition for crowd workers between AMT and the CPP system as if they were a single system.

Further, the experimental work in Chapter 8 resulted in the logging of a large number of

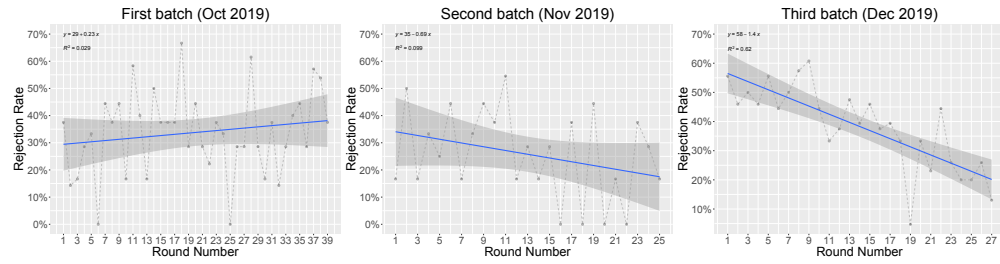


Figure 10.1: Activity stream map for poor assignments (Class D).

crowd UI interaction traces. The collected UI interaction traces represented the behaviour of crowd workers while they were playing Planning Poker. The collected data were stored in the CPPB dataset and can be accessed via the thesis publicly available repository<sup>1</sup>.

A data cleansing process was done to remove duplicate and sensitive data. In addition, any workers' identification was removed. Moreover, the eXtensible Event Stream (XES) standard was used to produce a friendly version of the log which was stored in the same thesis repository<sup>1</sup>.

A learned lesson from the extended CPP experiment is that once the CPP process is able to distinguish good and bad assignments, the accuracy of the crowd estimate is improved. Surprisingly, crowd workings often also provided extensive breakdowns of tasks as part of their rationale. Others proposed tools and places to get the right assistance. Moreover, some assignments were insightful in considering things like documentation and testing that would be needed to accomplish the task implementation. Another lesson is that, once the automated quality check is enabled and the crowd feedback loop is established, crowd workers recognise you as a requester who does due diligence and cares about quality. Thus, the number of unwanted assignments did not disappear immediately, but rather in a gradual pattern, as illustrated in Figure 10.1. The 30 software development tasks used in the extended CPP experiment and detailed in Chapter 8 were estimated by crowd workers in three batches over three months (October, November, and December). In the first batch, the trend line for the rejection rate was upward. In the second batch, the trend-line direction changed to downward. Then, in the third batch, the trend-line slope significantly increased and its direction remained downward. This further benefited workers making high-quality submissions, since they were more likely to be recognised.

On the side, crowd workers are also using special scripts and advanced technology for their own benefit. For example, a worker sent an apology message after being rejected by the quality model several times trying to justify that the tools being used by the worker were behind the bad assignments. That was surprising for the researcher, since a general assumption was that crowd HITs were being processed and resolved by humans. Yet, humans are smart,

<sup>1</sup><https://github.com/crowd-planning-poker>

and they delegate the work to machines again for their own benefit. Here is an excerpt from the worker email.

*“Hi there, I’m worker [\*\*\*\*\*] and I wanted to explain what happened with a batch of your hits today. As you may know, many Mturk workers use scripts to help them ... One of the scripts I use for this purpose was recently updated and something went horribly wrong...”*

Such a delegation resulted in a large number of low quality assignments as illustrated in previous experiments. This may make the requester job harder or may invalidate the original promise of crowdsourcing that rely on the consensus of the majority.

Chapter 9 utilised the large number of UI logs gathered in previous experiments to conduct an ethnographic study. Such a study is one among few ethnographic studies of effort estimation. By investigating the UI interaction log, four crowd estimator personas (John, Johanna, Sarah, and Smith) were developed to describe different workers engaging in their software effort estimation tasks. Their behavioural traces helped identify them and associating these personas with their outcomes.

While drawing the personas, a realisation came about which part of the CPP process should be handled by which worker. As mentioned earlier, workers are human and their skills vary, and thus, some of them may be better doing the first part of the CPP and others complete the rest.

Based on that behaviour, John is suggested for the first round of CPP, then Johanna for the next rounds. Sarah can be helpful but extra training is necessary. Smith is the crowd worker that needs to be prevented from participating in the game. Further analysis of the log data may help to derive additional conclusions..

Moreover, crowd problems do not necessarily need technical solutions; sometimes, behavioural theory can be very effective, as shown by the use of loss attention in the experiment.

## 10.4 Thesis Scope and Validity

One concern of the thesis outcome is how generalisable are the results. Since the population of software issues is unknown, non-probability sampling techniques are used. Namely, Quota sampling is used along with Convenience sampling to select the experiment’s issues. In addition, the estimation process is carried out over software issues rather than a whole software project. The advantages of this are that software issues are atomic workable items that do not significantly differ from a software project to another. This gives the flexibility to generalise the outcome to the issue or task level of a given software project, especially if it is

developed using the same technology and programming language as of the selected issues, e.g. Java.

Moreover, looking to generalisation from an estimation view, different lengths of actual efforts and difficulty levels are selected to examine the crowd worker response, regardless how accurate their estimates are. The experimental work proves that crowd workers respond positively to the issue difficulty by raising their estimate for difficult and lengthy issues and lowering their estimate for easier ones. This means that, even if the crowd worker is not accurate, they are responsive to different issues, which suggests that crowd workers are able to identify the difficulty level regardless of the issue in hand.

A limitation of the study was employing issues created for open-source projects. This decision was necessary as the experiment required a source of software tasks that could be provided to anonymous crowd workers and that had been annotated with the expert-estimated and actual work cost. This meant there was a risk that the crowd workers could access the issue trackers themselves and simply supply the actual reported cost, creating a threat to the validity of the reliability results.

This risk was mitigated in several ways. First, the issue identifiers were not supplied to the crowd workers, and issues were selected from issue trackers that required user registration. This created an additional step to deter workers. Second, workers were asked for a category of estimate, rather than an absolute person-hour value, creating an additional step if the source issue was accessed. Finally, workers were encouraged to supply their estimate and it was clear that payment was not contingent on supplying the correct result. Consequently, there is no evidence in the behaviour logs that the workers accessed project issue trackers, although this may have occurred outside the CPP user interface.

Confidentiality was a challenge that directed the research towards data available from open-source software projects. Private organisations are not willing to publish their internal software issues to be estimated by the public crowd. Although software issues from the open-source community use the same technology and programming languages, e.g. Java, timing in the context of open-source projects might not be as critical as it is in commercial software projects.

As a response, the selected issues represent the issues that have the best expert estimates among the other issues in the JOSSE dataset, as explained in Chapter 8 Section 8.2. In addition, the crowd estimates were judged by using the logged time as a ground truth. Logged time is the actual time spent in resolving an issue, excluding any kind of stopping or work pauses. In addition, the estimation is based on the issue, not the whole project, where significant differences between open-source and commercial projects accrue. Moreover, different types of issues, such as bugs and software enhancements, were selected to ensure coverage of possible issues that may occur in a commercial project.

While the result of the thesis's experimental work suggests that crowd workers are able to produce estimates as good as those of experts, it does not mean that such estimates might be acceptable for a given development team. The usability of CPP outcomes in a development team needs to be further investigated, as explained in the future work section of this chapter.

While this thesis addresses software effort estimation scalability by making Planning Poker more efficient, another scalability concern regards the availability of human capital willing to take on CPP-like tasks and produce the required estimates. Thus, it is important to investigate whether the wider scope of a software development team that includes people working for the same organisation or in the same community would demonstrate the same ability to produce effort estimates for their project. Doing so would enable CPP to be played by the same development team, which may resolve a need for external human resources to undertake the estimation tasks. Perhaps future advancements of NLP/ML algorithms may enable the processing of contextual data, which will help in closing this gap.

## 10.5 Future CPP Research Work

This thesis started the investigation of a new research niche, crowd effort estimation using Planning Poker. This means that several related topics have still to be researched and developed to improve comprehension of the method, for example, applying the proposed CPP in an industrial software development project. Moreover, other challenges have also appeared which need to be addressed, such as data confidentiality. This section will detail several future research directions.

### 10.5.1 Applying Crowd Planning Poker in an Industrial Case Study

This thesis provides answers to some fundamental questions, and it also examined some hypotheses in several open-source communities. However, there is still a gap around using CPP in an industrial context. Among other aspects, there are three demanding dimensions that need to be investigated in the industrial context:

- Effects of CPP on the software development team: this dimension should investigate how CPP can affect the development team members and their response to such a change. It also looks into the team's perspective and reception of such estimates. Will the development team accept software effort estimates predicted from outside the team, and will the team act upon those estimates?
- Reliability and efficiency of CPP estimates in an industry software project using an internal crowd, e.g. company employees, and an external public crowd, e.g. AMT.

Varying the source of the crowd may impact the estimates' reliability, especially if the crowd has a conflict of interest that may introduce bias in the final estimates. Further, using an internal crowd may seem efficient since no extra cost will accrue, but will the workers in an organisation be willing to participate in CPP?

- The best use of CPP in an industrial software development team may be the use of the estimates as informative rather than determinative. What other situations are there where CPP is most helpful, e.g. during pandemics?

However, one challenge to research in this area is data confidentiality. Commercial organisations may be reluctant to use CPP since software stories and issues may contain sensitive information that they cannot disclose to a public crowd. Such a challenge may be resolved by using an internal crowd, e.g. company employees, or obfuscation techniques. Therefore, CPP could be used by a software team to obtain an initial estimate for a task along with some initial guidance, prior to the task being triaged by a team member. Further studies are needed to understand how a software team could incorporate crowd estimates within existing triage workflows.

### **10.5.2 Investigating the Effects of Obfuscating on Estimate Reliability**

As a further work in this direction, it is necessary to investigate the extent to which issues can be obfuscated to address the confidentiality concern, without reducing the reliability of the estimate. Obfuscating can be used as a technique to resolve the issue of disclosing sensitive information among other details of a software task that needs to be estimated by a public crowd. Several directions need to be addressed in this regard:

- Obfuscating issue descriptions may have a negative impact since such details may be connected to the overall effort. Therefore, what kind of details can be obfuscated without a large impact on the estimate's reliability?
- Obfuscating may introduce generality in the software task description, therefore how can such generality be addressed or specified?
- What obfuscation techniques should be employed, and what are the contexts that they work best in?

If an obfuscation technique has no impact on estimate reliability and it can protect the originating party's confidentiality, then a software tasks repository can be proposed where each

task is annotated with estimated and actual effort to be used in future software effort estimation.

Moreover, a possible future research direction is to develop a repository to store and track effort for software stories and issues. Such a repository can help in ML SEE methods as well. It would also enable a deep investigation of factors and techniques affecting software effort reliability.

Equally important is to investigate the possibility of measuring the specificity of an issue, as issues that concern less project-specific activities may be less sensitive for a project. Doing so would take software development projects a step closer to systematised and automated software development.

### 10.5.3 Extending Crowd Planning Poker Applications

While this thesis focuses on software effort estimation using CPP, there are other applications with similar needs to software effort estimation, such as expert judgement, crowd availability, and asynchronous communication. Other applications of CPP could include budget estimation for complex projects that involve international working teams, assignee selection for large-scale development projects, and code writing for distributed development teams.

CPP also can help in assisting and assessing machine-based software development, perhaps synthesising machine and human decision making in automated software development, for instance, machine-directed software architecture. More different applications can be identified given that the need for experts can be substituted with a carefully orchestrated crowd, as demonstrated in this thesis's application to software effort estimation. Therefore, we take a step closer to the automation of software development.

### 10.5.4 Extending Ethnographic Effort Estimation Study

As demonstrated in Chapter 9, the qualitative analysis of crowd estimators' behaviour using ethnographic tools was a key to discovering and identifying four personas. Therefore, it identified future improvements as explained in this chapter, one of which is to conduct an ethnography study on the in-person Planning Poker (the original version) to address several aspects of the process, including participants, moderation, and activities.

Further, the results of such a study can be compared with the results of the ethnography study of CPP to find and address weaknesses and strengths of both methods by exchanging process designs for both methods. That might result in a hybrid method that uses CPP for the majority of issues which do not require extensive communication and calls upon the team for issues that require a face-to-face meeting in order to resolve them.

## 10.6 A Final Thought...

Automation is what software does for other real-world disciplines, including accounting, administration, and an organisation's business processes. However, despite a plethora of assistive tools, including compilers, version control systems, interactive development environments and continuous integration platforms, software development remains a labour intensive discipline. As Martin [271] argues in his classic, *Clean Code*,

*“Some have suggested that we are close to the end of code... That soon all code will be generated instead of written. That programmers simply won't be needed... Nonsense!”*

Nevertheless, while the research of software engineering has advanced toward systematised, measurable, and reproducible software development methods, there is plenty of work to do in the area to push the boundaries of the automatable in software engineering.

It is not an easy topic to tackle. Many authors refer to software development as an art, or craft, rather than an engineering discipline, see, for example the books of Martin [271], Knuth [272] and Hunt and Thomas [273]. To make headway in this agenda, rather than attempting automation in the difficult space between requirements and implementation, the automation of smaller niches in software development, e.g. automated methods to develop business-process-based software can be investigated. Contemporary software development has a range of *high friction, high frequency* micro-tasks, of which SEE is just one. Introducing automations that reduce the cost of any single task type could have significant economic benefits for the practice of software engineering.



## Appendix A

# Details of JOSSE Open Source Software Project

The JOSSE dataset consists of multiple open-source software projects. While the dataset refers to those projects using a project key, Table A lists additional details about each project.

Project	Community	Brief	Type
ACCUMULO	Apache	Apache Accumulo is a sorted key-value database based on Google's Bigtable. It is a distributed system with high scalability. It relies on other Apache systems such as Hadoop and it was developed using Java.	Database Software
AEROGEAR	jBoss	AeroGear is a cross-platform mobile application development platform. It is an open-source RedHat project that is created for enterprise mobile apps.	Mobile Development
AMBARI	Apache	Apache Ambari was designed to help system administrators manage and integrate a Hadoop cluster.	Data Processing software
ARROW	Apache	Apache Arrow is a data analytics framework for developing data software. It is based on a column-oriented format of data memory.	Data Processing software

ARTEMIS	Apache	Apache ActiveMQ is a messaging hub designed for enterprise communication systems	Enterprise system
BATCH	Spring	Spring Batch is a lightweight batch handling framework. It enables batch development for enterprise systems.	Software development
BEAM	Apache	Apache Beam provides a unified model for creating data processing pipelines used in ETL, for instance.	Data Processing software
CALCITE	Apache	Apache Calcite offers a framework and tools kit to develop data management systems.	Database Software
CARBONDATA	Apache	Apache CarbonData is column-oriented storage that can be used for Apache Hadoop ecosystem.	Database Software
DAFFODIL	Apache	Apache Daffodil implements Data Format Description Language and provides a parser to transform from DFDL format to XML/JSON.	Data Processing software
EXOJCR	JBoss	eXo JCR is an implementation of Java Specification Request.	Software development
FLINK	Apache	Apache Flink is a framework for dataflow streaming engines.	Data Processing software
GEODE	Apache	Apache Geode offers a platform to manage data. It provides consistent real-time and data-intensive access to applications using a distributed computing cloud.	Data Processing software
GTNPORTAL	JBoss	GateIn Portal is an enterprise web portal. It also provides a portal development framework.	Web Development
HDDS	Apache	Hadoop Distributed Data Store is a storage layer for distributed blocks with no namespace.	Data Processing software

IGNITE	Apache	Apache Ignite is a distributed Database caching platform that offers large Software volume storage and computation across clustered nodes.
INT	Apache	Apache Ant is a Java library for executing build files dependent on software packages.
JBAS	JBoss	Jboss Application Server, also known as WildFly. Enterprise system
JBEAP	JBoss	JBoss Enterprise Application Platform is an application server runtime platform used as an environment for transactional applications. Enterprise system
JBESB	JBoss	JBoss Enterprise Service Bus is one component of an SOA Platform. It is an integrator of Enterprise Applications. Enterprise system
JBFORUMS	JBoss	JBoss Forums is a web portlet created as subcomponent of JBoss Portal software. Web Development
JBLAB	JBoss	JBoss Labs provide an incubation environment for new JBoss projects. This project is closed. Communication Platform
JBPORTAL	JBoss	JBoss Portal offers a framework to build web portals. Web Development
JBTM	JBoss	JBoss jBPM is an engine that runs BPMN processes. It also provides a toolkit for creating process-oriented business applications. Enterprise system
METRON	Apache	Apache Metron is a security analytics framework to get benefits from security data and be able to respond to security incidents. Security Framework
MNG	Apache	Apache Maven is a software dependency management system that helps in software building, reporting, and documentation. Software development

MXNET	Apache	Apache MXNet is a deep neural networks learning framework. It offers training and deployment for neural networks.	Data Processing software
NETBEANS	Apache	NetBeans is a Java integrated development environment (IDE). It offers a Java development environment using different software modules and components.	Software development
NIFI	Apache	Apache NiFi is a data flow management framework that helps data to commute between systems.	Data Processing software
RF	JBoss	RichFaces is a library for JSF to enable AJAX in business applications.	Software development
SLING	Apache	Apache Sling is a web framework to create content-centric Java software that relies on a content repository such as Apache Jackrabbit.	Software development
SPR	Spring	Spring Framework is a Java development framework that offers a programming model for Java software.	Software development
STDCXX	Apache	Apache C++ Standard Library that offer classes and functions in C++.	Software development
STORM	Apache	Apache Storm is a framework that is written using the Clojure language for stream computation in distributed environments.	Data Processing software
STS	Spring	Spring Tool Suite is a plugin for the Eclipse IDE that is designed for Spring-based development. It offers a set of ready-made modules and libraries for deployment, debugging, and testing.	Software development
SWS	JBoss	Kiali is a management suite for connections and microservices of Istio.	Cloud Computing
TS	Apache	Traffic Server is a caching proxy server that is compliant with HTTP 1.1 and HTTP 2.	Web Server

---

ZOOKEEPER	Apache	Apache ZooKeeper provides a name registry for distributed systems.	Database Software
-----------	--------	--	-------------------

Table A.1: An overview brief of each project included in the JOSSE dataset.

---

## Appendix B

### Detailed ML Results For Each Dataset

While Chapter 5 lists a summarised version of the ML experiment, Table B.1 in this appendix shows the results for each dataset.

Dataset	Project	Method	Fold	F-Score	AUC_ROC
PPI	CPP_IN	RF-BERT	5	0.3876	0.6038
		BERT-BERT	5	0.502	0.618
		RF-TFIDF	5	0.6156	0.7538
	ACCUMULO	RF-BERT	5	0.9564	0.5214
		BERT-BERT	5	0.9668	0.594
		RF-TFIDF	5	0.9668	0.5356
	AEROGear	RF-BERT	5	0.418	0.5532
		BERT-BERT	5	0.6054	0.5112
		RF-TFIDF	5	0.514	0.6034
	AMBARI	RF-BERT	5	0.769	0.6034
		BERT-BERT	5	0.7974	0.6316
		RF-TFIDF	5	0.8458	0.5598
	ARROW	RF-BERT	5	0.7274	0.6128
		BERT-BERT	5	0.712	0.6352
		RF-TFIDF	5	0.809	0.5446
	ARTEMIS	RF-BERT	5	0.6232	0.3724
		BERT-BERT	5	0.7456	0.4784
		RF-TFIDF	5	0.6836	0.5302
	BATCH	RF-BERT	5	0.5846	0.646
		BERT-BERT	5	0.6142	0.6864

BEAM	RF-TFIDF	5	0.6438	0.6124
	RF-BERT	5	0.5192	0.5466
	BERT-BERT	5	0.5188	0.56
CALCITE	RF-TFIDF	5	0.5222	0.5298
	RF-BERT	5	0.7226	0.5754
	BERT-BERT	5	0.8202	0.5652
CARBONDATA	RF-TFIDF	5	0.7848	0.4084
	RF-BERT	5	0.5324	0.5896
	BERT-BERT	5	0.6068	0.595
EXOJCR	RF-TFIDF	5	0.5748	0.5664
	RF-BERT	5	0.2958	0.561
	BERT-BERT	5	0.4452	0.5758
GTNPORTAL	RF-TFIDF	5	0.3478	0.5604
	RF-BERT	5	0.895	0.5452
	BERT-BERT	5	0.9046	0.5902
HDDS	RF-TFIDF	5	0.9252	0.48
	RF-BERT	5	0.5564	0.576
	BERT-BERT	5	0.609	0.6132
INT	RF-TFIDF	5	0.5214	0.5574
	RF-BERT	5	0.7102	0.5448
	BERT-BERT	5	0.844	0.4788
JBAS	RF-TFIDF	5	0.8432	0.5784
	RF-BERT	5	0.5632	0.6272
	BERT-BERT	5	0.5724	0.6102
JBEAP	RF-TFIDF	5	0.5524	0.6144
	RF-BERT	2	0.3335	0.5515
	BERT-BERT	2	0.41	0.5355
JBESB	RF-TFIDF	2	0.4595	0.5575
	RF-BERT	5	0.83	0.6692
	BERT-BERT	5	0.8606	0.6018
JBFORUMS	RF-TFIDF	5	0.8078	0.7636
	RF-BERT	5	0.4626	0.5372
	BERT-BERT	5	0.4602	0.5272
JBLAB	RF-TFIDF	5	0.4592	0.5264
	RF-BERT	5	0.3288	0.4878
	BERT-BERT	5	0.5506	0.4866
	RF-TFIDF	5	0.4558	0.439
	RF-BERT	5	0.2518	0.4892

	BERT-BERT	5	0.4936	0.5174
	RF-TFIDF	5	0.336	0.4684
	RF-BERT	3	0.369	0.5427
JBPORTAL	BERT-BERT	3	0.499	0.545
	RF-TFIDF	3	0.3693	0.5263
	RF-BERT	5	0.3758	0.5038
JBTM	BERT-BERT	5	0.4762	0.4788
	RF-TFIDF	5	0.4974	0.547
	RF-BERT	5	0.6124	0.5734
METRON	BERT-BERT	5	0.6876	0.5996
	RF-TFIDF	5	0.6198	0.4862
	RF-BERT	2	0.7355	0.511
MNG	BERT-BERT	2	0.806	0.539
	RF-TFIDF	2	0.7925	0.507
	RF-BERT	5	0.5032	0.5544
MXNET	BERT-BERT	5	0.4942	0.5278
	RF-TFIDF	5	0.496	0.5138
	RF-BERT	5	0.7734	0.5156
NETBEANS	BERT-BERT	5	0.8208	0.444
	RF-TFIDF	5	0.7962	0.5426
	RF-BERT	5	0.8916	0.6108
NIFI	BERT-BERT	5	0.9108	0.6096
	RF-TFIDF	5	0.9012	0.5152
	RF-BERT	5	0.4788	0.5996
RF	BERT-BERT	5	0.5554	0.6042
	RF-TFIDF	5	0.549	0.6112
	RF-BERT	5	0.9522	0.4028
SLING	BERT-BERT	5	0.9522	0.6058
	RF-TFIDF	5	0.9522	0.5442
	RF-BERT	5	0.7708	0.5972
SPR	BERT-BERT	5	0.847	0.47
	RF-TFIDF	5	0.829	0.4334
	RF-BERT	5	0.589	0.6214
STDCXX	BERT-BERT	5	0.59	0.6322
	RF-TFIDF	5	0.512	0.551
	RF-BERT	5	0.7174	0.5176
STORM	BERT-BERT	5	0.7678	0.5322
	RF-TFIDF	5	0.802	0.5834



	STS	RF-BERT	5	0.5272	0.532
		BERT-BERT	5	0.6868	0.5502
		RF-TFIDF	5	0.575	0.4868
	SWS	RF-BERT	5	0.9092	0.525
		BERT-BERT	5	0.9304	0.5416
		RF-TFIDF	5	0.9304	0.3918
	TS	RF-BERT	5	0.5606	0.5034
		BERT-BERT	5	0.6922	0.5872
		RF-TFIDF	5	0.6976	0.6124
	ZOOKEEPER	RF-BERT	5	0.5592	0.5614
		BERT-BERT	5	0.536	0.5578
		RF-TFIDF	5	0.6096	0.5232
PORRU	MESOS	RF-BERT	5	0.3562	0.5614
		BERT-BERT	5	0.416	0.5432
		RF-TFIDF	5	0.3328	0.5654
	MULE	RF-BERT	5	0.3008	0.5786
		BERT-BERT	5	0.3542	0.5936
		RF-TFIDF	5	0.3036	0.5692
	TIMOB	RF-BERT	2	0.278	0.5175
		BERT-BERT	2	0.4195	0.52
		RF-TFIDF	2	0.3315	0.55
	XD	RF-BERT	5	0.3212	0.5646
		BERT-BERT	5	0.4262	0.6256
		RF-TFIDF	5	0.3772	0.627
DEEP-SE	APSTUD	RF-BERT	5	0.363	0.6176
		BERT-BERT	5	0.4294	0.626
		RF-TFIDF	5	0.4112	0.6226
	BAM	RF-BERT	3	0.4013	0.538
		BERT-BERT	3	0.4923	0.6027
		RF-TFIDF	3	0.3993	0.5343
	CLOV	RF-BERT	5	0.299	0.5754
		BERT-BERT	5	0.4632	0.6416
		RF-TFIDF	5	0.407	0.5914
	STUDIO	RF-BERT	5	0.3004	0.544
		BERT-BERT	5	0.4086	0.538
		RF-TFIDF	5	0.3566	0.553
	TDQ	RF-BERT	4	0.2942	0.6065
		BERT-BERT	4	0.322	0.62

TESB	RF-TFIDF	4	0.2858	0.5842
	RF-BERT	2	0.4605	0.6555
	BERT-BERT	2	0.5035	0.6865
TIMOB	RF-TFIDF	2	0.503	0.6355
	RF-BERT	3	0.3127	0.576
	BERT-BERT	3	0.4157	0.6537
	RF-TFIDF	3	0.3833	0.591

Table B.1: Performance of ML models and feature extraction methods. Both F-Score and AUC-ROC are used to measure the performance.

## **Appendix C**

### **POST Crowd Planning Poker Survey**

This appendix shows the survey that was given to crowd workers to further investigate their experience of Crowd Planning Poker. It is also used to draw and confirm behavioural conclusions, as explained in Chapter 9.

# The POST Crowd Planning Poker Survey

This is an invitation-only survey. You must enter your Worker ID. Start With Your ID.

\* Required

1. Your Worker ID \*

---

*Skip to question 2*

Here are links to the Participant Information Sheet and Consent Form. By taking this survey, you give the consent as described in the Crowd Planning Poker Survey Consent Form. Please answer the following questions to the best of your knowledge

## Participants Information Sheet

[https://gla-my.sharepoint.com/:b:/g/personal/m\\_alhamed\\_1\\_research\\_gla\\_ac\\_uk/EfzOrgxIJHNDj-J4S9WL0tgBCdjH0pDMRlvBPFX\\_KrTPEg?e=hoPfrq](https://gla-my.sharepoint.com/:b:/g/personal/m_alhamed_1_research_gla_ac_uk/EfzOrgxIJHNDj-J4S9WL0tgBCdjH0pDMRlvBPFX_KrTPEg?e=hoPfrq)

## Consent Form

[https://gla-my.sharepoint.com/:b:/g/personal/m\\_alhamed\\_1\\_research\\_gla\\_ac\\_uk/Eb6wld8ARJJJqeYymn\\_JoigBPKVcrbq7MDF4WNCpcmZF3A?e=DcJ5bd](https://gla-my.sharepoint.com/:b:/g/personal/m_alhamed_1_research_gla_ac_uk/Eb6wld8ARJJJqeYymn_JoigBPKVcrbq7MDF4WNCpcmZF3A?e=DcJ5bd)

2. How familiar you are with Planning Poker? \*

*Mark only one oval.*

	1	2	3	4	5	
Not at all familiar	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Very familiar

3. When you were estimating an issue how useful did you find each of the following sources of information. \*

(1=not at all useful, 5=very useful)

*Mark only one oval per row.*

	1	2	3	4	5
The specific issue description	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
General project information and context	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Comments made by issues contributors	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Peer estimates and justifications from previous crowd planning poker rounds	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

4. Please write any other source of information you found useful in estimation.

---

5. Thinking about the \*LAST\* issue you estimated, How familiar were you with the nature of the issue to be estimated ? \*

*Mark only one oval.*

	1	2	3	4	5	
Not at all familiar	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Very familiar with the type of project or issue

6. Thinking about the \*LAST\* issue you estimated, how much time do you allocate to each of these estimation activities? \*

*Mark only one oval per row.*

	Reading the present issue (issue description, project information and comments)	Thinking about similar issues	Analysing and breaking down the issue sub tasks	Searching the web for supplementary information
Little Time	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Some Time	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Most of the Time	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

7. Please write any other estimation activity not mentioned above and how much time you did allocate to it (Little, Some, Most of the time)

---



---



---



---



---

8. Thinking about an issue you were asked to estimate that did not seem familiar (the nature of the project or the work involved was unfamiliar), which of the following activities would you do ? \*

(1=not at all likely, 5=very likely)?

*Mark only one oval per row.*

	1	2	3	4	5
Look at every information in the page and probably click on every button to reveal as much information as possible	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Pick any estimate and write any justification	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Withdraw from the estimation page and exit	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Ask a friend or a colleagues about it	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Search for similar issues	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

9. If an activity you did for unfamiliar issue is not listed above, please write here and how likely you would do it (1=not at all likely, 5=very likely)?

---

10. When working on a Crowd Planning Poker issue, did you receive a warning about the quality of any of your estimates, saying that your assignment might be rejected? \*

*Mark only one oval.*

- ☐ Yes
- ☐ No      *Skip to question 13*
- ☐ Maybe

11. what did you do after reading the warning feedback ? \*

Select all that apply

*Check all that apply.*

- ☐ Revise my estimate selection
- ☐ Revise my justification
- ☐ Closed the webpage (exit the CPP task)
- ☐ Withdraw from the feedback page and exit
- ☐ Submit it without revising it

Other: ☐ \_\_\_\_\_

12. When I'm revising my estimate, I review the following sources of information: \*

*Check all that apply.*

- ☐ The estimate that I have selected<
- ☐ The justification and comments that I have written
- ☐ The specific issue description
- ☐ General project information and context
- ☐ Comments made by issues contributors
- ☐ Peer estimates and justifications from previous crowd planning poker rounds

Other: ☐ \_\_\_\_\_



13. What do you think makes for a good justification for an estimate?  
? \*

(1=not at all useful, 5=very useful)

Mark only one oval per row.

	1	2	3	4	5
The justification is composed of descriptions of smaller tasks	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
The justification is composed of estimates for a series of smaller sub-tasks	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
A reference to a similar issue	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
A reference to previous estimation on the same issue done in the previous round	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

14. Other things that you think they make a good justification

---

---

This content is neither created nor endorsed by Google.

Google Forms

## Bibliography

- [1] B. W. Boehm, “Software engineering economics,” *Software Pioneers*, p. 641686, 2002. [Online]. Available: [http://dx.doi.org/10.1007/978-3-642-59412-0\\_38](http://dx.doi.org/10.1007/978-3-642-59412-0_38)
- [2] M. Jørgensen, B. W. Boehm, and S. Rifkin, “Software development effort estimation: Formal models or expert judgment?” *IEEE Softw.*, vol. 26, no. 2, pp. 14–19, 2009. [Online]. Available: <https://doi.org/10.1109/MS.2009.47>
- [3] L. H. Putnam, “A general empirical solution to the macro software sizing and estimating problem,” *IEEE Trans. Software Eng.*, vol. 4, no. 4, pp. 345–361, 1978. [Online]. Available: <https://doi.org/10.1109/TSE.1978.231521>
- [4] P. M. Johnson, C. A. Moore, J. A. Dane, and R. S. Brewer, “Empirically guided software effort guesstimation,” *IEEE Softw.*, vol. 17, no. 6, pp. 51–56, 2000. [Online]. Available: <https://doi.org/10.1109/52.895168>
- [5] M. Cohn, *Agile Estimating and Planning*, ser. Robert C. Martin Series. Pearson Education, 2005.
- [6] M. Jørgensen, “A review of studies on expert estimation of software development effort,” *J. Syst. Softw.*, vol. 70, no. 1-2, pp. 37–60, 2004. [Online]. Available: [https://doi.org/10.1016/S0164-1212\(02\)00156-5](https://doi.org/10.1016/S0164-1212(02)00156-5)
- [7] ———, “Top-down and bottom-up expert estimation of software development effort,” *Information and Software Technology*, vol. 46, no. 1, p. 316, Jan 2004. [Online]. Available: [http://dx.doi.org/10.1016/s0950-5849\(03\)00093-4](http://dx.doi.org/10.1016/s0950-5849(03)00093-4)
- [8] J. Grenning, “Planning poker or how to avoid analysis paralysis while release planning,” *Hawthorn Woods: Renaissance Software Consulting*, vol. 3, pp. 22–23, 2002.
- [9] H. S. Hamza, A. Kamel, and K. M. Shams, “Software effort estimation using artificial neural networks: A survey of the current practices,” in *Tenth International Conference on Information Technology: New Generations, ITNG 2013, 15-17 April, 2013, Las Vegas, Nevada, USA*, S. Latifi, Ed. IEEE Computer Society, 2013, pp. 731–733. [Online]. Available: <https://doi.org/10.1109/ITNG.2013.111>

- [10] Z. abdelali, H. Mustapha, and N. Abdelwahed, “Investigating the use of random forest in software effort estimation,” *Procedia Computer Science*, vol. 148, p. 343352, 2019. [Online]. Available: <http://dx.doi.org/10.1016/j.procs.2019.01.042>
- [11] A. Corazza, S. D. Martino, F. Ferrucci, C. Gravino, F. Sarro, and E. Mendes, “How effective is tabu search to configure support vector regression for effort estimation?” in *Proceedings of the 6th International Conference on Predictive Models in Software Engineering, PROMISE 2010, Timisoara, Romania, September 12-13, 2010*, T. Menzies and G. Koru, Eds. ACM, 2010, p. 4. [Online]. Available: <https://doi.org/10.1145/1868328.1868335>
- [12] L. von Ahn, “Human computation, december 7, 2005,” Ph.D. dissertation, Carnegie Mellon University, Pittsburgh, PA, 2005.
- [13] L. von Ahn, M. Blum, N. J. Hopper, and J. Langford, “CAPTCHA: using hard AI problems for security,” in *Advances in Cryptology - EUROCRYPT 2003, International Conference on the Theory and Applications of Cryptographic Techniques, Warsaw, Poland, May 4-8, 2003, Proceedings*, ser. Lecture Notes in Computer Science, E. Biham, Ed., vol. 2656. Springer, 2003, pp. 294–311. [Online]. Available: [https://doi.org/10.1007/3-540-39200-9\\_18](https://doi.org/10.1007/3-540-39200-9_18)
- [14] A. J. Quinn and B. B. Bederson, “Human computation,” *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, May 2011. [Online]. Available: <http://dx.doi.org/10.1145/1978942.1979148>
- [15] K. Mao, L. Capra, M. Harman, and Y. Jia, “A survey of the use of crowdsourcing in software engineering,” *J. Syst. Softw.*, vol. 126, pp. 57–84, 2017. [Online]. Available: <https://doi.org/10.1016/j.jss.2016.09.015>
- [16] S. Grimstad, M. Jørgensen, and K. Moløkken-Østvold, “Software effort estimation terminology: The tower of Babel,” *Inf. Softw. Technol.*, vol. 48, no. 4, pp. 302–310, 2006. [Online]. Available: <https://doi.org/10.1016/j.infsof.2005.04.004>
- [17] T. S. Group, “The CHAOS report 2015,” The Standish Group, Tech. Rep., 2015.
- [18] M. Usman, E. Mendes, and J. Börstler, “Effort estimation in agile software development: a survey on the state of the practice,” in *Proceedings of the 19th International Conference on Evaluation and Assessment in Software Engineering, EASE 2015, Nanjing, China, April 27-29, 2015*, J. Lv, H. J. Zhang, and M. A. Babar, Eds. ACM, 2015, pp. 12:1–12:10. [Online]. Available: <https://doi.org/10.1145/2745802.2745813>

- [19] L. Taff, J. Borchering, and W. Hudgins, "Estimeetings: development estimates and a front-end process for a large project," *IEEE Transactions on Software Engineering*, vol. 17, no. 8, p. 839849, 1991. [Online]. Available: <http://dx.doi.org/10.1109/32.83918>
- [20] K. Schwaber, *Agile project management with Scrum*. Microsoft press, 2004.
- [21] "Welcome to kernel.org bugzilla." [Online]. Available: <https://bugzilla.kernel.org/>
- [22] "Firefox bug list at bugzilla." [Online]. Available: <https://bugzilla.mozilla.org/buglist.cgi?product=Firefox>
- [23] "Red har issue tracker." [Online]. Available: <https://issues.redhat.com/projects>
- [24] A. Trendowicz and R. Jeffery, *Software Project Effort Estimation - Foundations and Best Practice Guidelines for Success*. Springer, 2014. [Online]. Available: <https://doi.org/10.1007/978-3-319-03629-8>
- [25] K. Moharrerri, A. V. Sapre, J. Ramanathan, and R. Ramnath, "Cost-effective supervised learning models for software effort estimation in agile environments," *2016 IEEE 40th Annual Computer Software and Applications Conference (COMPSAC)*, Jun 2016. [Online]. Available: <http://dx.doi.org/10.1109/compsac.2016.85>
- [26] K. Moløkken-Østfold, N. C. Haugen, and H. C. Benestad, "Using planning poker for combining expert estimates in software projects," *J. Syst. Softw.*, vol. 81, no. 12, pp. 2106–2117, 2008. [Online]. Available: <https://doi.org/10.1016/j.jss.2008.03.058>
- [27] E. Yechiam and G. Hochman, "Loss-aversion or loss-attention: The impact of losses on cognitive performance," *Cognitive Psychology*, vol. 66, no. 2, p. 212231, Mar 2013. [Online]. Available: <http://dx.doi.org/10.1016/j.cogpsych.2012.12.001>
- [28] V. R. Basili, "The experimental paradigm in software engineering," in *Experimental Software Engineering Issues: Critical Assessment and Future Directions, International Workshop Dagstuhl Castle, Germany, September 14-18, 1992, Proceedings*, ser. Lecture Notes in Computer Science, H. D. Rombach, V. R. Basili, and R. W. Selby, Eds., vol. 706. Springer, 1992, pp. 3–12. [Online]. Available: [https://doi.org/10.1007/3-540-57092-6\\_91](https://doi.org/10.1007/3-540-57092-6_91)
- [29] V. Basili, "The role of experimentation in software engineering: past, current, and future," *Proceedings of IEEE 18th International Conference on Software Engineering*, 1996. [Online]. Available: <http://dx.doi.org/10.1109/icse.1996.493439>

- [30] D. I. K. Sjøberg, T. Dybå, and M. Jørgensen, “The future of empirical methods in software engineering research,” in *International Conference on Software Engineering, ISCE 2007, Workshop on the Future of Software Engineering, FOSE 2007, May 23-25, 2007, Minneapolis, MN, USA*, L. C. Briand and A. L. Wolf, Eds. IEEE Computer Society, 2007, pp. 358–378. [Online]. Available: <https://doi.org/10.1109/FOSE.2007.30>
- [31] D. Budgen and P. Brereton, “Performing systematic literature reviews in software engineering,” *Proceedings of the 28th international conference on Software engineering*, May 2006. [Online]. Available: <http://dx.doi.org/10.1145/1134285.1134500>
- [32] A. Brooks, M. Roper, M. Wood, J. Daly, and J. Miller, “Replication's role in software engineering,” in *Guide to Advanced Empirical Software Engineering*. Springer London, 2008, pp. 365–379. [Online]. Available: [https://doi.org/10.1007/978-1-84800-044-5\\_14](https://doi.org/10.1007/978-1-84800-044-5_14)
- [33] F. Shull, V. R. Basili, J. C. Carver, J. C. Maldonado, G. H. Travassos, M. G. Mendonça, and S. C. P. F. Fabbri, “Replicating software engineering experiments: Addressing the tacit knowledge problem,” in *2002 International Symposium on Empirical Software Engineering (ISESE 2002), 3-4 October 2002, Nara, Japan*. IEEE Computer Society, 2002, pp. 7–16. [Online]. Available: <https://doi.org/10.1109/ISESE.2002.1166920>
- [34] N. Juristo and S. Vegas, “The role of non-exact replications in software engineering experiments,” *Empirical Software Engineering*, vol. 16, no. 3, p. 295324, Aug 2010. [Online]. Available: <http://dx.doi.org/10.1007/s10664-010-9141-9>
- [35] I. Turc, M.-W. Chang, K. Lee, and K. Toutanova, “Well-read students learn better: On the importance of pre-training compact models,” *arXiv preprint arXiv:1908.08962v2*, 2019.
- [36] S. Easterbrook, J. Singer, M. D. Storey, and D. E. Damian, “Selecting empirical methods for software engineering research,” in *Guide to Advanced Empirical Software Engineering*, F. Shull, J. Singer, and D. I. K. Sjøberg, Eds. Springer, 2008, pp. 285–311. [Online]. Available: [https://doi.org/10.1007/978-1-84800-044-5\\_11](https://doi.org/10.1007/978-1-84800-044-5_11)
- [37] C. Wohlin, P. Runeson, M. Hst, M. C. Ohlsson, B. Regnell, and A. Wessln, *Experimentation in Software Engineering*. Springer Berlin Heidelberg, 2012. [Online]. Available: <http://dx.doi.org/10.1007/978-3-642-29044-2>
- [38] R. L. Glass, “Pilot studies: What, why, and how,” *J. Syst. Softw.*, vol. 36, no. 1, pp. 85–97, 1997. [Online]. Available: [https://doi.org/10.1016/0164-1212\(95\)00197-2](https://doi.org/10.1016/0164-1212(95)00197-2)

- [39] M. Kasunic, "Conducting effective pilot studies," CARNEGIE-MELLON UNIV PITTSBURGH PA SOFTWARE ENGINEERING INST, Tech. Rep., 2004.
- [40] F. Macias, M. Holcombe, and M. Gheorghe, "Empirical experiments with xp," in *Proc. 3rd International Conference on eXtreme Programming and Agile Processes in Software Engineering-XP*, 2002, pp. 225–228.
- [41] D. S. Janzen, C. S. Turner, and H. Saiedian, "Empirical software engineering in industry short courses," in *20th Conference on Software Engineering Education and Training (CSEE&T 2007)*, 3-5 July 2007, Dublin, Ireland. IEEE Computer Society, 2007, pp. 89–96. [Online]. Available: <https://doi.org/10.1109/CSEET.2007.20>
- [42] P. Brink and M. Wood, *Advanced Design in Nursing Research*. SAGE Publications, 1998. [Online]. Available: <https://books.google.co.uk/books?id=hDRwa-JwmdcC>
- [43] E. van Teijlingen and V. Hundley, "The importance of pilot studies," *Nursing Standard*, vol. 16, no. 40, p. 3336, Jun 2002. [Online]. Available: <http://dx.doi.org/10.7748/ns2002.06.16.40.33.c3214>
- [44] L. M. Connelly, "Pilot studies," *Medsurg Nursing*, vol. 17, no. 6, p. 411, 2008.
- [45] "University of Glasgow - Colleges - College of Science & Engineering - Information for staff - Committees - Ethics Committee." [Online]. Available: <https://www.gla.ac.uk/colleges/scienceengineering/staff/committees/ethicscommittee/>
- [46] S. Pink, H. Horst, J. Postill, L. Hjorth, T. Lewis, and J. Tacchi, *Digital Ethnography: Principles and Practice*. SAGE Publications, 2015. [Online]. Available: <https://books.google.co.uk/books?id=tKViCgAAQBAJ>
- [47] M. Kim, L. Bergman, T. Lau, and D. Notkin, "An ethnographic study of copy and paste programming practices in oopl," *Proceedings. 2004 International Symposium on Empirical Software Engineering, 2004. ISESE 04.*, 2004. [Online]. Available: <http://dx.doi.org/10.1109/isese.2004.1334896>
- [48] [Online]. Available: <http://dx.doi.org/10.1109/ieeestd.2017.8016712>
- [49] T. DeMarco, *Controlling Software Projects: Management, Measurement & Estimation*, ser. Yourdon computing series. Yourdon Press, 1982.
- [50] P. Bourque and R. E. Fairley, *SWEBOK: Guide to the software engineering body of knowledge*. IEEE Computer Society, 2014.
- [51] B. Leonard, *GAO Cost estimating and assessment guide: Best practices for developing and managing capital program costs*. DIANE Publishing, 2009. [Online]. Available: [www.whitehouse.gov/omb/circulars/index.html](http://www.whitehouse.gov/omb/circulars/index.html).

- [52] P. G. Armour, "To plan, two plans," *Commun. ACM*, vol. 48, no. 9, pp. 15–19, 2005. [Online]. Available: <https://doi.org/10.1145/1081992.1082007>
- [53] F. P. B. Jr., *The mythical man-month - essays on software engineering (2. ed.)*. Addison-Wesley, 1995.
- [54] I. Sommerville, *Software engineering, 8th Edition*, ser. International computer science series. Addison-Wesley, 2007. [Online]. Available: <https://www.worldcat.org/oclc/65978675>
- [55] W. Zuill, "No Estimate Programming Series – Intro Post," dec 2012. [Online]. Available: <http://zuill.us/WoodyZuill/2012/12/10/no-estimate-programming-series-intro-post/>
- [56] M. Isaacs, "An unbiased look at the #NoEstimates debate." [Online]. Available: <https://techbeacon.com/app-dev-testing/noestimates-debate-unbiased-look-origins-arguments-thought-leaders-behind-movement>
- [57] J. Hope and R. Fraser, "Beyond budgeting: how managers can break free from the annual performance trap," *Choice Reviews Online*, vol. 41, no. 05, p. 412908412908, Jan 2004. [Online]. Available: <http://dx.doi.org/10.5860/choice.41-2908>
- [58] T. Vera, S. F. Ochoa, and D. Perovich, "Survey of software development effort estimation taxonomies," Technical Report. Pending ID. Computer Science Department, University of Chile, Tech. Rep., 2017.
- [59] K. Moløkken and M. Jørgensen, "A review of surveys on software effort estimation," in *2003 International Symposium on Empirical Software Engineering (ISESE 2003), 30 September - 1 October 2003. Rome, Italy*. IEEE Computer Society, 2003, pp. 223–231. [Online]. Available: <https://doi.org/10.1109/ISESE.2003.1237981>
- [60] R. Britto, V. Freitas, E. Mendes, and M. Usman, "Effort estimation in global software development: A systematic literature review," in *IEEE 9th International Conference on Global Software Engineering, ICGSE 2014, Shanghai, China, 18-21 August, 2014*. IEEE Computer Society, 2014, pp. 135–144. [Online]. Available: <https://doi.org/10.1109/ICGSE.2014.11>
- [61] N. Dalkey and O. Helmer, "An experimental application of the DELPHI method to the use of experts," *Management Science*, vol. 9, no. 3, pp. 458–467, apr 1963. [Online]. Available: <https://doi.org/10.1287/mnsc.9.3.458>
- [62] B. W. Boehm, "Software engineering economics," *IEEE Transactions on Software Engineering*, vol. SE-10, no. 1, p. 421, Jan 1984. [Online]. Available: <http://dx.doi.org/10.1109/tse.1984.5010193>

- [63] A. Aamodt and E. Plaza, "Case-based reasoning: Foundational issues, methodological variations, and system approaches," *AI Communications*, vol. 7, no. 1, p. 39–59, 1994. [Online]. Available: <http://doi.org/10.3233/AIC-1994-7104>
- [64] D. Yang, Y. Wan, Z. Tang, S. Wu, M. He, and M. Li, "Cocomo-u: An extension of cocomo ii for cost estimation with uncertainty," *Lecture Notes in Computer Science*, p. 132141, 2006. [Online]. Available: [http://dx.doi.org/10.1007/11754305\\_15](http://dx.doi.org/10.1007/11754305_15)
- [65] M. Jørgensen, "Forecasting of software development work effort: Evidence on expert judgement and formal models," *International Journal of Forecasting*, vol. 23, no. 3, p. 449462, Jul 2007. [Online]. Available: <http://dx.doi.org/10.1016/j.ijforecast.2007.05.008>
- [66] Digital.ai, "15th annual state of agile report," Digital.ai, Tech. Rep., 2021. [Online]. Available: <https://digital.ai/resource-center/analyst-reports/state-of-agile-report>
- [67] K. Moløkken and M. Jørgensen, "Software effort estimation: unstructured group discussion as a method to reduce individual bias." in *PPIG*, 2003.
- [68] "'guesstimate, n." OED Online," 2020. [Online]. Available: [www.oed.com/view/Entry/82257](http://www.oed.com/view/Entry/82257)
- [69] C. A. Moore, "Investigating individual software development: an evaluation of the leap toolkit," Ph.D. dissertation, University of Hawaii at Manoa, 2000.
- [70] J. A. Farquhar, "A preliminary inquiry into the software estimation process," RAND CORP SANTA MONICA CALIF, Tech. Rep., 1970.
- [71] A. Stellman and J. Greene, *Applied software project management*. " O'Reilly Media, Inc.", 2005.
- [72] M. G. Stochel, "Reliability and accuracy of the estimation process - wideband delphi vs. wisdom of crowds," in *Proceedings of the 35th Annual IEEE International Computer Software and Applications Conference, COMPSAC 2011, Munich, Germany, 18-22 July 2011*. IEEE Computer Society, 2011, pp. 350–359. [Online]. Available: <https://doi.org/10.1109/COMPSAC.2011.53>
- [73] K. Moløkken-Østfold and M. Jørgensen, "Group processes in software effort estimation," *Empir. Softw. Eng.*, vol. 9, no. 4, pp. 315–334, 2004. [Online]. Available: <https://doi.org/10.1023/B:EMSE.0000039882.39206.5a>
- [74] M. Elkjaer, "Stochastic budget simulation," *International Journal of Project Management*, vol. 18, no. 2, p. 139147, Apr 2000. [Online]. Available: [http://dx.doi.org/10.1016/s0263-7863\(98\)00078-7](http://dx.doi.org/10.1016/s0263-7863(98)00078-7)



- [75] J.-S. Chou, "Cost simulation in an item-based project involving construction engineering and management," *International Journal of Project Management*, vol. 29, no. 6, pp. 706–717, aug 2011. [Online]. Available: <https://doi.org/10.1016/j.ijproman.2010.07.010>
- [76] M. J. Shepperd and M. Cartwright, "Predicting with sparse data," *IEEE Trans. Software Eng.*, vol. 27, no. 11, pp. 987–998, 2001. [Online]. Available: <https://doi.org/10.1109/32.965339>
- [77] R. Saaty, "The analytic hierarchy process—what it is and how it is used," *Mathematical Modelling*, vol. 9, no. 3-5, pp. 161–176, 1987. [Online]. Available: <https://doi.org/10.1016%2F0270-0255%2887%2990473-8>
- [78] T. L. Saaty, "Decision-making with the AHP: why is the principal eigenvector necessary," *Eur. J. Oper. Res.*, vol. 145, no. 1, pp. 85–91, 2003. [Online]. Available: [https://doi.org/10.1016/S0377-2217\(02\)00227-8](https://doi.org/10.1016/S0377-2217(02)00227-8)
- [79] "Trademark Status & Document Retrieval: Planning Poker," jul 2008. [Online]. Available: [http://tsdr.uspto.gov/{#}caseNumber=3473287{&}caseType=US{\\_-}REGISTRATION{\\_-}NO{&}searchType=statusSearch](http://tsdr.uspto.gov/{#}caseNumber=3473287{&}caseType=US{_-}REGISTRATION{_-}NO{&}searchType=statusSearch)
- [80] M. Cohn, *User Stories Applied: For Agile Software Development*, ser. Addison-Wesley Signature Series (Beck). Pearson Education, 2004. [Online]. Available: [https://books.google.co.uk/books?id=DHZZP\\_YL3FXYC](https://books.google.co.uk/books?id=DHZZP_YL3FXYC)
- [81] V. Mahnic and T. Hovelja, "On using planning poker for estimating user stories," *J. Syst. Softw.*, vol. 85, no. 9, pp. 2086–2095, 2012. [Online]. Available: <https://doi.org/10.1016/j.jss.2012.04.005>
- [82] J. Wen, S. Li, Z. Lin, Y. Hu, and C. Huang, "Systematic literature review of machine learning based software development effort estimation models," *Inf. Softw. Technol.*, vol. 54, no. 1, pp. 41–59, 2012. [Online]. Available: <https://doi.org/10.1016/j.infsof.2011.09.002>
- [83] A. Ali and C. Gravino, "A systematic literature review of software effort prediction using machine learning methods," *J. Softw. Evol. Process.*, vol. 31, no. 10, 2019. [Online]. Available: <https://doi.org/10.1002/smr.2211>
- [84] A. Idri, T. M. Khoshgoftaar, and A. Abran, "Can neural networks be easily interpreted in software cost estimation?" in *Proceedings of the 2002 IEEE International Conference on Fuzzy Systems, FUZZ-IEEE'02, Honolulu, Hawaii, USA, May 12 - 17, 2002*. IEEE, 2002, pp. 1162–1167. [Online]. Available: <https://doi.org/10.1109/FUZZ.2002.1006668>

- [85] K. Dutta, V. Gupta, and V. S. Dave, "Analysis and comparison of neural network models for software development effort estimation," *J. Cases Inf. Technol.*, vol. 21, no. 2, pp. 88–112, 2019. [Online]. Available: <https://doi.org/10.4018/JCIT.2019040106>
- [86] A. B. Nassif, M. Azzeh, L. F. Capretz, and D. Ho, "Neural network models for software development effort estimation: a comparative study," *Neural Comput. Appl.*, vol. 27, no. 8, pp. 2369–2381, 2016. [Online]. Available: <https://doi.org/10.1007/s00521-015-2127-1>
- [87] M. M. Moshizi and V. K. Bardsiri, "The application of artificial neural networks in software effort estimation," *Journal of Advanced Computer Science and Technology Research*, vol. 7, no. 2, pp. 42–56, 2017.
- [88] A. Idri, F. a. Amazal, and A. Abran, "Analogy-based software development effort estimation: A systematic mapping and review," *Information and Software Technology*, vol. 58, p. 206230, Feb 2015. [Online]. Available: <http://dx.doi.org/10.1016/j.infsof.2014.07.013>
- [89] A. B. Nassif, M. Azzeh, L. F. Capretz, and D. Ho, "A comparison between decision trees and decision tree forest models for software development effort estimation," *2013 Third International Conference on Communications and Information Technology (ICCIT)*, Jun 2013. [Online]. Available: <http://dx.doi.org/10.1109/iccitechnology.2013.6579553>
- [90] Z. Abdelali, M. Hicham, and N. Abdelwahed, "An ensemble of optimal trees for software development effort estimation," *Lecture Notes in Networks and Systems*, p. 5568, 2019. [Online]. Available: [http://dx.doi.org/10.1007/978-3-030-11914-0\\_6](http://dx.doi.org/10.1007/978-3-030-11914-0_6)
- [91] J. Nayak, B. Naik, and H. S. Behera, "A comprehensive survey on support vector machine in data mining tasks: Applications & challenges," *International Journal of Database Theory and Application*, vol. 8, no. 1, p. 169186, Feb 2015. [Online]. Available: <http://dx.doi.org/10.14257/ijdta.2015.8.1.18>
- [92] A. Corazza, S. D. Martino, F. Ferrucci, C. Gravino, and E. Mendes, "Using support vector regression for web development effort estimation," in *Software Process and Product Measurement, International Conferences IWSM 2009 and Mensura 2009, Amsterdam, The Netherlands, November 4-6, 2009. Proceedings*, ser. Lecture Notes in Computer Science, A. Abran, R. Braungarten, R. R. Dumke, J. J. Cuadrado-Gallego, and J. Brunekreef, Eds., vol. 5891. Springer, 2009, pp. 255–271. [Online]. Available: [https://doi.org/10.1007/978-3-642-05415-0\\_19](https://doi.org/10.1007/978-3-642-05415-0_19)

- [93] J.-M. Desharnais, "Analyse statistique de la productivite des projets de developpement en informatique a partir de la technique des points de fonction," *Master's thesis, Univ. du Quebec a Montreal*, 1989.
- [94] J. W. Bailey and V. R. Basili, "A meta-model for software development resource expenditures," in *Proceedings of the 5th international conference on Software engineering*. IEEE Press, 1981, pp. 107–116.
- [95] A. J. Albrecht and J. E. G. Jr., "Software function, source lines of code, and development effort prediction: A software science validation," *IEEE Trans. Software Eng.*, vol. 9, no. 6, pp. 639–648, 1983. [Online]. Available: <https://doi.org/10.1109/TSE.1983.235271>
- [96] Y. Kultur, B. Turhan, and A. B. Bener, "Ensemble of neural networks with associative memory (ENNA) for estimating software development costs," *Knowl. Based Syst.*, vol. 22, no. 6, pp. 395–402, 2009. [Online]. Available: <https://doi.org/10.1016/j.knosys.2009.05.001>
- [97] S. M. Satapathy and S. K. Rath, "Effort estimation of web-based applications using machine learning techniques," in *2016 International Conference on Advances in Computing, Communications and Informatics, ICACCI 2016, Jaipur, India, September 21-24, 2016*. IEEE, 2016, pp. 973–979. [Online]. Available: <https://doi.org/10.1109/ICACCI.2016.7732171>
- [98] F. S. Gharehchopogh, "Neural networks application in software cost estimation: A case study," in *2011 International Symposium on Innovations in Intelligent Systems and Applications*. IEEE, jun 2011. [Online]. Available: <https://doi.org/10.1109/inista.2011.5946160>
- [99] B. Baskeles, B. Turhan, and A. Bener, "Software effort estimation using machine learning methods," in *2007 22nd international symposium on computer and information sciences*. IEEE, nov 2007. [Online]. Available: <https://doi.org/10.1109/iscis.2007.4456863>
- [100] S. M. Satapathy, B. P. Acharya, and S. K. Rath, "Early stage software effort estimation using random forest technique based on use case points," *IET Softw.*, vol. 10, no. 1, pp. 10–17, 2016. [Online]. Available: <https://doi.org/10.1049/iet-sen.2014.0122>
- [101] J. Shivhare and S. K. Rath, "Software effort estimation using machine learning techniques," in *7th India Software Engineering Conference, Chennai, ISEC '14, Chennai, India - February 19 - 21, 2014*, D. Janakiram, K. Sen, and V. Kulkarni, Eds. ACM, 2014, pp. 19:1–19:6. [Online]. Available: <https://doi.org/10.1145/2590748.2590767>

- [102] E. Mendes, N. Mosley, and S. Counsell, "Investigating web size metrics for early web cost estimation," *J. Syst. Softw.*, vol. 77, no. 2, pp. 157–172, 2005. [Online]. Available: <https://doi.org/10.1016/j.jss.2004.08.034>
- [103] M. Kassab and G. Destefanis, "Requirements effort estimation: The state of the practice," *Collegium of Economic Analysis Annals*, no. 43, pp. 87–102, 2017.
- [104] D. Yang, Q. Wang, M. Li, Y. Yang, K. Ye, and J. Du, "A survey on software cost estimation in the chinese software industry," in *Proceedings of the Second International Symposium on Empirical Software Engineering and Measurement, ESEM 2008, October 9-10, 2008, Kaiserslautern, Germany*, H. D. Rombach, S. G. Elbaum, and J. Münch, Eds. ACM, 2008, pp. 253–262. [Online]. Available: <https://doi.org/10.1145/1414004.1414045>
- [105] K. Moløkken-Østvold, M. Jørgensen, S. S. Tanilkan, H. Gallis, A. C. Lien, and S. E. Hove, "A survey on software estimation in the norwegian industry," in *10th IEEE International Software Metrics Symposium (METRICS 2004), 11-17 September 2004, Chicago, IL, USA*. IEEE Computer Society, 2004, pp. 208–219. [Online]. Available: <https://doi.org/10.1109/METRIC.2004.1357904>
- [106] B. Tanveer, L. Guzmán, and U. M. Engel, "Effort estimation in agile software development: Case study and improvement framework," *J. Softw. Evol. Process.*, vol. 29, no. 11, 2017. [Online]. Available: <https://doi.org/10.1002/smr.1862>
- [107] M. Usman, R. Britto, L.-O. Damm, and J. Brstler, "Effort estimation in large-scale software development: An industrial case study," *Information and Software Technology*, vol. 99, p. 2140, Jul 2018. [Online]. Available: <http://dx.doi.org/10.1016/j.infsof.2018.02.009>
- [108] S. K. Sehra, Y. S. Brar, N. Kaur, and S. S. Sehra, "Research patterns and trends in software effort estimation," *Inf. Softw. Technol.*, vol. 91, pp. 1–21, 2017. [Online]. Available: <https://doi.org/10.1016/j.infsof.2017.06.002>
- [109] T. T. Moores and J. S. Edwards, "Could large uk corporations and computing companies use software cost estimating tools? a survey," *European Journal of Information Systems*, vol. 1, no. 5, p. 311320, May 1992. [Online]. Available: <http://dx.doi.org/10.1057/ejis.1992.3>
- [110] [Online]. Available: <http://dx.doi.org/10.1109/ieeestd.2017.7955095>
- [111] T. Addison and S. Vallabh, "Controlling software project risks: an empirical study of methods used by experienced project managers," in *Proceedings of the 2002 annual*

- research conference of the South African institute of computer scientists and information technologists on Enablement through technology.* South African Institute for Computer Scientists and Information Technologists, 2002, pp. 128–140.
- [112] K. Moløkken-Østvold, “Effort and schedule estimation of software development projects,” Ph.D. dissertation, PhD thesis, University of Oslo, Norway, 2004.
- [113] G. Little, L. B. Chilton, M. Goldman, and R. C. Miller, “Turkit: human computation algorithms on mechanical turk,” in *Proceedings of the 23rd Annual ACM Symposium on User Interface Software and Technology, New York, NY, USA, October 3-6, 2010*, K. Perlin, M. Czerwinski, and R. Miller, Eds. ACM, 2010, pp. 57–66. [Online]. Available: <https://doi.org/10.1145/1866029.1866040>
- [114] M. S. Bernstein, G. Little, R. C. Miller, B. Hartmann, M. S. Ackerman, D. R. Karger, D. Crowell, and K. Panovich, “Soylent: a word processor with a crowd inside,” *Commun. ACM*, vol. 58, no. 8, pp. 85–94, 2015. [Online]. Available: <https://doi.org/10.1145/2791285>
- [115] E. Kamar, S. Hacker, and E. Horvitz, “Combining human and machine intelligence in large-scale crowdsourcing,” in *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems - Volume 1*, ser. AAMAS ’12. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems, 2012, p. 467–474.
- [116] D. Sobel, *Longitude : the true story of a lone genius who solved the greatest scientific problem of his time.* London: Harper Perennial, 2005.
- [117] J. Howe, “The rise of crowdsourcing,” *Wired magazine*, vol. 14, no. 6, pp. 1–4, 2006.
- [118] E. E. Arolas and F. González-Ladrón-de-Guevara, “Towards an integrated crowdsourcing definition,” *J. Inf. Sci.*, vol. 38, no. 2, pp. 189–200, 2012. [Online]. Available: <https://doi.org/10.1177/0165551512437638>
- [119] M. Hosseini, A. Shahri, K. Phalp, J. Taylor, and R. Ali, “Crowdsourcing: A taxonomy and systematic mapping study,” *Comput. Sci. Rev.*, vol. 17, pp. 43–69, 2015. [Online]. Available: <https://doi.org/10.1016/j.cosrev.2015.05.001>
- [120] T. W. Malone, R. Laubacher, and C. N. Dellarocas, “Harnessing crowds: Mapping the genome of collective intelligence,” *SSRN Electronic Journal*, 2009. [Online]. Available: <http://dx.doi.org/10.2139/ssrn.1381502>
- [121] H. Landemore, *Democratic Reason.* Princeton University Press, Dec 2012. [Online]. Available: <http://dx.doi.org/10.23943/princeton/9780691155654.001.0001>

- [122] D. Miorandi, V. Maltese, M. Rovatsos, A. Nijholt, and J. Stewart, Eds., *Social Collective Intelligence*. Springer International Publishing, 2014. [Online]. Available: <http://dx.doi.org/10.1007/978-3-319-08681-1>
- [123] E. Bonabeau, “Decisions 2.0: The power of collective intelligence,” *MIT Sloan management review*, vol. 50, no. 2, p. 45, 2009.
- [124] F. Wang, K. M. Carley, D. Zeng, and W. Mao, “Social computing: From social informatics to social intelligence,” *IEEE Intell. Syst.*, vol. 22, no. 2, pp. 79–83, 2007. [Online]. Available: <https://doi.org/10.1109/MIS.2007.41>
- [125] M. Niazi, S. Mahmood, M. Alshayeb, A. A. B. Baqais, and A. Q. Gill, “Motivators for adopting social computing in global software development: An empirical study,” *J. Softw. Evol. Process.*, vol. 29, no. 8, 2017. [Online]. Available: <https://doi.org/10.1002/smr.1872>
- [126] M. Ortu, G. Destefanis, B. Adams, A. Murgia, M. Marchesi, and R. Tonelli, “The JIRA repository dataset: Understanding social aspects of software development,” in *Proceedings of the 11th International Conference on Predictive Models and Data Analytics in Software Engineering, PROMISE 2015, Beijing, China, October 21, 2015*, A. Bener, L. L. Minku, and B. Turhan, Eds. ACM, 2015, pp. 1:1–1:4. [Online]. Available: <https://doi.org/10.1145/2810146.2810147>
- [127] M. D. Storey, A. Zagalsky, F. M. F. Filho, L. Singer, and D. M. Germán, “How social and communication channels shape and challenge a participatory culture in software development,” *IEEE Trans. Software Eng.*, vol. 43, no. 2, pp. 185–204, 2017. [Online]. Available: <https://doi.org/10.1109/TSE.2016.2584053>
- [128] M. Allahbakhsh, B. Benatallah, A. Ignjatovic, H. R. M. Nezhad, E. Bertino, and S. Dustdar, “Quality control in crowdsourcing systems: Issues and directions,” *IEEE Internet Comput.*, vol. 17, no. 2, pp. 76–81, 2013. [Online]. Available: <https://doi.org/10.1109/MIC.2013.20>
- [129] J. Corney, C. Torres-Sánchez, A. Jagadeesan, X. Yan, W. Regli, and H. Medellin, “Putting the crowd to work in a knowledge-based factory,” *Advanced Engineering Informatics*, vol. 24, no. 3, p. 243250, Aug 2010. [Online]. Available: <http://dx.doi.org/10.1016/j.aei.2010.05.011>
- [130] P. Donmez, J. G. Carbonell, and J. G. Schneider, “Efficiently learning the accuracy of labeling sources for selective sampling,” in *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Paris, France, June 28 - July 1, 2009*, J. F. E. IV, F. Fogelman-Soulié, P. A.

- Flach, and M. J. Zaki, Eds. ACM, 2009, pp. 259–268. [Online]. Available: <https://doi.org/10.1145/1557019.1557053>
- [131] V. Bhardwaj, R. J. Passonneau, A. Salieb-Aouissi, and N. Ide, “Anveshan: A framework for analysis of multiple annotators’ labeling behavior,” in *Proceedings of the Fourth Linguistic Annotation Workshop*, ser. LAW IV ’10. USA: Association for Computational Linguistics, 2010, p. 47–55.
- [132] D. Oleson, A. Sorokin, G. Laughlin, V. Hester, J. Le, and L. Biewald, “Programmatic gold: Targeted and scalable quality assurance in crowdsourcing,” in *Workshops at the Twenty-Fifth AAAI Conference on Artificial Intelligence*, 2011.
- [133] S. Zhu, S. K. Kane, J. Feng, and A. Sears, “A crowdsourcing quality control model for tasks distributed in parallel,” in *CHI Conference on Human Factors in Computing Systems, CHI ’12, Extended Abstracts Volume, Austin, TX, USA, May 5-10, 2012*, J. A. Konstan, E. H. Chi, and K. Höök, Eds. ACM, 2012, pp. 2501–2506. [Online]. Available: <https://doi.org/10.1145/2212776.2223826>
- [134] A. J. Mashhadi and L. Capra, “Quality control for real-time ubiquitous crowdsourcing,” *Proceedings of the 2nd international workshop on Ubiquitous crowdsourcing - UbiCrowd 11*, 2011. [Online]. Available: <http://dx.doi.org/10.1145/2030100.2030103>
- [135] P. G. Ipeirotis, F. Provost, and J. Wang, “Quality management on amazon mechanical turk,” *Proceedings of the ACM SIGKDD Workshop on Human Computation - HCOMP 10*, 2010. [Online]. Available: <http://dx.doi.org/10.1145/1837885.1837906>
- [136] A. D. Sarma, A. G. Parameswaran, and J. Widom, “Towards globally optimal crowdsourcing quality management: The uniform worker setting,” in *Proceedings of the 2016 International Conference on Management of Data, SIGMOD Conference 2016, San Francisco, CA, USA, June 26 - July 01, 2016*, F. Özcan, G. Koutrika, and S. Madden, Eds. ACM, 2016, pp. 47–62. [Online]. Available: <https://doi.org/10.1145/2882903.2882953>
- [137] Y. Baba and H. Kashima, “Statistical quality estimation for general crowdsourcing tasks,” in *The 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2013, Chicago, IL, USA, August 11-14, 2013*, I. S. Dhillon, Y. Koren, R. Ghani, T. E. Senator, P. Bradley, R. Parekh, J. He, R. L. Grossman, and R. Uthrusamy, Eds. ACM, 2013, pp. 554–562. [Online]. Available: <https://doi.org/10.1145/2487575.2487600>

- [138] T. D. LaToza, W. Ben Towne, A. van der Hoek, and J. D. Herbsleb, "Crowd development," *2013 6th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE)*, May 2013. [Online]. Available: <http://dx.doi.org/10.1109/chase.2013.6614737>
- [139] S. L. Lim and A. Finkelstein, "Stakerare: Using social networks and collaborative filtering for large-scale requirements elicitation," *IEEE Transactions on Software Engineering*, vol. 38, no. 3, p. 707735, May 2012. [Online]. Available: <http://dx.doi.org/10.1109/tse.2011.36>
- [140] K.-J. Stol and B. Fitzgerald, "Twos company, threes a crowd: a case study of crowdsourcing software development," *Proceedings of the 36th International Conference on Software Engineering*, May 2014. [Online]. Available: <http://dx.doi.org/10.1145/2568225.2568249>
- [141] K. Stol and B. Fitzgerald, "Researching crowdsourcing software development: perspectives and concerns," in *Proceedings of the 1st International Workshop on CrowdSourcing in Software Engineering, CSI-SE 2014, Hyderabad, India, June 2, 2014*, G. Fraser, T. D. LaToza, L. Mariani, F. Pastore, and N. Tillmann, Eds. ACM, 2014, pp. 7–10. [Online]. Available: <https://doi.org/10.1145/2593728.2593731>
- [142] W. Li, M. N. Huhns, W.-T. Tsai, and W. Wu, Eds., *Crowdsourcing: Cloud-Based Software Development*. Springer Berlin Heidelberg, 2015. [Online]. Available: <https://doi.org/10.1007/978-3-662-47011-4>
- [143] H. Yu, C. Miao, Z. Shen, J. Lin, C. Leung, and Q. Yang, "Infusing human factors into algorithmic crowdsourcing," in *Twenty-Eighth IAAI Conference*, 2016.
- [144] W.-T. Tsai, W. Wu, and M. N. Huhns, "Cloud-based software crowdsourcing," *IEEE Internet Computing*, vol. 18, no. 3, p. 7883, May 2014. [Online]. Available: <http://dx.doi.org/10.1109/mic.2014.46>
- [145] P. Minder and A. Bernstein, "Crowdlang: A programming language for the systematic exploration of human computation systems," in *Social Informatics - 4th International Conference, SocInfo 2012, Lausanne, Switzerland, December 5-7, 2012. Proceedings*, ser. Lecture Notes in Computer Science, K. Aberer, A. Flache, W. Jager, L. Liu, J. Tang, and C. Guéret, Eds., vol. 7710. Springer, 2012, pp. 124–137. [Online]. Available: [https://doi.org/10.1007/978-3-642-35386-4\\_10](https://doi.org/10.1007/978-3-642-35386-4_10)
- [146] R. A. Cochran, L. D'Antoni, B. Livshits, D. Molnar, and M. Veanes, "Program boosting: Program synthesis via crowd-sourcing," in *Proceedings of the 42nd Annual ACM SIGPLAN-SIGACT Symposium on Principles of*



- Programming Languages, POPL 2015, Mumbai, India, January 15-17, 2015*, S. K. Rajamani and D. Walker, Eds. ACM, 2015, pp. 677–688. [Online]. Available: <https://doi.org/10.1145/2676726.2676973>
- [147] T. D. LaToza, W. B. Towne, C. M. Adriano, and A. van der Hoek, “Microtask programming: building software with a crowd,” in *The 27th Annual ACM Symposium on User Interface Software and Technology, UIST '14, Honolulu, HI, USA, October 5-8, 2014*, H. Benko, M. Dontcheva, and D. Wigdor, Eds. ACM, 2014, pp. 43–54. [Online]. Available: <https://doi.org/10.1145/2642918.2647349>
- [148] E. Raymond, “The cathedral and the bazaar,” *Knowledge, Technology & Policy*, vol. 12, no. 3, p. 2349, Sep 1999. [Online]. Available: <http://dx.doi.org/10.1007/s12130-999-1026-0>
- [149] M. Jiménez, M. Piattini, and A. Vizcaíno, “Challenges and improvements in distributed software development: A systematic review,” *Adv. Softw. Eng.*, vol. 2009, pp. 710971:1–710971:14, 2009. [Online]. Available: <https://doi.org/10.1155/2009/710971>
- [150] M. Harman and B. F. Jones, “Search-based software engineering,” *Information and Software Technology*, vol. 43, no. 14, p. 833839, Dec 2001. [Online]. Available: [http://dx.doi.org/10.1016/s0950-5849\(01\)00189-6](http://dx.doi.org/10.1016/s0950-5849(01)00189-6)
- [151] A. Kittur, J. V. Nickerson, M. S. Bernstein, E. Gerber, A. D. Shaw, J. Zimmerman, M. Lease, and J. J. Horton, “The future of crowd work,” in *Computer Supported Cooperative Work, CSCW 2013, San Antonio, TX, USA, February 23-27, 2013*, A. S. Bruckman, S. Counts, C. Lampe, and L. G. Terveen, Eds. ACM, 2013, pp. 1301–1318. [Online]. Available: <https://doi.org/10.1145/2441776.2441923>
- [152] V. Casey, “Virtual software team project management,” *J. Braz. Comput. Soc.*, vol. 16, no. 2, pp. 83–96, 2010. [Online]. Available: <https://doi.org/10.1007/s13173-010-0013-3>
- [153] D. Cubranic, G. C. Murphy, J. Singer, and K. S. Booth, “Hipikat: A project memory for software development,” *IEEE Trans. Software Eng.*, vol. 31, no. 6, pp. 446–465, 2005. [Online]. Available: <https://doi.org/10.1109/TSE.2005.71>
- [154] I. Omoronyia, J. D. Ferguson, M. Roper, and M. Wood, “Using developer activity data to enhance awareness during collaborative software development,” *Comput. Support. Cooperative Work.*, vol. 18, no. 5-6, pp. 509–558, 2009. [Online]. Available: <https://doi.org/10.1007/s10606-009-9104-0>

- [155] M. Handel and J. D. Herbsleb, “What is chat doing in the workplace?” in *CSCW 2002, Proceeding on the ACM 2002 Conference on Computer Supported Cooperative Work, New Orleans, Louisiana, USA, November 16-20, 2002*, E. F. Churchill, J. F. McCarthy, C. Neuwirth, and T. Rodden, Eds. ACM, 2002, pp. 1–10. [Online]. Available: <https://doi.org/10.1145/587078.587080>
- [156] G. Fitzpatrick, P. Marshall, and A. Phillips, “CVS integration with notification and chat: lightweight software team collaboration,” in *Proceedings of the 2006 ACM Conference on Computer Supported Cooperative Work, CSCW 2006, Banff, Alberta, Canada, November 4-8, 2006*, P. J. Hinds and D. Martin, Eds. ACM, 2006, pp. 49–58. [Online]. Available: <https://doi.org/10.1145/1180875.1180884>
- [157] A. Guzzi, A. Bacchelli, M. Lanza, M. Pinzger, and A. van Deursen, “Communication in open source software development mailing lists,” in *Proceedings of the 10th Working Conference on Mining Software Repositories, MSR '13, San Francisco, CA, USA, May 18-19, 2013*, T. Zimmermann, M. D. Penta, and S. Kim, Eds. IEEE Computer Society, 2013, pp. 277–286. [Online]. Available: <https://doi.org/10.1109/MSR.2013.6624039>
- [158] M. Korkala and F. Maurer, “Waste identification as the means for improving communication in globally distributed agile software development,” *J. Syst. Softw.*, vol. 95, pp. 122–140, 2014. [Online]. Available: <https://doi.org/10.1016/j.jss.2014.03.080>
- [159] A. Flostrand, “Finding the future: Crowdsourcing versus the delphi technique,” *Business Horizons*, vol. 60, no. 2, p. 229236, Mar 2017. [Online]. Available: <http://dx.doi.org/10.1016/j.bushor.2016.11.007>
- [160] J. Kaivo-oja, T. Santonen, and Y. Myllylä, “The crowdsourcing delphi: combining the delphi methodology and crowdsourcing techniques,” in *ISPIM Conference Proceedings*. The International Society for Professional Innovation Management (ISPIM), 2013, p. 1.
- [161] A. Alkhatib, M. S. Bernstein, and M. Levi, “Examining crowd work and gig work through the historical lens of piecework,” in *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems, Denver, CO, USA, May 06-11, 2017*, G. Mark, S. R. Fussell, C. Lampe, m. c. schraefel, J. P. Hourcade, C. Appert, and D. Wigdor, Eds. ACM, 2017, pp. 4599–4616. [Online]. Available: <https://doi.org/10.1145/3025453.3025974>
- [162] T. Hoßfeld, M. Hirth, and P. Tran-Gia, “Modeling of crowdsourcing platforms and granularity of work organization in future internet,” in *Proceedings of the 23rd Inter-*

- national Teletraffic Congress*, ser. ITC '11. International Teletraffic Congress, 2011, p. 142–149.
- [163] A. P. Kulkarni, M. Can, and B. Hartmann, “Turkomatic: automatic recursive task and workflow design for mechanical turk,” in *Proceedings of the International Conference on Human Factors in Computing Systems, CHI 2011, Extended Abstracts Volume, Vancouver, BC, Canada, May 7-12, 2011*, D. S. Tan, S. Amershi, B. Begole, W. A. Kellogg, and M. Tungare, Eds. ACM, 2011, pp. 2053–2058. [Online]. Available: <https://doi.org/10.1145/1979742.1979865>
- [164] A. Lamkanfi, S. Demeyer, Q. D. Soetens, and T. Verdonck, “Comparing mining algorithms for predicting the severity of a reported bug,” in *15th European Conference on Software Maintenance and Reengineering, CSMR 2011, 1-4 March 2011, Oldenburg, Germany*, T. Mens, Y. Kanellopoulos, and A. Winter, Eds. IEEE Computer Society, 2011, pp. 249–258. [Online]. Available: <https://doi.org/10.1109/CSMR.2011.31>
- [165] C. Li and M. Shan, “Team formation for generalized tasks in expertise social networks,” in *Proceedings of the 2010 IEEE Second International Conference on Social Computing, SocialCom / IEEE International Conference on Privacy, Security, Risk and Trust, PASSAT 2010, Minneapolis, Minnesota, USA, August 20-22, 2010*, A. K. Elmagarmid and D. Agrawal, Eds. IEEE Computer Society, 2010, pp. 9–16. [Online]. Available: <https://doi.org/10.1109/SocialCom.2010.12>
- [166] D. Gao, Y. Tong, J. She, T. Song, L. Chen, and K. Xu, “Top-k team recommendation and its variants in spatial crowdsourcing,” *Data Science and Engineering*, vol. 2, no. 2, p. 136150, Mar 2017. [Online]. Available: <http://dx.doi.org/10.1007/s41019-017-0037-1>
- [167] A. Anagnostopoulos, L. Becchetti, C. Castillo, A. Gionis, and S. Leonardi, “Online team formation in social networks,” in *Proceedings of the 21st World Wide Web Conference 2012, WWW 2012, Lyon, France, April 16-20, 2012*, A. Mille, F. Gandon, J. Misselis, M. Rabinovich, and S. Staab, Eds. ACM, 2012, pp. 839–848. [Online]. Available: <https://doi.org/10.1145/2187836.2187950>
- [168] C. H. Park, K. Son, J. H. Lee, and S. Bae, “Crowd vs. crowd: large-scale cooperative design through open team competition,” in *Computer Supported Cooperative Work, CSCW 2013, San Antonio, TX, USA, February 23-27, 2013*, A. S. Bruckman, S. Counts, C. Lampe, and L. G. Terveen, Eds. ACM, 2013, pp. 1275–1284. [Online]. Available: <https://doi.org/10.1145/2441776.2441920>

- [169] M. A. Valentine, D. Retelny, A. To, N. Rahmati, T. Doshi, and M. S. Bernstein, "Flash organizations: Crowdsourcing complex work by structuring crowds as organizations," in *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems, Denver, CO, USA, May 06-11, 2017*, G. Mark, S. R. Fussell, C. Lampe, m. c. schraefel, J. P. Hourcade, C. Appert, and D. Wigdor, Eds. ACM, 2017, pp. 3523–3537. [Online]. Available: <https://doi.org/10.1145/3025453.3025811>
- [170] J. Huang, X. Sun, Y. Zhou, and H. Sun, "A team formation model with personnel work hours and project workload quantified," *Comput. J.*, vol. 60, no. 9, pp. 1382–1394, 2017. [Online]. Available: <https://doi.org/10.1093/comjnl/bxx009>
- [171] W. Wang, J. Jiang, B. An, Y. Jiang, and B. Chen, "Toward efficient team formation for crowdsourcing in noncooperative social networks," *IEEE Trans. Cybern.*, vol. 47, no. 12, pp. 4208–4222, 2017. [Online]. Available: <https://doi.org/10.1109/TCYB.2016.2602498>
- [172] M. F. Bosu and S. G. Macdonell, "Experience: Quality benchmarking of datasets used in software effort estimation," *Journal of Data and Information Quality*, vol. 11, no. 4, 2019.
- [173] M. Arora, S. Verma, Kavita, and S. Chopra, "A systematic literature review of machine learning estimation approaches in scrum projects," *Advances in Intelligent Systems and Computing*, p. 573586, 2020. [Online]. Available: [http://dx.doi.org/10.1007/978-981-15-1451-7\\_59](http://dx.doi.org/10.1007/978-981-15-1451-7_59)
- [174] A. Idri, M. Hosni, and A. Abran, "Systematic literature review of ensemble effort estimation," *J. Syst. Softw.*, vol. 118, pp. 151–175, 2016. [Online]. Available: <https://doi.org/10.1016/j.jss.2016.05.016>
- [175] P. Sharma and J. Singh, "Systematic literature review on software effort estimation using machine learning approaches," *2017 International Conference on Next Generation Computing and Information Systems (ICNGCIS)*, Dec 2017. [Online]. Available: <http://dx.doi.org/10.1109/icngcis.2017.33>
- [176] "Software Project Benchmarking - Home Page - ISBSG." [Online]. Available: <https://www.isbsg.org/>
- [177] S. K. T. Ziauddin and S. Zia, "An effort estimation model for agile software development," *Advances in computer science and its applications (ACSA)*, vol. 2, no. 1, pp. 314–324, 2012.
- [178] S. Porru, A. Murgia, S. Demeyer, M. Marchesi, and R. Tonelli, "Estimating story points from issue reports," in *Proceedings of the The 12th International Conference*

- on Predictive Models and Data Analytics in Software Engineering, PROMISE 2016, Ciudad Real, Spain, September 9, 2016.* ACM, 2016, pp. 2:1–2:10. [Online]. Available: <https://doi.org/10.1145/2972958.2972959>
- [179] J. Sayyad Shirabad and T. Menzies, “The PROMISE Repository of Software Engineering Databases.” School of Information Technology and Engineering, University of Ottawa, Canada, 2005. [Online]. Available: <http://promise.site.uottawa.ca/SERepository>
- [180] M. Choetkiertikul, H. K. Dam, T. Tran, T. Pham, A. Ghose, and T. Menzies, “A deep learning model for estimating story points,” *IEEE Trans. Software Eng.*, vol. 45, no. 7, pp. 637–656, 2019. [Online]. Available: <https://doi.org/10.1109/TSE.2018.2792473>
- [181] A. Zakrani, A. Idri, and M. Hain, “Software effort estimation using an optimal trees ensemble: An empirical comparative study,” *Proceedings of the 8th International Conference on Sciences of Electronics, Technologies of Information and Telecommunications (SETIT18), Vol.1*, p. 7282, Jul 2019. [Online]. Available: [http://dx.doi.org/10.1007/978-3-030-21005-2\\_7](http://dx.doi.org/10.1007/978-3-030-21005-2_7)
- [182] T. Mukhopadhyay, S. S. Vicinanza, and M. J. Prietula, “Examining the feasibility of a case-based reasoning model for software effort estimation,” *MIS Quarterly*, vol. 16, no. 2, p. 155, Jun 1992. [Online]. Available: <http://dx.doi.org/10.2307/249573>
- [183] M. J. Shepperd and C. Schofield, “Estimating software project effort using analogies,” *IEEE Trans. Software Eng.*, vol. 23, no. 11, pp. 736–743, 1997. [Online]. Available: <https://doi.org/10.1109/32.637387>
- [184] G. Boetticher, “Using machine learning to predict project effort: Empirical case studies in data-starved domains,” in *Model Based Requirements Workshop*, 2001, pp. 17–24.
- [185] F. González-Ladrón-de-Guevara, M. Fernández-Diego, and C. Lokan, “The usage of ISBSG data fields in software effort estimation: A systematic mapping study,” *J. Syst. Softw.*, vol. 113, pp. 188–215, 2016. [Online]. Available: <https://doi.org/10.1016/j.jss.2015.11.040>
- [186] M. J. Shepperd, Q. Song, Z. Sun, and C. Mair, “Data quality: Some comments on the NASA software defect datasets,” *IEEE Trans. Software Eng.*, vol. 39, no. 9, pp. 1208–1215, 2013. [Online]. Available: <https://doi.org/10.1109/TSE.2013.11>
- [187] A. K. Bardsiri, S. M. Hashemi, and M. Razzazi, “Statistical analysis of the most popular software service effort estimation datasets,” *Journal of Telecommunication, Electronic and Computer Engineering (JTEC)*, vol. 7, no. 1, pp. 87–96, 2015.

- [188] B. A. Kitchenham and E. Mendes, "Why comparative effort prediction studies may be invalid," in *Proceedings of the 5th International Workshop on Predictive Models in Software Engineering, PROMISE 2009, Vancouver, BC, Canada, May 18-19, 2009*, T. J. Ostrand, Ed. ACM, 2009, p. 4. [Online]. Available: <https://doi.org/10.1145/1540438.1540444>
- [189] P. Phannachitta, J. Keung, K. E. Bennin, A. Monden, and K. Matsumoto, "Filter-inc: Handling effort-inconsistency in software effort estimation datasets," *2016 23rd Asia-Pacific Software Engineering Conference (APSEC)*, 2016. [Online]. Available: <http://dx.doi.org/10.1109/apsec.2016.035>
- [190] E. Kocaguneli, T. Menzies, A. Bener, and J. W. Keung, "Exploiting the essential assumptions of analogy-based effort estimation," *IEEE Transactions on Software Engineering*, vol. 38, no. 2, p. 425438, Mar 2012. [Online]. Available: <http://dx.doi.org/10.1109/tse.2011.27>
- [191] T. K. Le-Do, K.-A. Yoon, Y.-S. Seo, and D.-H. Bae, "Filtering of inconsistent software project data for analogy-based effort estimation," *2010 IEEE 34th Annual Computer Software and Applications Conference*, Jul 2010. [Online]. Available: <http://dx.doi.org/10.1109/compsac.2010.56>
- [192] E. Kocaguneli, G. Gay, T. Menzies, Y. Yang, and J. W. Keung, "When to use data from other projects for effort estimation," in *ASE 2010, 25th IEEE/ACM International Conference on Automated Software Engineering, Antwerp, Belgium, September 20-24, 2010*, C. Pecheur, J. Andrews, and E. D. Nitto, Eds. ACM, 2010, pp. 321–324. [Online]. Available: <https://doi.org/10.1145/1858996.1859061>
- [193] M. F. Bosu and S. G. MacDonell, "A taxonomy of data quality challenges in empirical software engineering," in *22nd Australian Conference on Software Engineering (ASWEC 2013), 4-7 June 2013, Melbourne, Victoria, Australia*. IEEE Computer Society, 2013, pp. 97–106. [Online]. Available: <https://doi.org/10.1109/ASWEC.2013.21>
- [194] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.
- [195] F. N. David and J. W. Tukey, "Exploratory data analysis," *Biometrics*, vol. 33, no. 4, p. 768, Dec 1977. [Online]. Available: <http://dx.doi.org/10.2307/2529486>
- [196] J. P. Kincaid, R. Braby, and J. E. Mears, "Electronic authoring and delivery of technical information," *Journal of Instructional Development*, vol. 11, no. 2, p. 813, Jun 1988. [Online]. Available: <http://dx.doi.org/10.1007/bf02904998>

- [197] “Spell and grammar checker.” [Online]. Available: <https://languagetool.org/>
- [198] T. Menzies and M. J. Shepperd, “Special issue on repeatable results in software engineering prediction,” *Empir. Softw. Eng.*, vol. 17, no. 1-2, pp. 1–17, 2012. [Online]. Available: <https://doi.org/10.1007/s10664-011-9193-5>
- [199] G. K. Rajbahadur, S. Wang, Y. Kamei, and A. E. Hassan, “Impact of discretization noise of the dependent variable on machine learning classifiers in software engineering,” *IEEE Trans. Software Eng.*, vol. 47, no. 7, pp. 1414–1430, 2021. [Online]. Available: <https://doi.org/10.1109/TSE.2019.2924371>
- [200] V. Ionescu, “An approach to software development effort estimation using machine learning,” in *13th IEEE International Conference on Intelligent Computer Communication and Processing, ICCP 2017, Cluj-Napoca, Romania, September 7-9, 2017*. IEEE, 2017, pp. 197–203. [Online]. Available: <https://doi.org/10.1109/ICCP.2017.8117004>
- [201] “Open sourcing bert: State-of-the-art pre-training for natural language processing,” Nov 2018. [Online]. Available: <https://ai.googleblog.com/2018/11/open-sourcing-bert-state-of-art-pre.html>
- [202] J. R. Firth, “A synopsis of linguistic theory, 1930-1955,” *Studies in Linguistic Analysis (special volume of the Philological Society)*, vol. 1952-59, pp. 1–32, 1957.
- [203] Z. S. Harris, “Distributional structure,” *WORD*, vol. 10, no. 2-3, p. 146162, Aug 1954. [Online]. Available: <http://dx.doi.org/10.1080/00437956.1954.11659520>
- [204] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, “Attention is all you need,” in *Advances in neural information processing systems*, 2017, pp. 5998–6008.
- [205] S. P. Sotiroudis, S. K. Goudos, and K. Siakavara, “Neural networks and random forests: A comparison regarding prediction of propagation path loss for nb-iot networks,” in *8th International Conference on Modern Circuits and Systems Technologies, MOCAS 2019, Thessaloniki, Greece, May 13-15, 2019*. IEEE, 2019, pp. 1–4. [Online]. Available: <https://doi.org/10.1109/MOCAS.2019.8741751>
- [206] P. Abrahamsson, I. Fronza, R. Moser, J. Vlasenko, and W. Pedrycz, “Predicting development effort from user stories,” in *Proceedings of the 5th International Symposium on Empirical Software Engineering and Measurement, ESEM 2011, Banff, AB, Canada, September 22-23, 2011*. IEEE Computer Society, 2011, pp. 400–403. [Online]. Available: <https://doi.org/10.1109/ESEM.2011.58>

- [207] P. Ardimento and C. Mele, “Using bert to predict bug-fixing time,” *2020 IEEE Conference on Evolving and Adaptive Intelligent Systems (EAIS)*, May 2020. [Online]. Available: <http://dx.doi.org/10.1109/eais48028.2020.9122781>
- [208] E. M. Fávero, D. Casanova, and A. R. Pimentel, “Se3m: A model for software effort estimation using pre-trained embedding models,” *arXiv preprint arXiv:2006.16831*, 2020.
- [209] “Transformers.” [Online]. Available: <https://huggingface.co/transformers/index.html>
- [210] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, “Scikit-learn: Machine learning in Python,” *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.
- [211] C. McCormick, 2021. [Online]. Available: <https://www.chrismccormick.ai/the-bert-collection>
- [212] T. Fawcett, “An introduction to ROC analysis,” *Pattern Recognit. Lett.*, vol. 27, no. 8, pp. 861–874, 2006. [Online]. Available: <https://doi.org/10.1016/j.patrec.2005.10.010>
- [213] Y. Sasaki *et al.*, “The truth of the f-measure. 2007,” 2007.
- [214] M. Stone, “Cross-validatory choice and assessment of statistical predictions,” *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 36, no. 2, p. 111133, Jan 1974. [Online]. Available: <http://dx.doi.org/10.1111/j.2517-6161.1974.tb00994.x>
- [215] J. Grenning, “Agile 2008 - wisdom of crowds keynote and planning poker,” <https://blog.wingman-sw.com/archives/20>, August 2008.
- [216] J. Surowiecki, *The Wisdom of Crowds: Why the Many Are Smarter Than the Few*. Abacus, March 2005.
- [217] S. K. M. Yi, M. Steyvers, M. D. Lee, and M. J. Dry, “The wisdom of the crowd in combinatorial problems,” *Cogn. Sci.*, vol. 36, no. 3, pp. 452–470, 2012. [Online]. Available: <https://doi.org/10.1111/j.1551-6709.2011.01223.x>
- [218] “Manifesto for Agile Software Development.” [Online]. Available: <https://agilemanifesto.org/>
- [219] M. S. Bernstein, J. Brandt, R. C. Miller, and D. R. Karger, “Crowds in two seconds: enabling realtime crowd-powered interfaces,” in *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology, Santa Barbara, CA, USA, October 16-19, 2011*, J. S. Pierce, M. Agrawala, and S. R. Klemmer, Eds. ACM, 2011, pp. 33–42. [Online]. Available: <https://doi.org/10.1145/2047196.2047201>



- [220] F. Qi, X. Jing, X. Zhu, X. Xie, B. Xu, and S. Ying, "Software effort estimation based on open source projects: Case study of Github," *Inf. Softw. Technol.*, vol. 92, pp. 145–157, 2017. [Online]. Available: <https://doi.org/10.1016/j.infsof.2017.07.015>
- [221] J. L. Fleiss, "Measuring nominal scale agreement among many raters," *Psychological Bulletin*, vol. 76, no. 5, p. 378382, 1971. [Online]. Available: <http://dx.doi.org/10.1037/h0031619>
- [222] J. R. Landis and G. G. Koch, "The measurement of observer agreement for categorical data," *Biometrics*, vol. 33, no. 1, p. 159, Mar 1977. [Online]. Available: <http://dx.doi.org/10.2307/2529310>
- [223] B. J. McInnis, D. Cosley, C. Nam, and G. Leshed, "Taking a HIT: designing around rejection, mistrust, risk, and workers' experiences in amazon mechanical turk," in *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems, San Jose, CA, USA, May 7-12, 2016*, J. Kaye, A. Druin, C. Lampe, D. Morris, and J. P. Hourcade, Eds. ACM, 2016, pp. 2271–2282. [Online]. Available: <https://doi.org/10.1145/2858036.2858539>
- [224] D. Hovy, T. Berg-Kirkpatrick, A. Vaswani, and E. Hovy, "Learning whom to trust with mace," in *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, 2013*, pp. 1120–1130.
- [225] H. Yu, Z. Shen, C. Miao, and B. An, "Challenges and opportunities for trust management in crowdsourcing," in *2012 IEEE/WIC/ACM International Conferences on Intelligent Agent Technology, IAT 2012, Macau, China, December 4-7, 2012*. IEEE Computer Society, 2012, pp. 486–493. [Online]. Available: <https://doi.org/10.1109/WI-IAT.2012.104>
- [226] V. Naroditskiy, N. R. Jennings, P. Van Hentenryck, and M. Cebrian, "Crowdsourcing contest dilemma," *Journal of The Royal Society Interface*, vol. 11, no. 99, p. 20140532, Oct 2014. [Online]. Available: <http://dx.doi.org/10.1098/rsif.2014.0532>
- [227] G. Akerlof, "The market for lemons: Quality uncertainty and the market mechanism," *Essential Readings in Economics*, p. 175188, 1995. [Online]. Available: [http://dx.doi.org/10.1007/978-1-349-24002-9\\_9](http://dx.doi.org/10.1007/978-1-349-24002-9_9)
- [228] R. Snow, B. OConnor, D. Jurafsky, and A. Y. Ng, "Cheap and fast—but is it good?" *Proceedings of the Conference on Empirical Methods in Natural Language Processing - EMNLP 08*, 2008. [Online]. Available: <http://dx.doi.org/10.3115/1613715.1613751>

- [229] J. M. Rzeszutarski and A. Kittur, “Instrumenting the crowd: using implicit behavioral measures to predict task performance,” in *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology, Santa Barbara, CA, USA, October 16-19, 2011*, J. S. Pierce, M. Agrawala, and S. R. Klemmer, Eds. ACM, 2011, pp. 13–22. [Online]. Available: <https://doi.org/10.1145/2047196.2047199>
- [230] M. Kutlu, T. McDonnell, Y. Barkallah, T. Elsayed, and M. Lease, “Crowd vs. expert: What can relevance judgment rationales teach us about assessor disagreement?” in *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval, SIGIR 2018, Ann Arbor, MI, USA, July 08-12, 2018*, K. Collins-Thompson, Q. Mei, B. D. Davison, Y. Liu, and E. Yilmaz, Eds. ACM, 2018, pp. 805–814. [Online]. Available: <https://doi.org/10.1145/3209978.3210033>
- [231] R. Lukyanenko, J. Parsons, and Y. F. Wiersma, “The iq of the crowd: Understanding and improving information quality in structured user-generated content,” *Information Systems Research*, vol. 25, no. 4, p. 669689, Dec 2014. [Online]. Available: <http://dx.doi.org/10.1287/isre.2014.0537>
- [232] M. Acosta, A. Zaveri, E. Simperl, D. Kontokostas, S. Auer, and J. Lehmann, “Crowdsourcing linked data quality assessment,” in *The Semantic Web - ISWC 2013 - 12th International Semantic Web Conference, Sydney, NSW, Australia, October 21-25, 2013, Proceedings, Part II*, ser. Lecture Notes in Computer Science, H. Alani, L. Kagal, A. Fokoue, P. Groth, C. Biemann, J. X. Parreira, L. Aroyo, N. F. Noy, C. Welty, and K. Janowicz, Eds., vol. 8219. Springer, 2013, pp. 260–276. [Online]. Available: [https://doi.org/10.1007/978-3-642-41338-4\\_17](https://doi.org/10.1007/978-3-642-41338-4_17)
- [233] F. Sidi, P. H. S. Panah, L. S. Affendey, M. A. Jabar, H. Ibrahim, and A. Mustapha, “Data quality: A survey of data quality dimensions,” in *2012 International Conference on Information Retrieval & Knowledge Management, Kuala Lumpur, Malaysia, March 13-15, 2012*, R. Mahmod, R. Abdullah, L. N. Abdullah, T. M. T. Sembok, A. F. Smeaton, F. Crestani, S. Doraisamy, R. A. Kadir, and M. Ismail, Eds. IEEE, 2012, pp. 300–304. [Online]. Available: <https://doi.org/10.1109/InfRKM.2012.6204995>
- [234] F. Daniel, P. Kucherbaev, C. Cappiello, B. Benatallah, and M. Allahbakhsh, “Quality control in crowdsourcing: A survey of quality attributes, assessment techniques, and assurance actions,” *ACM Comput. Surv.*, vol. 51, no. 1, pp. 7:1–7:40, 2018. [Online]. Available: <https://doi.org/10.1145/3148148>
- [235] T. McDonnell, M. Lease, M. Kutlu, and T. Elsayed, “Why is that relevant? collecting annotator rationales for relevance judgments,” in *Fourth AAAI Conference on Human Computation and Crowdsourcing*, 2016.

- [236] A. Dumitrache, O. Inel, L. Aroyo, B. Timmermans, and C. Welty, "Crowdtruth 2.0: quality metrics for crowdsourcing with disagreement," *arXiv preprint arXiv:1808.06080*, 2018.
- [237] S. Dow, A. P. Kulkarni, S. R. Klemmer, and B. Hartmann, "Shepherding the crowd yields better work," in *CSCW '12 Computer Supported Cooperative Work, Seattle, WA, USA, February 11-15, 2012*, S. E. Poltrock, C. Simone, J. Grudin, G. Mark, and J. Riedl, Eds. ACM, 2012, pp. 1013–1022. [Online]. Available: <https://doi.org/10.1145/2145204.2145355>
- [238] R. Drapeau, L. B. Chilton, J. Bragg, and D. S. Weld, "Microtalk: Using argumentation to improve crowdsourcing accuracy," in *Fourth AAAI Conference on Human Computation and Crowdsourcing*, 2016.
- [239] W. A. Mason and D. J. Watts, "Financial incentives and the "performance of crowds"," in *Proceedings of the ACM SIGKDD Workshop on Human Computation, Paris, France, June 28, 2009*, P. N. Bennett, R. Chandrasekar, M. Chickering, P. G. Ipeirotis, E. Law, A. Mityagin, F. J. Provost, and L. von Ahn, Eds. ACM, 2009, pp. 77–85. [Online]. Available: <https://doi.org/10.1145/1600150.1600175>
- [240] D. Kahneman and A. Tversky, "Prospect theory: An analysis of decision under risk," *Econometrica*, vol. 47, no. 2, p. 263, Mar 1979. [Online]. Available: <http://dx.doi.org/10.2307/1914185>
- [241] D. Li, L. Qiu, J. Liu, and C. Xiao, "Analysis of behavioral economics in crowdsensing: A loss aversion cooperation model," *Sci. Program.*, vol. 2018, pp. 4 350 183:1–4 350 183:18, 2018. [Online]. Available: <https://doi.org/10.1155/2018/4350183>
- [242] L. Wang, T. Xu, and J. Chen, "Research on decision-making behavior of crowdsourcing task based on loss aversion and incentive level," *Kybernetes*, vol. 49, no. 5, p. 15071528, Aug 2019. [Online]. Available: <http://dx.doi.org/10.1108/k-12-2018-0689>
- [243] L. Walasek and N. Stewart, "Context-dependent sensitivity to losses: Range and skew manipulations." *Journal of Experimental Psychology: Learning, Memory, and Cognition*, vol. 45, no. 6, p. 957968, Jun 2019. [Online]. Available: <http://dx.doi.org/10.1037/xlm0000629>
- [244] P. Hooimeijer and W. Weimer, "Modeling bug report quality," in *22nd IEEE/ACM International Conference on Automated Software Engineering (ASE 2007), November 5-9, 2007, Atlanta, Georgia, USA*, R. E. K. Stirewalt, A. Egyed, and B. Fischer, Eds. ACM, 2007, pp. 34–43. [Online]. Available: <https://doi.org/10.1145/1321631.1321639>

- [245] “CreateHIT - Amazon Mechanical Turk.” [Online]. Available: [https://docs.aws.amazon.com/AWSMechTurk/latest/AWSMturkAPI/ApiReference\\_{\\_}CreateHITOperation.html](https://docs.aws.amazon.com/AWSMechTurk/latest/AWSMturkAPI/ApiReference_{_}CreateHITOperation.html)
- [246] L. Irani, “Amazon mechanical turk,” *The Blackwell Encyclopedia of Sociology*, p. 13, Jun 2017. [Online]. Available: <http://dx.doi.org/10.1002/9781405165518.wbeos0994>
- [247] S. R. Munoz and S. I. Bangdiwala, “Interpretation of kappa and b statistics measures of agreement,” *Journal of Applied Statistics*, vol. 24, no. 1, p. 105112, Feb 1997. [Online]. Available: <http://dx.doi.org/10.1080/02664769723918>
- [248] H. Sharp and H. Robinson, “An ethnographic study of XP practice,” *Empir. Softw. Eng.*, vol. 9, no. 4, pp. 353–375, 2004. [Online]. Available: <https://doi.org/10.1023/B:EMSE.0000039884.79385.54>
- [249] P. H. Carstensen, C. Sørensen, and T. Tuikka, “Let’s talk about bugs!” *Scandinavian Journal of Information Systems*, vol. 7, no. 1, p. 6, 1995.
- [250] H. Unphon and Y. Dittrich, “Software architecture awareness in long-term software product evolution,” *Journal of Systems and Software*, vol. 83, no. 11, p. 22112226, Nov 2010. [Online]. Available: <http://dx.doi.org/10.1016/j.jss.2010.06.043>
- [251] H. Shah, S. Sinha, and M. J. Harrold, “Outsourced, offshored software-testing practice: Vendor-side experiences,” *2011 IEEE Sixth International Conference on Global Software Engineering*, Aug 2011. [Online]. Available: <http://dx.doi.org/10.1109/icgse.2011.32>
- [252] D. B. Martin, J. Rooksby, M. Rouncefield, and I. Sommerville, “‘good’ organisational reasons for ‘bad’ software testing: An ethnographic study of testing in a small software company,” in *29th International Conference on Software Engineering (ICSE 2007)*, Minneapolis, MN, USA, May 20–26, 2007. IEEE Computer Society, 2007, pp. 602–611. [Online]. Available: <https://doi.org/10.1109/ICSE.2007.1>
- [253] C. Passos, D. S. Cruzes, T. Dybå, and M. G. Mendonça, “Challenges of applying ethnography to study software practices,” in *2012 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement, ESEM ’12, Lund, Sweden - September 19 - 20, 2012*, P. Runeson, M. Höst, E. Mendes, A. A. Andrews, and R. Harrison, Eds. ACM, 2012, pp. 9–18. [Online]. Available: <https://doi.org/10.1145/2372251.2372255>
- [254] H. Sharp, Y. Dittrich, and C. R. B. de Souza, “The role of ethnographic studies in empirical software engineering,” *IEEE Trans. Software Eng.*, vol. 42, no. 8, pp. 786–804, 2016. [Online]. Available: <https://doi.org/10.1109/TSE.2016.2519887>

- [255] K. Rönkkö, “Software practice from the inside: Ethnography applied to software engineering,” Ph.D. dissertation, Blekinge Institute of Technology, 2002.
- [256] J. O'Neill and D. Martin, “Relationship-based business process crowdsourcing?” *Lecture Notes in Computer Science*, p. 429446, 2013. [Online]. Available: [http://dx.doi.org/10.1007/978-3-642-40498-6\\_33](http://dx.doi.org/10.1007/978-3-642-40498-6_33)
- [257] J. O'Neill, S. Roy, A. Grasso, and D. B. Martin, “Form digitization in BPO: from outsourcing to crowdsourcing?” in *2013 ACM SIGCHI Conference on Human Factors in Computing Systems, CHI '13, Paris, France, April 27 - May 2, 2013*, W. E. Mackay, S. A. Brewster, and S. Bødker, Eds. ACM, 2013, pp. 197–206. [Online]. Available: <https://doi.org/10.1145/2470654.2470683>
- [258] N. Gupta, “An ethnographic study of crowdwork via amazon mechanical turk in india,” Ph.D. dissertation, University of Nottingham, 2017.
- [259] N. Gupta, D. Martin, B. V. Hanrahan, and J. O'Neill, “Turk-life in india,” *Proceedings of the 18th International Conference on Supporting Group Work*, Nov 2014. [Online]. Available: <http://dx.doi.org/10.1145/2660398.2660403>
- [260] M. L. Gray, S. Suri, S. S. Ali, and D. Kulkarni, “The crowd is a collaborative network,” in *Proceedings of the 19th ACM Conference on Computer-Supported Cooperative Work & Social Computing, CSCW 2016, San Francisco, CA, USA, February 27 - March 2, 2016*, D. Gergle, M. R. Morris, P. Bjørn, and J. A. Konstan, Eds. ACM, 2016, pp. 134–147. [Online]. Available: <https://doi.org/10.1145/2818048.2819942>
- [261] A. Cooper *et al.*, *The inmates are running the asylum: [Why high-tech products drive us crazy and how to restore the sanity]*. Sams Indianapolis, 2004, vol. 2.
- [262] D. Ford, T. Zimmermann, C. Bird, and N. Nagappan, “Characterizing software engineering work with personas based on knowledge worker actions,” in *2017 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement, ESEM 2017, Toronto, ON, Canada, November 9-10, 2017*, A. Bener, B. Turhan, and S. Biffl, Eds. IEEE Computer Society, 2017, pp. 394–403. [Online]. Available: <https://doi.org/10.1109/ESEM.2017.54>
- [263] S. Faily and J. Lyle, “Guidelines for integrating personas into software engineering tools,” *Proceedings of the 5th ACM SIGCHI symposium on Engineering interactive computing systems - EICS 13*, 2013. [Online]. Available: <http://dx.doi.org/10.1145/2494603.2480318>
- [264] M. Aoyama, “Persona-and-scenario based requirements engineering for software embedded in digital consumer products,” *13th IEEE International Conference on*

- Requirements Engineering (RE05)*, 2005. [Online]. Available: <http://dx.doi.org/10.1109/re.2005.50>
- [265] M. S. Bernstein, G. Little, R. C. Miller, B. Hartmann, M. S. Ackerman, D. R. Karger, D. Crowell, and K. Panovich, "Soylent: a word processor with a crowd inside," in *Proceedings of the 23rd Annual ACM Symposium on User Interface Software and Technology, New York, NY, USA, October 3-6, 2010*, K. Perlin, M. Czerwinski, and R. Miller, Eds. ACM, 2010, pp. 313–322. [Online]. Available: <https://doi.org/10.1145/1866029.1866078>
- [266] O. Ayalon and E. Toch, "Crowdsourcing privacy design critique: An empirical evaluation of framing effects," *Proceedings of the Annual Hawaii International Conference on System Sciences*, 2018. [Online]. Available: <http://dx.doi.org/10.24251/hicss.2018.598>
- [267] D. Stergiadis, "Persona modeling by crowdsourcing using the repertory grid technique," 2017.
- [268] G. Kazai and I. Zitouni, "Quality management in crowdsourcing using gold judges behavior," in *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining, San Francisco, CA, USA, February 22-25, 2016*, P. N. Bennett, V. Josifovski, J. Neville, and F. Radlinski, Eds. ACM, 2016, pp. 267–276. [Online]. Available: <https://doi.org/10.1145/2835776.2835835>
- [269] W. Reinhardt, B. Schmidt, P. Sloep, and H. Drachsler, "Knowledge worker roles and actions-results of two empirical studies," *Knowledge and Process Management*, vol. 18, no. 3, p. 150174, Jul 2011. [Online]. Available: <http://dx.doi.org/10.1002/kpm.378>
- [270] "Process Mining and Process Analysis - Fluxicon." [Online]. Available: <https://fluxicon.com/>
- [271] R. C. Martin, *Clean Code - a Handbook of Agile Software Craftsmanship*. Prentice Hall, 2009. [Online]. Available: [http://vig.pearsoned.com/store/product/1,1207,store-12521\\_isbn-0132350882,00.html](http://vig.pearsoned.com/store/product/1,1207,store-12521_isbn-0132350882,00.html)
- [272] D. E. Knuth, *The art of computer programming, Volume I: Fundamental Algorithms, 3rd Edition*. Addison-Wesley, 1997. [Online]. Available: <https://www.worldcat.org/oclc/312910844>
- [273] A. Hunt and D. Thomas, *The Pragmatic Programmer : From Journeyman to Master*. Addison-Wesley Professional, 1999.

- [274] G. Mark, S. R. Fussell, C. Lampe, m. c. schraefel, J. P. Hourcade, C. Appert, and D. Wigdor, Eds., *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems, Denver, CO, USA, May 06-11, 2017*. ACM, 2017. [Online]. Available: <https://doi.org/10.1145/3025453>
- [275] A. S. Bruckman, S. Counts, C. Lampe, and L. G. Terveen, Eds., *Computer Supported Cooperative Work, CSCW 2013, San Antonio, TX, USA, February 23-27, 2013*. ACM, 2013. [Online]. Available: <http://dl.acm.org/citation.cfm?id=2441776>
- [276] J. S. Pierce, M. Agrawala, and S. R. Klemmer, Eds., *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology, Santa Barbara, CA, USA, October 16-19, 2011*. ACM, 2011. [Online]. Available: <https://doi.org/10.1145/2047196>
- [277] K. Perlin, M. Czerwinski, and R. Miller, Eds., *Proceedings of the 23rd Annual ACM Symposium on User Interface Software and Technology, New York, NY, USA, October 3-6, 2010*. ACM, 2010. [Online]. Available: <https://doi.org/10.1145/1866029>