Predictiveness and Effectiveness of Story Points in Agile Software Development

by

Vali Tawosi

A dissertation submitted in fulfilment
of the requirements for the degree of
Doctor of Philosophy
of
University College London

University College London
Department of Computer Science

March 2023

Declaration

I, Vali Tawosi, confirm that the work presented in this thesis is my own. Where information has been derived from other sources, I confirm that this has been indicated in the thesis. The work presented in this thesis is original work undertaken between January 2019 and March 2023 at University College London.

Parts of this document have been published in peer-reviewed venues. I list these papers in Section 1.3. They represent Chapters 3, 4, 5, and 6, respectively.

Date: 2nd March 2023
Name: Vali Tawosi
Signature:

Abstract

Agile Software Development (ASD) is one of the most popular iterative software development methodologies, which takes a different approach from the conventional sequential methods. Agile methods promise a faster response to unanticipated changes during development, typically contrasted with traditional project development, which assumes that software is specifiable and predictable.

Traditionally, practitioners and researchers have utilised different Functional Size Measures (FSMs) as the main cost driver to estimate the effort required to develop a project (Software Effort Estimation – SSE). However, FSM methods are not easy to use with ASD. Thus, another measure, namely Story Point (SP), has become popular in this context. SP is a relative unit representing an intuitive mixture of complexity and the required effort of a user requirement.

Although recent surveys report on a growing trend toward intelligent effort estimation techniques for ASD, the adoption of these techniques is still limited in practice. Several factors limit the accuracy and adaptability of these techniques. The primary factor is the lack of enough noise-free information at the estimation time, restricting the model's accuracy and reliability.

This thesis concentrates on SEE for ASD from both the technique and data perspectives. Under this umbrella, I first evaluate two prominent state-of-the-art works for SP estimation to understand their strengths and weaknesses. I then introduce and evaluate a novel method for SP estimation based on text clustering. Next, I investigate the relationship between SP and development time by conducting a thorough empirical study. Finally, I explore the effectiveness of SP estimation methods when used to estimate the actual time. To carry out this research, I have curated the TAWOS (**T**awosi **A**gile **W**eb-based **O**pen-**S**ource) dataset, which consists of over half a million issues from Agile, open-source projects. TAWOS has been made publicly available to allow for the reproduction and extension in future work.

Keywords: *Software Effort Estimation*, *Agile Software Development*, *Story Points Estimation*

Impact Statement

The production of software is an elaborate engineering process no different from other engineering disciplines. Yet, the prediction of the effort required to develop a software project has been challenging for this industry since its inception in the 1950s. This challenge had grown bigger in the last two decades when the agile methodologies started estimating effort for individual software tasks rather than the whole project.

Story point is the most popular unit of effort agile software development teams use. Consequently, researchers have built several estimation models for story point estimation using Artificial Intelligence (AI).

In this thesis, I focus on the predictability of story points by these approaches and the efficacy of story points as an effort measure. The results provide the research community with a new understanding of the limitations of the current state-of-the-art. These approaches, despite using advanced deep-learning models, fail to outperform simple baseline techniques, which do not use any information from the task itself to make an estimation. This finding may shift the focus in future work towards exploring new effort drivers instead of the utilisation of more powerful algorithms. Moreover, via an extensive empirical study, we show that the human-estimated story points are biased, rendering them unsuitable as ground truth for training AI-based models. We suggest that practitioners in the industry and researchers in academia use actual development time as a ground truth when available.

Furthermore, this thesis provides the software engineering research community with a holistic and versatile dataset of around half a million software tasks (a.k.a. issues) collected from several open-source projects, making it well-suited to several research avenues and cross-analyses therein. This dataset can foster research in this area by providing the researchers with easier means to reproduce and extend future work.

Acknowledgements

This document, the culmination of a challenging but rewarding journey called PhD, could not have been possible without the generous support of several exceptional individuals to whom I would like to express my sincerest gratitude.

First and foremost, I would like to express my most profound appreciation to my first supervisor, Professor Federica Sarro, for all the tremendous support, invaluable guidance, and mentorship she provided me with all along this journey. I am grateful for her trust in me and her patience with me during times of uncertainty.

Secondly, I would like to express my gratitude to my second supervisor, Professor Mark Harman, for his continuous support, encouragement, and sage advice. Both my supervisors were and will continue to be my greatest source of inspiration.

I also extend my special thanks to Dr Afnan AlSubaihin for the generous help and guidance she provided me throughout my PhD. I extend my gratitude to Rebecca Moussa, my friend and lab mate, for her incredible help and contributions, and to my other co-author, Dr Alessio Petrozziello, for his valuable contributions.

I would also like to thank the amazing people in CREST and SSE who welcomed me to the lab and were always there when I needed help: Professor William Langdon, Dr Giovani Guizzo, Dr Jie Zhang, Dr Max Hort, Dario Asprone, Dr Profir-Petru Partachi, Dr Aymeric Bolt, Dr Maria Kechagia, Dr David Kelly, and everyone else who made my PhD journey more pleasant.

I am incredibly grateful to my parents and siblings, who sacrificed whatever they could to see me succeed.

Last but certainly not least, a big thanks to the love of my life Dr Fatima Najibi, whose love gave me strength and her support throughout all the ups and downs of my PhD, without ever losing her faith in me, made this thesis possible.

Thank you all.

UCL Research Paper Declaration Form:

REFERENCING THE DOCTORAL CANDIDATE'S OWN PUBLISHED WORK(S)

1. **For a research manuscript that has already been published:**

   (a) **What is the title of the manuscript?**
   A Versatile Dataset of Agile Open Source Software Projects

   (b) **Please include a link to or doi for the work:**
   https://doi.org/10.1145/3524842.3528029

   (c) **Where was the work published?**
   Proceedings of the 19th International Conference on Mining Software Repositories (MSR)

   (d) **Who published the work?**
   ACM

   (e) **When was the work published?**
   17 October 2022

   (f) **List the manuscript's authors in the order they appear on the publication:**
   Vali Tawosi, Afnan Al-Subaihin, Rebecca Moussa, Federica Sarro

   (g) **Was the work peer-reviewed?**
   Yes

   (h) **Have you retained the copyright?**
   Yes

   (i) **Was an earlier form of the manuscript uploaded to a preprint server (e.g. medRxiv)? If 'Yes', please give a link or doi**
   https://arxiv.org/abs/2202.00979

2. **For multi-authored work, please give a statement of contribution covering all authors:**

Vali Tawosi did the literature review and data gathering, implemented the software to pull the raw data and extract computed features, curated the dataset, and wrote the manuscript. Afnan Al-Subaihin contributed to the writing by reviewing the paper and improving it. Rebecca Moussa helped review the paper and improve the writing. And Federica Sarro supervised the project, reviewed the paper, and improved the writing.

3. **In which chapter(s) of your thesis can this material be found?**
   Chapter 3

e-Signatures confirming that the information above is accurate:

Candidate: Vali Tawosi
Date: 6 March 2023

Supervisor signature: Federica Sarro
Date: 6 March 2023

UCL Research Paper Declaration Form:

REFERENCING THE DOCTORAL CANDIDATE'S OWN PUBLISHED WORK(S)

1. **For a research manuscript that has already been published:**

   (a) **What is the title of the manuscript?**
   Agile Effort Estimation: Have We Solved the Problem Yet? Insights From A Replication Study

   (b) **Please include a link to or doi for the work:**
   https://doi.org/10.1109/TSE.2022.3228739

   (c) **Where was the work published?**
   IEEE Transactions on Software Engineering (TSE)

   (d) **Who published the work?**
   IEEE

   (e) **When was the work published?**
   14 December 2022

   (f) **List the manuscript's authors in the order they appear on the publication:**
   Vali Tawosi, Rebecca Moussa, Federica Sarro

   (g) **Was the work peer-reviewed?**
   Yes

   (h) **Have you retained the copyright?**
   Yes

   (i) **Was an earlier form of the manuscript uploaded to a preprint server (e.g. medRxiv)? If 'Yes', please give a link or doi**
   https://arxiv.org/abs/2201.05401

2. **For multi-authored work, please give a statement of contribution covering all authors:**

Vali Tawosi did the literature review, implemented and performed the experiments, analysed the results, and wrote the manuscript. Rebecca Moussa helped review the paper and improve the writing, and Federica Sarro supervised the project, helped design the experiments, reviewed the paper, and improved the writing.

3. **In which chapter(s) of your thesis can this material be found?**
   Chapter 4

e-Signatures confirming that the information above is accurate:

Candidate: Vali Tawosi
Date: 6 March 2023

Supervisor signature: Federica Sarro
Date: 6 March 2023

UCL Research Paper Declaration Form:

REFERENCING THE DOCTORAL CANDIDATE'S OWN PUBLISHED WORK(S)

1. **For a research manuscript that has already been published:**

   (a) **What is the title of the manuscript?**
   Investigating the Effectiveness of Clustering for Story Point Estimation

   (b) **Please include a link to or doi for the work:**
   https://doi.org/10.1109/SANER53432.2022.00101

   (c) **Where was the work published?**
   Proceedings of 2022 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER)

   (d) **Who published the work?**
   IEEE

   (e) **When was the work published?**
   21 July 2022

   (f) **List the manuscript's authors in the order they appear on the publication:**
   Vali Tawosi, Afnan Al-Subaihin, Federica Sarro

   (g) **Was the work peer-reviewed?**
   Yes

   (h) **Have you retained the copyright?**
   Yes

   (i) **Was an earlier form of the manuscript uploaded to a preprint server (e.g. medRxiv)? If 'Yes', please give a link or doi**
   https://discovery.ucl.ac.uk/id/eprint/10143360/

2. **For multi-authored work, please give a statement of contribution covering all authors:**

Vali Tawosi did the literature review, implemented and performed the experiments, analysed the results, and wrote the manuscript. Afnan Al-Subaihin helped with the implementation of the method, review the paper and improve the writing, and Federica Sarro supervised the project, helped design the experiments, reviewed the paper, and improved the writing.

3. **In which chapter(s) of your thesis can this material be found?**
   Chapter 5

e-Signatures confirming that the information above is accurate:

Candidate: Vali Tawosi
Date: 6 March 2023

Supervisor signature: Federica Sarro
Date: 6 March 2023

UCL Research Paper Declaration Form:

REFERENCING THE DOCTORAL CANDIDATE'S OWN PUBLISHED WORK(S)

1. **For a research manuscript that has already been published:**
   (a) **What is the title of the manuscript?**
   On the Relationship Between Story Points and Development Effort in Agile Open-Source Software
   (b) **Please include a link to or doi for the work:**
   https://doi.org/10.1145/3544902.3546238
   (c) **Where was the work published?**
   Proceedings of the 16th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)
   (d) **Who published the work?**
   ACM
   (e) **When was the work published?**
   19 September 2022
   (f) **List the manuscript's authors in the order they appear on the publication:**
   Vali Tawosi, Rebecca Moussa, Federica Sarro
   (g) **Was the work peer-reviewed?**
   Yes
   (h) **Have you retained the copyright?**
   Yes
   (i) **Was an earlier form of the manuscript uploaded to a preprint server (e.g. medRxiv)? If 'Yes', please give a link or doi**
   https://discovery.ucl.ac.uk/id/eprint/10151116/

2. **For multi-authored work, please give a statement of contribution covering all authors:**

Vali Tawosi did the literature review, implemented and performed the experiments, analysed the results, and wrote the manuscript. Rebecca Moussa helped review the paper and improve the writing, and Federica Sarro supervised the project, helped design the experiments, reviewed the paper, and improved the writing.

3. **In which chapter(s) of your thesis can this material be found?**
   Chapter 6

e-Signatures confirming that the information above is accurate:

Candidate: Vali Tawosi
Date: 6 March 2023

Supervisor signature: Federica Sarro
Date: 6 March 2023

CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1

Introduction

Like any other engineering discipline, it is crucial for managers of software projects to know how much time, resources and workforce they will need to finish the project [1]. However, unfinished, overdue, and over-budgeted software projects have been distressing the software production industry for many years [2]. The primary reason for this management pitfall is usually the miscalculation of the required effort to finish the project in the early stages of the project development life cycle when the project still has many unknowns, and the process possesses considerable uncertainty. Both overestimation and underestimation can cause severe problems for a software production company. The company might lose a bid or misspend its resources with the former. While the latter may cause delay, low-quality product, or even failure, and consequently, unsatisfied customers and financial losses [3]–[5].

Definition 1.1: Software Effort Estimation

The process of estimating the amount of effort needed to develop a software project is called Software Effort Estimation (SEE).

The problem of unknowns and uncertainties is usually exacerbated by the subjective estimation of human experts. Human beings have been shown to be biased thinkers, having limited experience and yet weighting memories that they recall the most, mixing logic with emotion, and being volatile to other individuals' opinion [6], which can make their judgement subjective and inconsistent.

One solution to the biased judgement problem is to learn patterns from factual recorded data from the past and perform automated inference on the current situation, i.e., use machine intelligent SEE techniques. For more than three decades, researchers proposed different approaches to intelligent SEE, from statistical models [7] to Artificial Intelligence (AI) based models [4]. Many of these approaches harvest information from finished projects in the past to predict the effort of the new ones. This information, which is called *cost* or *effort drivers*, usually are related to the software size [8]. The model takes as input the cost drivers for a new software project and maps it to a continuous output value that represents the estimated effort needed to develop the project, usually in terms of person-hour or person-month.

The approach taken to estimate effort for a software project can vary based on the development methodology followed to build it. In traditional (i.e., non-agile) software development methodologies, the estimation is usually done at the project level [9]; in contrast with newer Agile Software Development (ASD) methodologies, the estimation is also focused on a much finer-grained iteration level or even task level. Project-level estimation focuses on estimating the effort required to complete a whole software project. To this end, Functional Size Measures (FSM), such as Function Point (FP) [10], or COSMIC Function Point (CFP) [11], have been usually used as a cost driver [1], [12]–[15]. However, task-level estimation is interested in estimating the effort needed to complete a single task which can be the development of a new feature, a requested change, or a bug repair [1]. In these cases, FSM methods are not easy to use [16]; thus, another measure, namely **Story Point**, has become popular [17], [18].

Definition 1.2: Story Point

Story Point (SP) is a relative unit that represents an intuitive mixture of complexity and required effort of a *user story*.[a]

[a] In the ASD context, a user story is a user-valued functionality which is specified in the form of one or a few sentences in the everyday language of the user.

1.1. PROBLEM STATEMENT

With the increasing popularity of agile methodologies in the software industry, several studies have been conducted to investigate effort estimation in ASD [19]. Most of these studies focus on story points as the primary metric for the effort needed to develop software tasks, and propose intelligent SEE models to estimate it [20]. Such models are envisioned to substitute the expert estimator, or in a consensus-based method like planning poker, to contribute an estimation as another member of the estimation team. Building a tool to accurately estimate the SP value for a task would enable the project manager, as well as the development team, to provision and plan the development process with higher confidence and deliver the software on time. However, this attempt to automate the estimation process might interfere with the agile manifesto which emphasises the value of “individuals and interactions over processes and tools”. Developers might have a difficult time trusting the estimation of a tool over their own, which in turn might lead to a reduction in the quality of interactions and a drop in productivity. To mitigate this potential problem, explainable methods may be effective. An intelligent SEE model that can explain the reasons behind its decision (i.e., estimation) will increase the user’s trust and confidence in the generated estimation [21].

Nevertheless, the adoption of intelligent SEE models is still limited in practice, as their estimation performance is not acceptable. Most of the studies that propose such models train them using story points from the previous (delivered) tasks to estimate story points for the new ones [21]–[26]. However, Usman et al. [18] found (and recently confirmed by Fernández-Diego et al. [19]) that human subjective estimation is the most commonly used approach for story point estimation in ASD. While the accuracy of the subjective estimation techniques is sensitive to the practitioners’ expertise and prior experience, thus, prone to bias. Nevertheless, one of the issues that seem neglected in the literature is the use of a biased human-estimated effort measure (i.e., story points) as the gold standard to train intelligent SEE models.

Therefore, considering the central role that story point plays in agile effort estimation and the possible bias in human expert estimations, this thesis aims at an empirical investigation of the predictiveness and effectiveness of story points in ASD. To this end, two main research questions are addressed: **(question A)** *How effective the state-of-the-art intelligent story point estimation models are in predicting story points*? and **(question B)** *To what extent human expert-estimated story points are a good indicator of effort in ASD*?

To address question A, I evaluate state-of-the-art story point estimation methods via an extensive empirical evaluation using a large dataset of agile open source issues, which I have curated for such a purpose. The design and results of this study are presented in Chapter 4, while the dataset is introduced in Chapter 3.

In Chapter 5 I introduce a novel approach to estimate story points for ASD and compare it to the previous methods.

To answer question B, I investigate the relationship between story points and the ASD task development effort, expressed in terms of time needed to realise it. The design and results of this study are presented in Chapter 6.

And finally, in Chapter 7, I present the idea of using the unbiased and factual development time of ASD tasks, instead of story points, to train effort estimation models.

1.2. CONTRIBUTIONS

Addressing the two main research questions posed in the previous section, this thesis:

1. Investigates the estimation performance of two prominent previous methods introduced for story point estimation (including the state-of-the-art deep learning-based method) via a replication study and reveals that these models are outperformed by naïve baseline techniques for the majority of the cases. This suggests that using complicated methods does not necessarily produce a more accurate model. The replication study led to new findings which, in some cases rebutted the previous findings. This further proves the importance of replication studies and making replication packages publicly available in order to support reproduction, replication and extension of previous work.

2. Proposes a new approach based on topic modelling and clustering for story point estimation, evaluates it using a large dataset of issues, and compares it to previous approaches. The estimation performance of the proposed approach is as good as the state-of-the-art and still not statistically significantly better than the baseline techniques in all cases, which does not justify its additional complexity.

3. Via an extensive empirical study investigates the relationship between story points and the development time of software tasks in an open-source ASD context. The results show that the expert-estimated story point does not strongly correlate with the development time. Moreover, expert estimation is not consistent throughout the project. Specifically, expert estimators are less consistent in estimating user stories with larger than 5 points. These findings suggest that the expert-estimated story point is biased.

4. Evaluates the efficacy of approaches introduced for story point estimation in estimating the development time of the open-source software tasks. The results show that all the investigated methods can estimate the development time with an average absolute error of less than a single working day (i.e., 8.5 hours).

5. Provides the agile development effort estimation research community with a holistic and versatile dataset called the TAWOS (an acronym for

Tawosi **A**gile **W**eb-based **O**pen-**S**ource) dataset that contains a wealth of information on more than half a million issues (i.e., individual software tasks including user stories, bugs, etc.) from 44 agile open-source software. The TAWOS dataset is well-suited to several research avenues and cross-analyses therein, including effort estimation, issue prioritization, issue assignment and many more. This dataset is extensively used in this thesis and made publicly available online for further research and replication purposes, and curated for future expansions.

6. Adopts Open Science practices by making all the scripts, codes, tools, and data used for conducting the studies reported in this thesis publicly available to encourage replication and extension of this work and foster research in the area of software effort estimation.

1.3. LIST OF PAPERS

Chapters 3 to 6 of this thesis are published in peer-reviewed venues. Hence, the reader may notice a mixed use of ‘I’ and ‘we’ throughout this document. I continue to use ‘we’ in the paper chapters to acknowledge my co-authors’ contribution, as the publication was the result of the collaboration with them. I provided the title, list of authors, and venue for each of these papers at the beginning of the respective chapter. Moreover, the papers are presented as published in each Chapter except for the removal of some sections to reduce repetition throughout the thesis and the change in the appearance of some of the figures and tables that required editing to fit into the thesis format.

The following is the list of papers published so far, in the order they appear in the thesis:

1. V. Tawosi, F. Sarro, A. Petrozziello, and M. Harman, “Multi-objective software effort estimation: A replication study,” *IEEE Transactions on Software Engineering (TSE)*, vol. 48, no. 8, pp. 3185–3205, 2021[1]

2. V. Tawosi, A. Al-Subaihin, R. Moussa, and F. Sarro, “A versatile dataset of agile open source software projects,” in *Proceedings of 19th International Conference on Mining Software Repositories (MSR)*, ACM, 2022

3. V. Tawosi, R. Moussa, and F. Sarro, “Agile effort estimation: Have we solved the problem yet? insights from a replication study,” *IEEE Transactions on Software Engineering (TSE)*, pp. 1–19, 2022

[1]This paper reports the result of a replication study performed during the first year of my PhD and referred to in the last paragraph of Section 2.1.

4. V. Tawosi, A. Al-Subaihin, and F. Sarro, "Investigating the effectiveness of clustering for story point estimation," in *Proceedings of the 29th IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER)*, IEEE, 2022, pp. 816–827

5. V. Tawosi, R. Moussa, and F. Sarro, "On the relationship between story point and development effort in agile open-source software," in *16th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*, ACM/IEEE, vol. 16, 2022

1.4. ORGANISATION OF THE THESIS

In the remainder of this thesis, I first present the literature on software effort estimation for both traditional and agile software development methodologies in Chapter 2.

Chapter 3 presents the TAWOS dataset, which is used throughout this thesis to evaluate different approaches to story point estimation.

Chapter 4 presents the replication and extension of a prominent previous study in agile software development effort estimation.

Then, Chapter 5 presents LHC-SE, the text clustering approach to story point estimation introduced in this thesis, and compares it to the previous work.

Chapter 6 investigates the correlation between the expert-estimated story points with development time and the consistency of expert estimates throughout the project. This chapter suggests that intelligent agile effort estimation approaches should use the development time to train whenever it was available.

Therefore, Chapter 7 adapts and evaluates all the approaches investigated in this thesis to train with and estimate the development time.

Finally, Chapter 8 presents the conclusions drawn from the main findings and provides directions for future work.

Chapter 2

Literature Review

This chapter presents a brief history of intelligent software effort estimation, followed by a literature review of the main subjects investigated in the thesis.

Section 2.1 describes how intelligent SEE systems use the information from the past to determine the estimated effort for new projects. This section looks at the problem from the traditional (i.e., non-agile) software development methodologies' point of view. Then, Section 2.2 and 2.3 describe the agile planning and software size measures used in SEE and their evolution, respectively.

The previous work on task-level effort estimation, which used user stories to estimate their story point, is discussed in Section 2.4. Section 2.5 presents studies that used story points as a cost driver to estimate effort. And finally, the studies investigating the relation between story point and other size and effort measures are presented in Section 2.6.

2.1. TRADITIONAL SOFTWARE EFFORT ESTIMATION

Early efforts to build an estimation technique sought to find a set of factors related to the software size and cost and use them to build a formula using regression analysis [7]. With the advent of intelligent techniques in software effort estimation, different approaches have been investigated, including analogy-based techniques (e.g. Categorical Regression [31], Case-Based Reasoning [32]), Machine Learning techniques (e.g. Classification and Regression Trees (CART) [33], Artificial Neural Networks [34], Support Vector Regression [35], Bayesian Networks [36]), Search-Based approaches (e.g. Tabu Search and Genetic Programming) [5], [37], [38], and combinations of two or more of these methods (e.g. [39]–[42]).

These approaches usually exploit the relations between available information on a set of past projects to build a model that can be used to predict the effort for a new project. The information presented in the set (i.e., predictors or cost drivers) is the factors identified to be related to the effort. This information is measured on previous projects and stored in a database. Examples of such cost drivers are the number of files, the number of inputs and outputs, the functional size of the software, the size of the team, and team experience, etc. A prediction model takes as input the predictor values for a new project and returns a scalar value that represents the estimated effort to develop a new software project.

Depending on the prediction approach, the predictors are used in a different way. For example, a linear regression technique combines the predictors through a linear equation with coefficients coming from the statistics of the previous projects, while in Case-Based reasoning, the predictors are exploited to find the most similar projects from the past, then use the effort measured for the similar project(s) from the past to predict the effort for the new project.

One of the most successful approaches to building intelligent effort estimation models for traditional software development is Search-Based Software Engineering (SBSE) [37]. SBSE reformulates software engineering problems as search problems. The SBSE solution space is explored using a search technique equipped with software metrics able to discriminate between good and bad solutions, which will tend to guide the search towards the optimal solution(s) [43].

In Search-Based Effort Estimation (SBEE), the optimization method builds many candidate models and tries to identify the optimal model, i.e., the one providing the most accurate estimates [44]. Such a model can be described by the following equation [5]:

$$EstimatedEffort = w_1 op_1 c_1 + ... + w_n op_n c_n + C \tag{2.1}$$

where c_i represents the value of the i^{th} cost driver, w_i its weight, and C represents a constant, while op_i represents the i^{th} mathematical operator (i.e. $+$, $-$, $\times$, etc.) of the model. Any value except for the cost drivers in Equation 2.1 can be optimized to maximise the model's accuracy. Consequently, the search space consists of all the models that could possibly be built by varying the weights and operators in Equation 2.1. Considering the number of project features and the range of coefficients and operators, the search space can become dramatically large, which makes this problem suitable to be solved by search-based approaches.

The fitness of a model is evaluated by its prediction accuracy. In a single objective SBEE, a single measure of accuracy is used to compare different models and consequently derive the best one [45], [46]. However, in the context of effort estimation, there are several measures of accuracy, each one focusing on a different aspect. Since there is no defined way of aggregating different accuracy measures for SEE, a multi-objective solution is inevitable [47], where a number of competing measures are optimised simultaneously [5], [48].

Sarro et al. [38] proposed a bi-objective solution named Confidence Guided Effort Estimation (CoGEE), which maximises the accuracy and minimises the confidence interval of the estimation error during the model building. CoGEE outperformed three state-of-the-art intelligent effort estimation methods and moved the distribution of the error within claimed thresholds for industrial human-expert-based best practices. Therefore, CoGEE has set the state-of-the-art for multi-objective project-level effort estimation and has been the only one to achieve human-competitive results thus far.[1]

As an initial project for my PhD, I replicated Sarro et al.'s work [38]. In this replication [5], we carried out an empirical study in order to answer the same research questions posed in the original study by means of the same experimental design extended by the use of a recent state-of-the-art baseline benchmark (i.e., LP4EE [49]), four additional variants of multi-objective evolutionary algorithms under the hood of CoGEE, execution time evaluation of the different variants, and a completely new and independent implementation of CoGEE based on a popular Java Evolutionary Computation framework (i.e., `JMetal` [50]), which has decreased the running time by over 99.8% with respect to the original `R` version.

2.2. AGILE PLANNING

In software development, agile methodology emerged with an emphasis on individuals and interactions over processes and tools, working software over comprehensive documentation, customer collaboration over contract negotiation, and responding to change over following a plan [51]. Agile planning, therefore, is influenced by all these four principles.

Agile methodologies use iterative processes to realize incremental delivery of software functionalities. A software project, thus, is divided into multiple releases, which can accelerate the product launch into the market. Each release

[1]Sarro et al. [38] won the ACM SIGEVO 13th Annual (2016) "Humies" Award for Human-Competitive Results Produced by Genetic and Evolutionary Computation `http://www.human-competitive.org.`

is composed of several one to four weeks-long iterations called **Sprints**. Short sprints and frequent releases are favoured because they provide faster feedback and improve a product's return on investment [52].

A survey conducted in 2017 identified Scrum as the most popular agile development method [53]. In Scrum, there are five levels of planning: Portfolio, Product, Release, Sprint, and Daily planning [52]. At each of these planning levels, the team may estimate the size of work or effort needed to complete it. However, the estimation gets more detailed as the planning enters a lower level. For example, in release planning, the team typically estimates the effort required for the high-level features that are intended to be included in the upcoming releases [54]. Estimation at this level is not expected to be as accurate as the sprint level. In the Sprint planning session, which occurs at the beginning of each sprint, the specific product *Backlog* items that the Scrum team will work on in the next sprint are agreed upon and estimated [55].

Definition 2.1: User Story

A **User Story** is a user-valued functionality that is specified in the form of one or a few sentences in the everyday language of the user.

Definition 2.2: Backlog

A **Backlog** is a breakdown of work that contains an ordered list of user stories or tasks that a Scrum team maintains for a project.

Sprint planning starts with determining the available capacity of the team to perform work during the sprint. The capacity can be expressed in the preferred unit of effort estimation, e.g., story points, ideal days, or effort-hours [52], but it must be in the same unit used for effort estimation [54].

2.3. SOFTWARE SIZE MEASURES

Albrecht [56] was the first who introduced a disciplined method for measuring software product size before it was built, called **Function Point Analysis** (FPA), based on the functionality that the software product is built to deliver to the customer. Soon after, he showed that there is a strong correlation between Function Points and the final effort of a software [10]. Although FPA was designed to measure software from the single domain of business applications [57], it is still widely applied in the software production industry [58].

COSMIC[2] **Function Points** (CFP) is the second generation of software functional size measurement developed by an international group of software measurement experts [11]. Unlike FPA, CFP takes non-functional requirements into consideration as well and is suitable for a broader range of application domains including, but not limited to, business applications, mobile applications, real-time software, big-data applications, and service-oriented software [57]. However, in the context of agile software development (particularly Scrum and Kanban methods), the practitioners introduced and used alternative measures as agile-specific software size measurement units [52], [59], [60]. The most popular of these units is called Story Points (SP) [53]. SP is a relative unit that represents an intuitive mixture of complexity and the required effort of a user story [59].

Each user story in the backlog is assigned a story point value. There is no set formula for defining the size of a story. Rather, a story point estimate is an amalgamation of the amount of effort involved in developing the feature, its complexity, and the risk inherent in its development process. One of the most popular methods to estimate backlog items is Planning Poker [53]. Planning Poker is a consensus-based technique. During a planning poker session, the team of developers who are working on the product engage in an intense discussion to expose assumptions, acquire a shared understanding, and size the work items. User stories estimated during the planning poker session are grouped or binned together with respect to their effort size [52]. A common approach to determine the story point value of a user story is to select one of the smallest stories in the Backlog and assign it one story point. Then the more complex and extensive user stories get more points relative to their size. Therefore, the effort required to develop a story that is assigned with five SPs should be almost five times of a story that is assigned with one SP (see Figure 2.1). Unlike FPA and CFP, SP lacks a disciplined method of measurement; therefore, the developers use it as a relative measure to keep the relative difference of stories in size. SP estimation is required to be consistent throughout the project [59].

2.4. AGILE EFFORT ESTIMATION

During the Sprint planning in Scrum, the estimation is focused on a user story and task[3] level [52]. Therefore, the major and most accessible feature at this

[2]Common Software Measurement International Consortium

[3]During the sprint planning, some user stories are decomposed into multiple tasks.

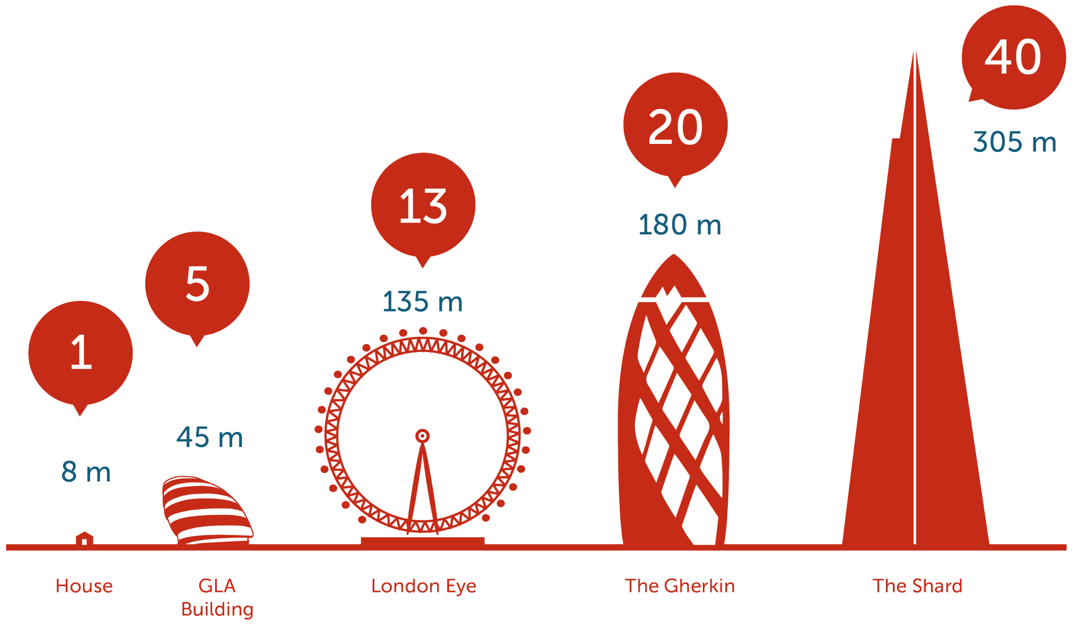


Figure 2.1: Using relative estimated values for user stories with different sizes. Image source [61].

level is the textual description. This requires the automated agile effort estimation approaches to utilise Natural language Processing (NLP) methods to extract features from the text.

The first study, published in 2011, is the work of Abrahamsson et al. [22]. They proposed to train a prediction model on 17 features extracted from user stories, such as the priority and number of the characters in the user story and 15 binary variables representing the occurrence of 15 keywords in the user stories. These 15 keywords are the 15 most frequent terms in user stories. They used regression models, Neural Networks (NN), and Support Vector Machines (SVM) to build estimation models for two industrial case studies, one consisting of 1,325 user stories and the other of 13 user stories. The best results were obtained by SVM.

Five years later, Porru et al. [23] proposed to classify user stories into SP classes. Their approach uses features extracted from 4,908 user story descriptions recorded in Jira issue reports collected from eight open-source projects. Specifically, they extract the type of the issue, the component(s) assigned to it, and the TF-IDF derived from the title and description of the issue. Their study confirmed Abrahamsson et al.'s findings that the user story and its length are useful predictors for story point estimation. Their results also suggest that more than 200 issues are needed for training the classifier to obtain a model with satisfactory accuracy.

In 2018, Scott and Pfahl [25] used developer-related features alongside the features extracted from 4,142 user stories of eight open-source projects to estimate the story points using SVM. Developer-related features include the developer's reputation, workload, work capacity, and the number of comments. The results showed that using only developers' features as input for the SVM estimator leads it to outperform Random Guessing, Mean, and Median baseline estimators. This approach also outperformed the two models using only features extracted from the text of the user story and using a combination of the developers' and text features.

At the same time, Soares [24] proposed the use of different NLP techniques with auto-encoder neural networks to classify user stories based on the semantic differences in their title in order to estimate their effort in terms of SP. He used TF/IDF and document embedding with four variants of auto-encoders and evaluated these models on 3,439 issue reports from six open-source projects. The results revealed no significant difference in the SP estimation accuracy of these approaches. Soares speculated that this might be due to the relative semantic simplicity of issue report titles.

In 2019, Choetkiertikul et al. [26] proposed a new approach to SP estimation based on the combination of two deep learning architectures to build an end-to-end prediction system called Deep-SE. They used raw user story text as input to their system. A word embedding was used to convert each word in a user story into a fixed-length vector, which is then fed to the deep learning architecture to map it to a space in which the semantically related stories are placed close to each other. In the final step, they used a regressor to map the deep representation into the SP estimate. They evaluated Deep-SE on 23,313 issues from 16 open-source projects and showed that it outperforms both baseline estimators and Porru et al.'s approach [23] based on Mean Absolute Error. Both these approaches (i.e., Deep-SE and Porru et al.'s approach) are investigated in Chapter 4.

Marapelli et al. [62] built a model based on a tree-structured RNN with Convolutional Neural Network (CNN) to predict SP for user stories. This model adopts a Bi-directional LSTM (BiLSTM), slightly improving Deep-SE's prediction performance. However, Marapelli et al. do not perform a statistical test to show if the difference is statistically significant.

Subsequently, Abadeer and Sabetzadeh [63] evaluated the effectiveness of Deep-SE for SP prediction with a commercial dataset of 4,727 user stories collected from a healthcare data science company. They found that Deep-

SE outperforms random guessing, mean and median baselines statistically significantly, however, with a small effect size.

Recently, Fu and Tantithamthavorn [21] proposed GPT2SP, a Transformer-based deep learning model for SP estimation of user stories. They evaluated GPT2SP on the dataset of 23,313 issues shared by Choetkiertikul et al. [26]. They investigated the performance of their model against Deep-SE for both within- and cross-project estimation scenarios and found that GPT2SP outperforms Deep-SE with a 6%-47% improvement over MAE for the within-project scenario and a 3%-46% improvement for the cross-project scenarios. However, when we attempted to use the GPT2SP source code made available by the authors, we found a bug in the computation of the Mean Absolute Error (MAE), which might have inflated the GPT2SP's accuracy reported in the original paper. Our proposed fix has been accepted and merged into the repository at https://github.com/awsm-research/gpt2sp/pull/2. The results obtained based on this fix revealed that, in the within-project scenario, GPT2SP outperforms the median baseline and Deep-SE in only six cases out of 16, where the difference is statistically significant in only three cases against the median (two with negligible and one with small effect size), and two cases against Deep-SE (both with negligible effect size). [4]

One important aspect of software engineering data which seems overlooked in previous work on agile Story Point (SP) estimation is the inherent high variability in the data, which can hinder the construction of accurate estimation models [65], [66]. The previous study on traditional software engineering effort estimation considered this aspect and successfully used clustering techniques to reduce the variability, and sometimes it led to the construction of more accurate effort estimation models [65], [67]–[72]. In Chapter 5, I investigate the effect of clustering user stories in open-source projects on agile SP estimation.

2.5. STORY POINT AS A COST DRIVER FOR AGILE EFFORT ESTIMATION

There are studies that leveraged human-estimated story points as a cost driver to build effort estimation models.

Zia et al. [73] considered human-expert estimated story size and story complexity to compute SP for user stories, and they used it to estimate the actual effort and cost for software projects. They introduced a regression-based model considering the characteristics of agile development. The model

[4] A technical report describing these results in more detail can be found at [64].

was applied to 21 previously developed small software projects and produced estimations with a mean absolute error of four days. Later, Popli and Chauhan [74] proposed a similar approach but evaluated the model on one small project.

Ungan et al. [75] investigated the accuracy of multiple linear estimators and a simple Artificial Neural Network estimator on 10 industrial projects as a case study. All the projects had their actual effort, and SP recorded by developers and their CFP were automatically approximated using a tool named CUBIT. Results showed that when the estimator uses CFP or SP as independent variables, the accuracy of effort estimation is low or at most acceptable, and none of the two models is superior to the other.

Raslan et al. [76] proposed a fuzzy logic technique-based effort estimation framework for user stories. The approach feeds expert-estimated SP alongside other parameters as input to a trapezoidal membership function to estimate the actual effort. The model is not evaluated on any real data; however, the authors designed a framework based on the proposed model in MATLAB to make it ready for evaluation and possible adoption.

Satapathy et al. [77] used a dataset of 21 projects from the work of Zia et al. [73] and evaluated the effort estimation accuracy of different machine learning techniques, namely, Decision Trees (DT), Stochastic Gradient Boosting (SGB), and Random Forest (RF). Based on the results, the DT model underperformed the technique previously proposed by Zia et al. [73]. Whereas the SGB and RF models performed better.

2.6. STORY POINT VS ACTUAL EFFORT

A few studies have investigated the correlation of the estimated story points and other effort proxies in the literature [16], [26], [78], [79]. However, the correlation between Story Points, as an undisciplined measure, and Function Point (FP), as a disciplined method of measuring the software size, has been an open debate for years since different experiments resulted in opposite findings [78], [79].

The first study assessing the relationship between SP and functional size measures (FSM) was carried out in 2011 by Santana et al. [16]. The authors of this study quantitatively analysed the relationship between FP and SP in a case study involving 2,191 user stories from 18 iterations of an agile software project developed by a private company. They found a strong positive correlation between SP and FP (Spearman's $\rho = 0.71$). Subsequently, Huijgens and Solingen [78] replicated Santana et al.'s work on a different case study and found a contrasting result. They gathered data from 14 iterations performed by two teams (*A* and *B*) in a Dutch banking organization that recorded estimations in SP and computed the size in FP for all iterations. The results of Spearman's rank correlation revealed a medium (-0.36) and strong (-0.60) negative correlation between SP and FP for Teams *A* and *B*, respectively.

SP has also been compared to the COSMIC function points and actual effort. Salmanoglu et al. [79] compared the correlation of SP and CFP with the actual effort spent on agile software development. They carried out three case studies from three large Turkish companies producing software solutions for security, financial, and telecommunication industries, reporting CFP, SP and the actual effort measured in person-hours. They plotted SP and CFP values against the actual effort to measure the linearity of functional size and actual effort and observed a stronger correlation between CFP and actual effort in comparison with SP.

Choetkiertikul et al. [26] carried out a preliminary analysis on the relationship between SP and development time on a set of 16 open-source projects mined from Jira repositories for a total of 23,313 issues [26]. The main aim of their study, however, was to build a machine-learning model to estimate SP (see Section 2.4). They computed development time from the issue changelog by considering the duration between the time the issue's status was set to *In-Progress* and the time it was set to *Resolved*. This was regarded by the authors as the most representative proxy for the actual effort they were able to extract from the data with respect to the completion time of an issue. However, this definition also includes the waiting time between development stages as part of the development time. In Chapter 6, I adopt a similar proxy for the issue resolution time and use two additional proxies for development time (proposed in Chapter 3) to take into account the waiting time (which is usually considered part of the development time) [27]. Choetkiertikul et al. [26] found a positive correlation between the SP and their proxy for development time with a mean of 0.47 and 0.51 and a standard deviation of 0.19 and 0.18, according to the Spearman's rank and Pearson correlation, respectively. Our investigation on a larger dataset showed a positive but weaker correlation, especially based on the Pearson correlation coefficient (Chapter 6). Across the 32 projects investigated in Chapter 6, we obtained a mean value of 0.40 for Spearman's ρ and 0.35 for Pearson's r coefficient, with a standard deviation of 0.11 for both.

Chapter 3

The TAWOS Dataset

Published as: “A Versatile Dataset of Agile Open Source Software Projects”
By: Vali Tawosi[1], Afnan Al-Subaihin[1], Rebecca Moussa[1], and Federica Sarro[1]
[1] Department of Computer Science, University College London, United Kingdom
At: The 19th IEEE International Conference on Mining Software Repositories (MSR’22)

Abstract – Agile software development is nowadays a widely adopted practice in both open-source and industrial software projects. Agile teams typically heavily rely on issue management tools to document new issues and keep track of outstanding ones, in addition to storing their technical details, effort estimates, assignment to developers, and more. Previous work utilised the historical information stored in issue management systems for various purposes; however, when researchers make their empirical data public, it is usually relevant solely to the study’s objective. This chapter presents a more holistic and versatile dataset containing a wealth of information on more than half a million issues from 44 open-source agile software, making it well-suited to several research avenues and cross-analyses therein, including effort estimation, issue prioritization, issue assignment, and many more. We make this data publicly available on GitHub to facilitate ease of use, maintenance, and extensibility.

3.1. INTRODUCTION

The early 2000s witnessed a surge in the adoption of Agile Software Development alongside the release of the *Agile Software Development Manifesto* in 2001 [51]. Agile techniques boast a faster response to unanticipated alterations that can arise during development, such as changes in user requirements, development environments, and delivery deadlines; typically contrasted with traditional ‘plan-based’ project development, which operates under the assumption that software is specifiable and predictable [80]. Agile Software Development is currently among the most common software development methods in project management [81].

Managing agile software development is commonly aided by an issue tracking tool, which allows agile teams to log and organize outstanding development tasks (e.g., bug fixes, functional and non-functional enhancements), in addition to hosting meta-data related to these tasks. Issue tracking tools, such as Jira [82], provide a trove of historical information regarding project evolution that promises great value for Empirical Software Engineering research. Such data has been employed to address many software engineering problems such as effort estimation [26], [29], task prioritization [83]–[85], task assignment [86], task description enhancement [87], iteration planning [88] and exploring social and human aspects [89]–[92]. However, the data made available by previous empirical studies are usually mainly relevant solely to the study's objective. Therefore, we aim to pave the way for a more holistic and versatile dataset containing a wealth of information on open-source software projects, which can serve as a single source for many possible research avenues and enable novel investigations on the interplay of multiple factors as well as draw observations across multiple research studies.

We call this dataset the TAWOS[1] (**T**awosi **A**gile **W**eb-based **O**pen-**S**ource) dataset. It encompasses data from 13 different repositories and 44 projects, with 508,963 issues contributed by 208,811 users.

The dataset is publicly hosted on GitHub [93] as a relational database and designed such that it is amenable to future expansions by the community. Prospective contributors are welcome to join our effort to maintain, grow and further enhance the database by issuing a pull request on GitHub.

3.2. DATASET DESCRIPTION

This section describes how the data is sampled from the web, how it is organised into, and stored as a relational database to ease access and usage, and the essential characteristics that make it serviceable to several empirical software engineering research areas.

3.2.1. Data Extraction

This dataset was mined during the latter half of October 2020. The mining process targeted 13 major open source Jira repositories[2]: *Apache, Appcelerator, Atlassian, DNNSoftware, Hyperledger, Lsstcorp, Lyrasis DuraSpace, MongoDB, Moodle, MuleSoft, Spring, Sonatype, and Talendforge*. Most of these repositories were employed by

[1]Tawos means Peacock in several of the middle-eastern languages.

[2]A repository in Jira is a collection of projects usually under development by a single company/organisation, or a group of inter-related teams, that share resources and follow the same organisational regulations.

previous work, and they all used Jira as an issue management platform, which ensures uniformity of structure and availability of information. From each of these repositories, projects were selected such that they adopt iterative development and record story points for their issues, thus suggesting that they follow an agile methodology. We considered projects that have at least 200 issues with recorded story point entries in order to have enough data to enable statistical analyses resulting in meaningful conclusions.

A total of 904 projects from the aforementioned repositories were considered, among which we selected the 44 that satisfy the collection constraints. To extract issue information, we used the Jira REST Java Client (JRJC) [94]; JRJC was used alongside our own tool, implemented in Java, to extract further features that are not implemented in JRJC (see Section 3.2.5).

3.2.2. Data Storage

The final dataset is modelled and stored as a relational database. This enables users of the dataset to employ SQL for easy horizontal and vertical data sampling in addition to allowing easier future expansion. We elected to host the dataset in the MySQL Database Management System as it is free, lightweight, and ubiquitous. The database can be downloaded from a GitHub repository together with the instructions on how to install and use it [93].

3.2.3. Data Characteristics

The TAWOS dataset contains 508,963 issues from 44 projects. The projects are diverse in terms of different project characteristics. Each project contains issues that range from 313 to 66,741 issues. The projects span different programming languages, different application domains and different team geographical locations. Table 3.1 shows the number of various elements for each of the projects contained in the dataset currently. Those include the number of all issues, issues categorised as a bug report, distinct users (i.e. bug report contributors, etc.), developers, change logs and comments, links to other issues, components, sprints, versions, and the number of issues with story points assigned.

3.2.4. Data Structure

Figure 3.1 shows the Entity-Relationship Diagram of the database. The core entity is the `Issue` table, which holds the main information about issue reports. Some of its fields are directly extracted from the issue report, such as the issue type (e.g., story, bug, improvement), status (e.g., open, in progress, closed), description, etc., whereas others are derived from the information stored and/or the events that occurred during the issue's lifecycle. We elaborate on these derived fields in Section 3.2.5.

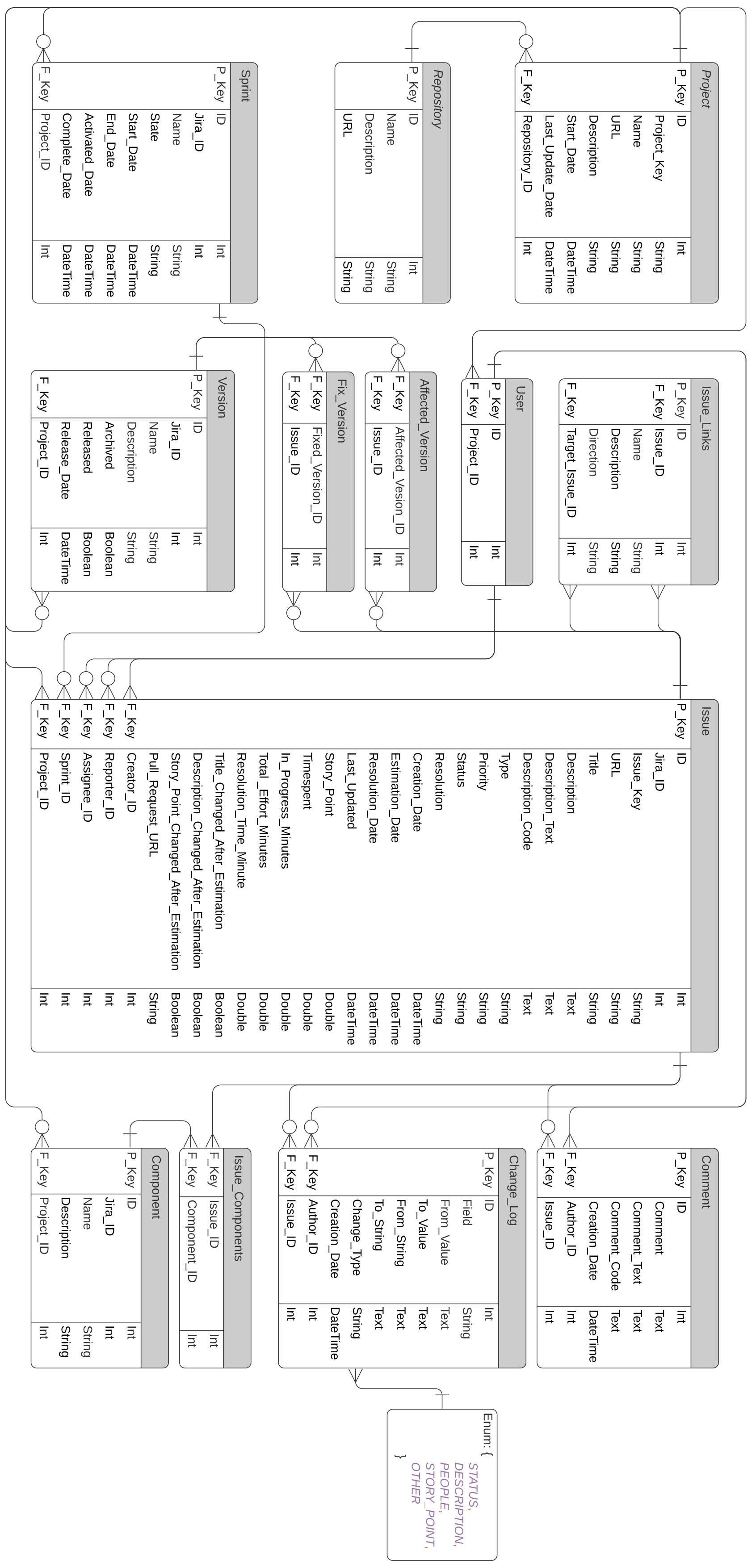


Figure 3.1: Entity-Relationship Diagram (ERD) for the TAWOS Issues Database.

Other important tables are `Comment` and `Change_Log` tables. Comments hold the documented discussions of the team around the issue development. Change logs hold all the changes made by the users on the issue report by recording the field that received the change, the previous value, the next value and the nature of the change. Both these tables store the chronological order of the events in the `Creation_Date` field. Information about the Sprints, Versions and Components of the issues are also stored in separate tables. The `Issue_Links` table captures the links between the issues. The `User` table stores all the distinct users who interacted with each project and links the events and information to their authors and user roles. Any personally identifiable information of users, like their usernames and emails, are redacted from this dataset.

3.2.5. Computed and Derived Fields

To further enrich the dataset, we have augmented the mined data with several additional features that are computed or derived from the source Jira repositories, as described below.

Issue Description Text and Code. The `Description` field holds the long description of the user story or bug report, which can contain natural text interleaved with code snippets or stack traces. To facilitate processing, we separate the code snippets/stack traces and the natural text describing the issue into the `Description_Code` and `Description_Text` fields, respectively. We maintain the original description in the `Description` field. The same is done for the `Comment` field, from which we extract the `Comment_Code` and `Comment_Text`.

This is motivated by previous work showing that code tokens may have a different meaning from those found in natural language text, hence ought to be analysed separately [23], [25], [29].

Resolution Time. The field `Resolution_Time_Minutes` stores the time span (in minutes) between when an issue is created and when it is marked as "*Resolved*". This period can be considered an approximation of the time the development team took to resolve the issue. This is usually the target variable used for bug resolution/fixing time estimation [95]–[97].

Other proxies for time are provided, such as `In_Progress_Time` and `Total_Effort_Time`, indicating, respectively, the implementation time and the development (including code review and testing) time.

SP Estimation Date: This field records the time when the `Story_Point` field of the Jira issue report was populated by the developer. This information might be useful, for example, for studies on software effort estimation, in order to properly take into account the chronological order of the estimates and avoid unrealistic usage of the data as described in previous studies [98]–[100].

Date and Time. The date and time stored in different Jira repositories may have different timezones, as the projects usually have contributors from all around the World.

Therefore, we converted and stored all dates and times to a unified timezone, namely the Coordinated Universal Time (UTC).

Field Change Flag. It is important to keep track of the changes developers made to some of the issue fields. For example, the title and description of the issue are two important pieces of information used by recent automated approaches to produce effort estimates [26]; therefore, it is important to know whether these fields have been edited after the initial estimate was done. The `Title_Changed_After_Estimation` and `Description_Changed_After_Estimation` fields store this flag. We also provide a flag that shows whether the SP has been changed after the initial estimate. Note that these flags are based on the change logs of the issue.

Change Type in Change Log. This field is calculated to categorise change log updates into one of five categories: “STATUS” indicates a change from one status to another in the Jira workflow of a given issue; “DESCRIPTION” indicates a change to the issue title or description; “PEOPLE” indicates that the user (Change_Log.Field=’assignee’ or ’reporter’) of the issue was changed; “STORY_POINT” indicates that the Story Point field of the issue was updated. Any other changes were categorised as “OTHER”.

3.2.6. Extensibility and Maintainability

The TAWOS database is designed such that it is easily extensible by attaching additional information to the corpus. This can help facilitate studying different problems and/or aspects of the same problem. Sharing and managing the dataset as a GitHub repository enables us to update, expand and enrich its content, whether by us or by the community as external contributions (i.e., pull requests). GitHub also guarantees that the information can be safely stored long-term, thus preventing the issues often faced in previous work where the data provided are not reachable anymore (e.g., due to the use of a volatile storing platform such as institutional webpages which change when researchers move to another institution).

3.3. ORIGINALITY AND RELEVANCE

Previous studies have extracted information from issue reports managed in Jira to build predictive models for Story Point (SP) estimation in agile software projects [23], [25], [26], however not all of them have made their data public [23], [25]. Choetkiertikul et al. [26] shared their data in a replication package [101]; however, it only consists of features considered in their study (i.e., the issue key, title, description, and story point of the mined issues).

The dataset presented herein encompasses all the projects considered in previous

Table 3.1: Descriptive statistics of the TAWOS dataset.

Repository	Project Name	Project Key	Programming Language	# Issues	# Bugs	# Users	# Developers	# Change Log	# Comments	# Links	# Components	# Sprints	# Versions	# Story Points
	Crowd	CWD	Java	4,311	1,841	2,663	105	62,408	7,440	2,624	50	44	227	214
	Confluence Cloud	CONFCLOUD	Java	23,409	10,071	24,064	513	321,439	64,655	7,694	147	477	17	352
	Software Cloud	JSWCLOUD	Java	11,702	3,505	15,187	211	201,512	30,143	4,492	33	74	68	318
	Jira Cloud	JRACLOUD	Java	25,669	8,339	30,020	557	295,951	74,473	8,176	66	59	170	361
	Confluence Server	CONFSERVER	Java	42,324	25,477	30,755	422	1,608,633	125,591	23,401	104	565	1,121	662
Atlassian	Atlassian Software Server	JSWSERVER	Java	12,862	6,007	15,468	182	304,682	35,400	5,724	44	70	433	351
	Jira Server	JRASERVER	Java	44,165	20,630	36,585	462	1,162,959	130,457	22,020	115	50	598	380
	Bamboo	BAM	Java	14,252	6,050	7,092	107	256,321	28,638	6,330	115	14	391	528
	Clover	CLOV	Java	1,501	531	347	20	25,812	2,259	338	15	48	63	387
	FishEye	FE	Java	5,533	2,896	2,371	74	112,723	8,914	2,044	9	109	245	240
	Mesos	MESOS	C++	10,157	4,891	1,282	252	108,349	30,152	6,342	42	227	87	3,272
Apache	MXNet	MXNET	C++	1,404	373	156	50	49,295	384	90	9	41	0	209
	Usergrid	USERGRID	Java	1,339	349	97	37	15,435	1,535	270	15	38	8	487
	Command-Line Interface	CLI	JavaScript	645	399	165	29	10,956	2,233	188	12	98	145	374
	Titanium Mobile Platform	TIDOC	JavaScript	3,059	1,344	421	62	81,454	7,712	710	6	217	261	1,297
	Aptana Studio	APSTUD	JavaScript	8,135	6,152	3,365	15	107,961	19,138	1,606	49	12	91	890
Appcelerator	Appcelerator Studio	TISTUD	JavaScript	5,979	3,455	654	63	147,215	19,880	4,051	56	163	126	3,406
	The Titanium SDK	TIMOB	JavaScript	22,059	15,742	3,170	161	483,361	83,252	11,120	52	301	568	4,665
	Appcelerator Daemon	DAEMON	JavaScript	313	123	36	5	4,062	469	90	44	62	20	242
	Alloy Framework	ALOY	JavaScript	1,519	646	386	30	36,312	4,491	586	15	118	172	315
DNN Tracker	DotNetNuke Platform	DNN	C#	10,060	7,319	1,092	33	197,067	32,015	3,766	143	NA	70	2,594
	Blockchain Explorer	BE	JavaScript	802	164	149	64	8,621	1,634	300	0	47	0	373
	Fabric	FAB	Go	13,682	3,562	1,283	457	151,811	23,056	5,312	26	142	55	636
Hyperledger	Indy Node	INDY	Python	2,321	826	133	59	40,111	5,884	1,626	6	76	26	681
	Sawtooth	STL	Python	1,663	318	174	56	15,800	576	454	29	22	4	966
	Indy SDK	IS	Rust	1,531	396	177	92	21,842	2,971	602	10	75	30	720
Lsstcorp	Lsstcorp Data management	DM	Python	26,506	2,551	277	211	310,891	71,744	19,722	259	396	4	20,664
Lyrasis	Lyrasis Dura Cloud	DURACLOUD	Java	1,125	374	32	12	11,559	1,443	264	14	7	86	666
	Compass	COMPASS	Java	1,791	737	484	17	23,617	2,077	820	87	91	77	499
	Java driver	JAVA	Java	3,560	1,028	1,439	35	42,995	11,018	772	35	46	107	238
MongoDB	C++ driver	CXX	C++	2,032	502	409	39	30,193	4,756	838	13	56	70	224
	MongoDB Core Server	SERVER	C++	48,663	22,342	8,837	452	1,030,545	136,823	40,084	37	NA	444	784
	Evergreen	EVG	Go	10,299	2,636	300	67	204,228	16,939	2,866	6	NA	26	5,402
Moodle	Moodle	MDL	PhP	66,741	41,355	12,230	554	1,298,195	481,606	52,356	97	151	373	1,594
Mulesoft	Mule	MULE	Java	11,816	5,421	1,449	146	233,760	16,627	3,622	129	311	274	4,170
	Mule APIkit	APIKIT	JavaScript	886	467	123	34	16,137	744	154	19	124	96	473
Sonatype	Nexus	NEXUS	Java	9,912	5,975	2,896	82	168,909	26,159	3,956	91	143	167	1,845
Spring	DataCass	DATACASS	Java	798	166	205	10	7,070	919	226	11	54	154	243
	XD	XD	Java	3,707	610	189	31	43,227	4,120	940	18	66	37	3,705
	Talend Data Quality	TDQ	Java	15,315	6,288	708	131	249,243	33,438	8,590	88	144	245	1,843
	Talend Data Preparation	TDP	Java	5,670	2,180	320	48	107,565	6,187	3,388	10	79	68	813
Talendforge	Talend Data Management	TMDM	Java	9,137	6,374	478	110	173,623	31,071	5,438	31	76	141	297
	Talend Big Data	TBD	Java	4,624	2,731	553	98	70,596	5,447	1,348	35	46	149	344
	Talend Enterprise Service Bus	TESB	Java	15,985	4,451	590	118	169,426	17,929	2,228	40	90	371	1,000
	Total			508,963	237,594	208,811	6,313	10,023,871	1,612,399	267,568	2,232	5,029	7,885	69,724

studies[3] [23], [25], [26] augmented with more issues and features.[4] Furthermore, it includes 28 additional projects which have never been used by any of these previous studies.

A total of 31,960 issues from 26 projects stored in the TAWOS dataset has been used in this thesis to replicate and extend the work by Choetkiertikul et al. [26] (Chapter 4). This sample of issues has also been used to evaluate the effectiveness of clustering for SP estimation [29] (Chapter 5). A larger sample of 37,440 issues from 37 projects is also used in a large empirical investigation of the relationship between story points and actual development time of open-source issues [30] (Chapter 6). And finally, a sample of 9,806 issues from 15 projects are used to investigate the ability of SP estimation models to estimate development time in Chapter 7.

We believe that the TAWOS dataset can help expedite the research in the area of agile software development effort estimation. In addition to providing a unified benchmark for such studies, it also helps circumvent the challenges faced and the time consumed when mining such data from the web. For example, we note that Choetkiertikul et al. [26] could not mine the same data used in the study by Porru et al. [23], likely because the repositories mined had changed during the time period between the two studies.

The use of different data in similar studies hinders the immensely useful opportunity to draw observations from across different studies performed at different times around a certain subject matter. We hope that our dataset can help the community tackle this challenge. Although our dataset has been primarily designed to aid in software engineering estimation tasks, it also includes information relevant to other software engineering research and is designed to be expanded by other contributors. This allows and promotes the investigation of a wider range of SE aspects, as discussed in the next section.

3.4. RESEARCH OPPORTUNITIES

In addition to benefiting effort estimation studies, the TAWOS dataset promises value to many other areas of software engineering research, including developer productivity studies, iteration planning and task scheduling.

An important research topic in Requirement Engineering is **requirement prioritization** [83], [102], [103], **early failure prediction** [104], and especially in an agile setting, the selection of issues for the next iteration [105], [106]. The TAWOS dataset can support such studies by providing a large collection of issues, with known

[3]The only exception is the MuleStudio project used by Choetkiertikul et al. [26], for which we could not find the data source online.

[4]The TAWOS dataset has 485,650 more issues in total and 46,411 more issues with Story Points compared to the one shared by Choetkiertikul et al. [26]. It also contains more issues for each of the 16 projects included in Choetkiertikul et al. [26]'s dataset.

priorities and iterations (i.e., Sprints and Releases) coupled with various aspects providing a full-picture view of the issues, projects and assignees. Additionally, as the dataset makes historical project evolution from multiple repositories available, it enables cross-project analysis.

The TAWOS database provides information about the versioning of the software under development. This information includes the version's name, description, and release date and whether it is archived or released. Versions connect to issues via two relations: Affected versions and Fix versions. The former is the version where a bug or problem was found, whereas the latter is the version where a feature is released or a bug is fixed. This information can be used to track the bug's lifecycle, and possibly if the link to the pull request which resolves the bug is presented in the `Pull_Request_URL` field, it can be tracked to the code. This information opens up avenues of research in **software testing and maintenance**.

The TAWOS dataset also contains information on the developer assigned to a given issue, in addition to various information regarding resolution time and the assignee's statistics. Such data enables, for example, the use of machine learning models to help automatically recommend the best developer for a new issue. Additionally, the dataset provides other useful information that can be considered for optimising **task assignment**, for example, considering developers' workload [107]. The dataset also provides the issue status transitions, which can be used to analyse activities and events to predict the **time to fix a bug, or bug triage** [95]–[97].

3.5. SUMMARY

In this chapter, I introduced and described the TAWOS dataset, a holistic dataset of software issues mined from open-source software repositories maintained in Jira. We have indicated just some of the research avenues for which the TAWOS dataset could be exploited. We envision that the wealth of information provided, coupled with the ability for other researchers to participate in the growth of the dataset, will enable novel research endeavours on the interplay among several and different aspects of open-source agile software projects. For example, if a researcher uses our dataset to analyse the corpus of issue comments with regard to developers affects (e.g., emotions, sentiments, politeness), they can extend the dataset by issuing a pull request and thereby augmenting the existing data with the results of their investigation (e.g., augment the comments written by developers with emotions such as surprise, anger, sadness and fear). This data can be re-used in subsequent research investigating the interplay between, for example, developer emotions and productivity.

The potential users of this dataset are encouraged to consult our online documentation [93] in order to understand possible limitations and select data that best fits the aim of their investigations.

Chapter 4

Previous Methods to Estimate Story Points

Published as: "Agile Effort Estimation: Have We Solved the Problem Yet? Insights From A Replication Study"
By: Vali Tawosi[1], Rebecca Moussa[1], and Federica Sarro[1]
[1] Department of Computer Science, University College London, United Kingdom
At: The IEEE Journal of Transactions on Software Engineering (TSE)

Abstract – In the last decade, several studies have explored automated techniques to estimate the effort of agile software development. In this chapter, we perform a close replication and extension of a seminal work proposing the use of Deep Learning for Agile Effort Estimation (namely Deep-SE), which has set the state-of-the-art since. Specifically, we replicate three of the original research questions aiming at investigating the effectiveness of Deep-SE for both within-project and cross-project effort estimation. We benchmark Deep-SE against three baselines (i.e., Random, Mean and Median effort estimators) and a previously proposed method to estimate agile software project development effort (dubbed TF-IDF-SE), as done in the original study. To this end, we use the data from the original study and an additional dataset of 31,960 issues mined from TAWOS, as using more data allows us to strengthen confidence in the results and further mitigate external validity threats. The results of our replication show that Deep-SE outperforms the Median baseline estimator and TF-IDF-SE in only very few cases with statistical significance (8/42 and 9/32 cases, respectively), thus confounding previous findings on the efficacy of Deep-SE. The two additional RQs revealed that neither augmenting the training set nor pre-training Deep-SE led to an improvement in its accuracy and convergence speed. These results suggest that using semantic similarity is not enough to differentiate user stories with respect to their story points; thus, future work has yet to explore and find new techniques and features that obtain accurate agile software development estimates.

4.1. INTRODUCTION

In agile development, the software is realized through repeated iterations called sprints (usually 2–4 weeks) and is built in small incremental parts known as releases [59]. The development team designs, implements, tests, and delivers a distinct product version or a working release in each sprint. This allows adaptation to changed requirements at any point during the project life cycle. Accordingly, the development team is required to complete a number of user-valued functionalities, called user stories, in every sprint [1], [108]. User stories are specified in the form of one or two sentences in a user's everyday language. Because of these small incremental iterations, it is essential for an agile team to focus on estimating the effort required to complete every single user story rather than the whole project. This allows them to prioritise the user stories and to organise and manage the successful completion of sprints. The estimation at this granularity is referred to as task-level estimation herein.

Story Point (SP) is commonly used to measure the effort needed to implement a user story [1], [18] and agile teams mainly rely on expert-based estimation [26], [59]. However, similar to traditional software project effort estimation [100], [109], task-level effort estimation is not immune to the expert's subjective assessment [18]. Subjective assessment may not only lead to inaccurate estimations but also, more importantly to an agile team, may introduce an inconsistency in estimates throughout different sprints. Thus an intelligent task-level effort estimation technique seems exigent.

Abrahamsson et al. [22] point out the three advantages of using an intelligent task-level effort estimation technique for agile development. First, an automated technique can use all the project information available from its inception and the history of the previous tasks to produce estimations for new ones. Second, the decision of an automated technique is not influenced nor affected by any pressure from the opinions of other individuals. Third, an automated estimation is repeatable and predictable; thus, it will always provide a consistent prediction. One of the readily available information that intelligent task-level effort estimation techniques can use is the user story, which is composed of a title and a description of the issue, usually stored in issue-tracking systems (see Table 4.13 for a few examples of user stories).

For more than a decade, researchers proposed techniques to predict a task's required effort in SP using features extracted from user stories [22]–[26], [29]. In most cases, these studies urged that their model would be a decision support system for expert estimators in agile teams.

This study explores the methods introduced to estimate story points for agile development at the task level. We perform a close replication and extension of the study which had introduced the state-of-the-art, namely Deep-SE[1] (regarded as the

[1]Deep-SE has been published in 2019 in IEEE TSE and been cited 143 times (source Google Scholar) at the time of writing this chapter.

"original study" in this chapter) [26]. In close replications, the original study's conceptual, methodological, and substantive aspects are mostly kept invariant, and only minor variations are allowed to be introduced [110]. This type of replication serves as an initial check to see if the findings of the original study hold or can be generalized in slightly varied conditions, like the validation against a new dataset. If close replications confirm the findings of an original study, an empirical generalization can be established [110]–[112].

Deep-SE is an end-to-end deep learning-based approach which accepts user story (i.e., issue title and description of a task recorded in Jira issue management system) as input and estimates story point as an output. In this work, we replicate three out of six research questions investigated in the original study.[2] These research questions compare Deep-SE to three naïve baseline techniques and to another task-level effort estimation technique introduced by Porru et al. [23] (we refer to it as "TF-IDF-SE" herein[3]), and they also investigate the suitability of Deep-SE for cross-project estimation.

Our results show that Deep-SE does not statistically significantly outperform the Median baseline or TF-IDF-SE in at least half of the projects evaluated in the original study, which confounds the results of the original study [26]. We also observe that Deep-SE's accuracy in cross-project estimation is not better than within-project estimation, which corroborates the original findings [26].

In addition, we extend the study by evaluating the replicated research questions on an additional larger and more diverse dataset (consisting of 26 projects with 31,960 user stories in total), which is sampled from the TAWOS dataset [27] (Chapter 3) and called the Tawosi dataset in this chapter. The extended evaluation of Deep-SE, which we carried out using the Tawosi dataset, confirmed the findings of our replication and strengthened the confidence in its results. The results confirmed the conclusion made in the original study only for one research question (cross-project estimation). As for the two other research questions (i.e., sanity check and comparison with TF-IDF-SE), we found variations that led to a different conclusion than the original study's. We discuss the possible origin of the variations in Section 4.5.

We also pose two new research questions. The first one aims to investigate how

[2]Two of the three research questions from the original study, which are not replicated herein, investigate the usefulness of Deep-SE's internal components by replacing them with simpler options. Since the original study showed that the best results had been achieved with the components already used in Deep-SE, we do not replicate these two research questions. The third research question investigates the performance of Deep-SE in estimating adjusted story points. They adjust the estimated story points of each issue using various information extracted from the issue reports after the issue was completed [26]. We did not replicate this research question since the data used for this research question and the details to extract them were missing from the replication package provided by the original study [101]. Moreover, the results in the original paper showed that using the adjusted story point benefited all the methods by reducing the range of the SP distribution for all projects.

[3]This approach is called "TF/IDF-SVM" in the published paper. I renamed it to "TF-IDF-SE" in this chapter to be consistent with the other chapters of this thesis.

using additional user stories from within-company and across different projects based on a realistic chronological order can affect Deep-SE's prediction performance. The second one explores the effectiveness of an expensive pre-training step used in the original study on the accuracy of the estimation and the speed of the convergence of Deep-SE. As stated by the original study, pre-training needs many hours to run and as many as 50,000 issues (without labels) per repository.[4].

The answers to these research questions showed that augmenting the training set with issues from other projects has no appreciable effect on the prediction accuracy of Deep-SE. Also, pre-training the lower levels of Deep-SE does not only not significantly improve its accuracy, but it also does not have a tangible impact on its convergence speed towards the best solution; thus, this step can be disregarded.

Section 2.4 described previous work proposing intelligent techniques to estimate story points. Table 4.1 summarises those studies and compares them to this study.

The rest of this chapter is organised as follows. Section 4.2 briefly describes the structure of Deep-SE and TF-IDF-SE. Section 4.3 describes the design of our replication study in detail, including the research questions, the data and experimental method used to address them, and threats to validity. Section 4.4 reports and discusses the results. Further insights on differences between the results of the original study and our replication are discussed in Section 4.5. Finally, Section 4.7 concludes the chapter with a plausible answer to the question "*Why couldn't Deep-SE outperform the baseline techniques for all the cases?*" and suggestions for future work.

4.2. THE DEEP-SE AND TF-IDF-SE APPROACHES FOR AGILE EFFORT ESTIMATION

This section provides an overview of the two story point effort estimation techniques used in this study, namely Deep-SE [26] and TF-IDF-SE [23], which make up the current state-of-the-art in agile effort estimation.

Both Deep-SE [26], and TF-IDF-SE [23] leverage the similarity between the target user story and previously estimated user stories to come up with an estimation for the target user story. TF-IDF-SE relies on one of the simplest techniques for text similarity, namely the term frequency-inverse document frequency (TF/IDF) statistic. On the other hand, Deep-SE uses advanced techniques in deep learning to exploit the semantic similarity between user stories. When Choetkiertikul et al. [26] proposed Deep-SE, it was natural to compare it with TF-IDF-SE. In the following, we provide a brief description of each approach and refer the reader to the papers where they were proposed originally for a full description [23], [26].

[4]A repository in Jira is a collection of projects usually under development within a single company or a collection of inter-related organisations/teams which share resources and follow the same organisational regulations.

Table 4.1: Summary of the Related Work.

Reference	Approach	Baseline	Measure	Data	Results
Abrahamsson et al. [22], 2011, IEEE Int. Sym. on Empirical Software Engineering and Measurement	Regression-based ML (Regression, NN, SVR)	expert estimation	MMRE, MMER, PRED(25%)	17 features extracted from 1,338 issue reports from two industrial projects	In one project, the developers' subjective estimates work better than the proposed approach (SVR), in the other project the accuracy of SVR lies in the range of subjective estimates of developers
Porru et al. [23], 2016, Int. Conf. on Predictive Models and Data Analytics in Software Engineering	Classification-based ML (KNN, NB, DT, SVM), The SVM variant is referred to as "TF-IDF-SE" in this paper	ZeroR (always selects as the outcome the most represented class)	MMRE	TF/IDF features and length of the title and description, and type and component of 4,908 issue reports from eight open-source projects	Estimating SP using an automated tool is feasible. SVM results in an MMRE between 0.16 and 0.61 after initial training of more than 200 issues
Scott and Pfahl [25], 2018, Int. Conf. on Software and System Process	Classification-based ML (multiple two-class SVMs)	Mean, Median, and Random baselines	MAE and SA	TF/IDF features, length, and five developer-related features extracted from 4,142 issue reports from eight open-source projects	SVM classifier with only five developer-related features outperforms SVM with textual features or a combination of textual and developers' features
Soares [24], 2018, IEEE Int. Joint Conf. on Neural Networks	Classification-based ML (AutoEncoder NN)	TF/IDF with SVM	Precision, Recall and F-measure	Distributed memory paragraph vector and distributed bag-of-words paragraph vector extracted from the title of 3,439 issue reports from six open-source projects	Found no significant difference among the different variants of AutoEncoders used in the study regarding SP estimation accuracy
Choetkiertikul et al. [26], 2019, IEEE Transactions on Software Engineering	Regression-based deep-learning called Deep-SE (word embedding + LSTM + RHWN + regression)	Mean, Median, and Random baselines, and KNN	MAE, MdAE, and SA	Title and description of 23,313 issue reports from 16 open-source projects, and dataset used by Porru et al. [23]	Deep-SE outperforms the baselines and TF-IDF-SE in all cases
Abadeer and Sabetzadeh [63], 2021, IEEE 29th International Requirements Engineering Conference Workshops (REW)	Deep-SE (word embedding + LSTM + RHWN + regression)	Mean, Median, and Random baselines	MAE, MdAE, and SA	Title and description of 4,727 issue reports from one industrial project	Deep-SE outperforms the baselines
Fu and Tantithamthavorn [21], 2022, IEEE Transactions on Software Engineering	Transformer-based deep-learning called GPT2SP	Mean and Median baselines and Deep-SE	MAE	Title and description of 23,313 issue reports from 16 open-source projects used by Choetkiertikul et al. [26]	GPT2SP outperforms Deep-SE and the baselines statistically significantly
This Study	Deep-SE (word embedding + LSTM + RHWN + regression) and TF-IDF-SE (SVM)	Mean, Median, and Random baselines	MAE, MdAE, and SA	Title, description, type and component of 31,960 issue reports from 26 open-source projects, in addition to the datasets used by Choetkiertikul et al. [26]	Deep-SE does not outperform the baselines and TF-IDF-SE statistically significantly in all cases

Abbreviations: ML = Machine Learning / SVM = Support Vector Machine [113]/ SVR = Support Vector Regression [41] / NN = Neural Network [114]/ KNN= K Nearest-Neighbour [32]/ NB = Naïve Bayese [115]/ DT = Decision Tree / LSTM = Long-Short Term Memory [116], [117] / RHWN = Recurrent Highway Network [118]/ MAE = Mean Absolute Error / MdAE = Median Absolute Error / SA = Standard Accuracy [119], [120] / MMRE = Mean Magnitude of Relative Error [121]/ MMER = Mean Magnitude of Error Relative to Estimation (a.k.a estimation MMRE) [121]/ PRED(l%)= Prediction at level l [121]

4.2.1. Deep-SE

Choetkiertikul et al. [26] proposed Deep-SE as a deep learning model to estimate the story point of a single software task, which is represented by an issue report containing the title and description of the issue.

Their model is composed of four components: (1) Word Embedding, (2) Document representation using Long-Short Term Memory (LSTM) [116], [117], (3) Deep representation using Recurrent Highway Network (RHWN) [118], and (4) Differentiable Regression.

The first component converts each word in the title and description of issues (i.e., user story) into a fixed-length vector (i.e., word embedding). These word vectors serve as an input sequence to the LSTM layer, which computes a vector representation for the whole document. The document vector is then fed into the RHWN, which transforms the document vector multiple times before outputting a final vector which represents the text. The vector serves as input to the regressor, which predicts the output story point. The word embedding and LSTM layers are pre-trained without using SP values to come up with a proper parameter initialization for the main deep structure (referred to as the pre-trained language models), which helps achieve faster convergence of the model.

Pre-training leverages two sources of information: the predictability of natural language and the availability of free texts without labels (e.g., issue reports without story points). The first source comes from the property of languages that the next word can be predicted using previous words, thanks to grammar and common expressions [26]. As per Choetkiertikul et al.'s [26] suggestion, we used the pre-trained language models that they shared, given that this step takes longer to execute. The pre-trained language models are trained on a corpus of 50,000 issues selected from each repository[5]. The issues are not required to have story points to be included in the pre-training stage and may also come from projects that are not considered for the SP estimation study.

Moreover, Deep-SE does not apply any pre-processing on the textual input (i.e., title and description) to remove non-natural text like links and code snippets. Thus, we did not apply any pre-processing in order to perform a close replication.

Furthermore, Deep-SE aims at estimating a value closest to the target SP value. To this end, it uses a regressor in its last layer; thus, the estimate could be a real number rather than an integer. Although the use of the Fibonacci series, the T-Shirt size, or the Planning Poker card set[6] for SP scales is usually recommended and applied by expert

[5]No explanation is given in the original study on how the 50,000 issues are selected. We noticed that their pre-training dataset takes into use every issue available in a repository until the number of issues reaches 50,000.

[6]Planning Poker card set includes the following numbers: 0, 0.5, 1, 2, 3, 5, 8, 13, 20, 40, 100, and ∞.

estimation and planning poker practitioners [59], [122], Deep-SE's estimations would not follow a Fibonacci scale, and the authors did not round the predicted SPs to the nearest SP integer value on the Fibonacci scale.

4.2.2. TF-IDF-SE

Porru et al. [23] treat SP estimation as a classification problem. Their idea is to use the title and description of issues (i.e., user stories) and issue-related data available at the estimation time (i.e., issue type and component), to build a machine-learning classifier for estimating the story points required to address an issue. They extract TF/IDF statistics and user story length from issue reports and use issue type and the component of the issue to build a dataset in which rows represent issues and columns are the TF/IDF values alongside the binary values representing the association of an issue to a type or component. Specifically, they concatenate the title and description of the issue reports and call it context. They separate the code snippets (if any) from natural language text in the context and analyse two chunks separately to extract TF/IDF features. They concatenate three feature sets, two extracted from text and code chunks and one extracted from issue type and components, before reducing the dimension by feature selection methods. The selected features are then fed to a Support Vector Classifier to classify issues into SP classes.

Given that the TF-IDF-SE method treats the task-level effort estimation problem as a multi-classification one (i.e., each class label is a value in the Fibonacci scale), the number of classes must be determined a-priory. Thus, its adoption might not be directly applicable to projects that do not follow the Fibonacci scale [26].

4.3. EMPIRICAL STUDY DESIGN

This section provides a detailed description of the research questions investigated for the replication and extension of the original study [26], together with the data, benchmarks, and evaluation methods used to answer these questions. To carry out both the replication and extended study, we use the implementation and configuration made publicly available by Choetkiertikul et al. [26] for both Deep-SE and TF-IDF-SE [101], subject to some alterations needed to fix some errors.[7] To allow the replication of our work, our scripts and data are publicly available [123].

[7]Since Porru et al. [23] did not provide any public implementation for TF-IDF-SE, we use the one provided by Choetkiertikul et al. [101] Moreover, the link to the replication package provided in Porru et al. [23] is no longer maintained by the authors and when contacted the authors were not able to provide us with more data nor tools.

4.3.1. Research Questions

The first research question asked by Choetkiertikul et al. [26] investigates whether Deep-SE is a suitable method to estimate story points. To this end, they assess whether it is able to outperform simpler baseline estimators for within-project estimation (i.e., building a prediction model by using past issues of a given project and using such a model to predict the story points for new issues within the same project).

Specifically, they compare Deep-SE with the Mean and Median baselines and report the SA values for all three techniques. SA is a measure of their accuracy performance against Random Guessing (see Sections 4.3.3 and 4.3.4 for the definition of the benchmarks and the evaluation method). Therefore, in our replication, we investigate the same research question:

RQ1. Sanity Check: *Is Deep-SE suitable for story point estimation?*

We evaluate this research question through two sub-questions. We first replicate the original study by evaluating Deep-SE on the same data used by Choetkiertikul et al. [26]: **RQ1.1. Sanity Check - Replication**, where we compare Deep-SE's accuracy with baseline estimators and the results reported in the original study.

Second, we extend RQ1.1 by evaluating Deep-SE on a larger dataset comprising 26 projects (see Section 4.3.2 for detail) to further strengthen the confidence in the results of RQ1: **RQ1.2. Sanity Check - Extension.**

If we find that Deep-SE outperforms the baseline techniques for RQ1.1 and RQ1.2, we can confirm the conclusion made in the original study.

Choetkiertikul et al. [26] also compared Deep-SE to the previous state-of-the-art for within-project estimation proposed by Porru et al. [23] (i.e., TF-IDF-SE). This motivates our second research question:

RQ2. Deep-SE vs TF-IDF-SE: *How does Deep-SE perform against TF-IDF-SE in story point estimation?*

Similarly to RQ1, we investigate two sub-questions for RQ2 by replicating the same experiment with the same data used in the original study (**RQ2.1. Deep-SE vs TF-IDF-SE - Replication**) and by extending it and experimenting with a larger dataset (**RQ2.2. Deep-SE vs TF-IDF-SE - Extension**). Deep-SE outperforming TF-IDF-SE for both RQ2.1 and RQ2.2 will further confirm the conclusion made in the original study.

Another important question tackled by the original study is the performance of Deep-SE for cross-project estimation, i.e., building a prediction model using issues from another project and using such a model to predict the story points for the issues of a new target project for which there are no issues to train an accurate model [23], [26]. Cross-project estimation is generally deemed a more difficult task than within-project estimation since the training and target projects might be heterogeneous [124]–[126]. Thus, our third research question assesses the ability of Deep-SE in cross-project estimation, as posed in the original study:

RQ3. Cross-project Estimation. *Is Deep-SE suitable for cross-project estimation?*

We tackle this question by replicating the two experiments done by Choetkiertikul et al. [26] in our first sub-question **RQ3.1. Cross-project Estimation - Replication**. Specifically, in the first experiment to train Deep-SE, we use a project belonging to the same repository of the target project (i.e., *cross-project within-repository estimation*). In the second experiment, we use as training data the issues from a project belonging to a repository different from that of the target project (i.e., *cross-project cross-repository estimation*). We replicate this research question with the same pair of source and target projects and compare the results of Deep-SE to the Mean and Median baselines, as done in the original study. We observe that Choetkiertikul et al. [26] did not consider the temporal ordering of the issues' creation time in collecting issues for their training and testing sets for this research question. However, previous work on software engineering prediction models showed that neglecting contextual details (like the temporal ordering of the data) is a potential threat to the conclusion stability of the study [99], [127], [128]. In a real-world scenario, one would be able to use only the issues from past projects to train the model. Thus, all the issues in the training (and validation) set should precede, in creation time, the earliest created issue from the test set. Surprisingly, for 11 out of 16 experiments, more than 97% of the issues used to train the models in RQ3.1 was created after the start date of the target project, thus making the use of Deep-SE in practice infeasible.[8] Therefore, we extend the original analysis in order to assess Deep-SE's performance in a realistic scenario, which takes into account the chronological order of the data used for cross-project estimation. This motivates our second sub-question: **RQ3.2. Cross-project Estimation - Extension**, where we only use the issues created before the start date of the target project as the cross-project training data for Deep-SE. We compare the results of Deep-SE with Mean and Median baseline techniques.

Our next research question focuses on another practical usage scenario: The case where an engineer needs to estimate the effort for a target project that is new and for which there are not enough story issues realised yet; thus, there is not enough data from the project itself to train an accurate deep-learning-based prediction model. As a result, the data would need to be augmented with the use of external sources. This is a very common case for deep learning models, which need an abundance of training data to perform well. Previous work investigates how to augment the training set in different application domains [129], [130]. Our research question aims at analysing if augmenting the training set (composed of a few instances from the target project) with a larger number of issues from other projects helps Deep-SE produce more accurate estimates:

[8]Among the five remaining projects, for two of them, 75% and 37% of the issues were created after the start date of the target projects, and for three (in which MULESOFT was used as the source or target project) we could not determine the percentage as the start/end time of the MULESOFT project is not known as this project repository is no longer accessible.

RQ4. Augmented Training. *How does Deep-SE perform when using a training set augmented with other projects' issues?*

To answer RQ4, we use the same data splits used for RQ1.2, but we add to the training set historical issues available from all the other projects belonging to the same repository of the target project (i.e., all projects are within the same organisation), yet respecting the chronological order of the issues. In particular, we include in the training set only those issues from other projects within the same repository that is created before the earliest created issue in the validation set. Therefore, we augment the target project training data with every issue that a company already had in their repository at the time of predicting the story points for new issues of a target project. Using more instances in the training data changes its original distribution; however, the same data is used to train all approaches, and no transformation is applied to the test set. This allows for a fair and correct empirical analysis answering RQ4.

Our fifth and last research question investigates a technical aspect of the deep learning approach proposed by Choetkiertikul et al. [26]: The utility of pre-training for Deep-SE. Pre-training is described as a way to come up with a good parameter initialization for Deep-SE without using the labels (i.e., story points). Choetkiertikul et al. [26] pre-train the lower layers of Deep-SE (i.e., embedding and LSTM layers), which operate at the word and sentence levels. In the absence of pre-training, the parameters are typically initialized randomly, but Choetkiertikul et al. [26] stated that a good initialization (through pre-training of embedding and LSTM layers) allows a faster convergence towards good solutions. However, their study did not compare the performance of Deep-SE with and without using such a pre-training step. In this work, we investigate the actual benefit of using such a pre-training step since pre-training is very expensive both in terms of running time and amount of data required (Choetkiertikul et al. report that their pre-training step needs around 50,000 issues to work properly and took 46 hours and 48 minutes for nine repositories to run[26]). Given these limitations, if the use of pre-training does not improve the estimation performance (accuracy-wise and execution time-wise), its usage would not be advisable. This motivates our next research question:

RQ5. Pre-Training Effectiveness. *Does pre-training Deep-SE's lower levels improve its accuracy and/or convergence speed?*

To answer this question, we compare Deep-SE with random weight initialization to Deep-SE initialized with weights tuned by pre-training (i.e., the pre-training step used by Choetkiertikul et al. [26]).

The comparison is performed on estimation accuracy, running time, and the number of epochs each performs before converging to an optimal solution.

4.3.2. Data

To execute a close replication of the research questions to that of the original study, we adopted the datasets made publicly available by Choetkiertikul et al. [26]. Moreover, we also gather our own dataset to strengthen our confidence in the results and further mitigate the threat to the external validity of the study. We explain each of these datasets in the following, while a summary of the descriptive statistics of all three datasets is in Table 4.2.

4.3.2.1. The Choetkiertikul Dataset

Choetkiertikul et al. [26] mined 16 open-source projects from nine different repositories (namely Apache, Appcelerator, DuraSpace, Atlassian, Moodle, Lsstcorp, MuleSoft, Spring, and Talendforge) and gathered a total number of 23,313 issues (i.e., user stories) with recorded story points, after filtering out all those issues that had assigned a story point of zero, or negative, or an unrealistically large value (i.e., greater than 100). Choetkiertikul et al. collected their dataset on August 8, 2016, and made it publicly available [101]. We refer to this dataset as the Choet dataset and we use it to answer RQs 1.1, 3.1, and 5.

4.3.2.2. The Porru Dataset

Choetkiertikul et al. [26] also collected eight open-source projects stored in six open-source repositories, aiming at benchmarking Deep-SE against Porru's TF-IDF-SE approach using a common dataset.[9]

In total, the Porru dataset, as collected by Choetkiertikul et al. [26], [101], contains 4,904 issues. Among these eight projects, six are common with the Choet dataset (i.e., TIMOB, TISTUD, APSTUD, MESOS, MULE, and XD), although they contain a different subset of issues as Porru et al. applied a set of more restrictive filtering criteria than those used by Choetkiertikul et al. in building the Choet dataset [26]. These criteria are as follows: (1) Story points must have been assigned once and never updated afterwards. Updated story points may mean that the information provided in the issue report had misleading information, and they do not want to confuse the classifier with noisy input. (2) The issue must be addressed. Issues not addressed are likely unstable; hence, they might confuse the classifier. (3) Once the story points are assigned, the informative fields of the issue (i) must be already set and (ii) their value must not have been changed afterwards. They define informative fields used as dependent variables for classification, including Issue Type, Description, Summary, and Component(s). (4) The values in the story points field must correspond to one of those included in the

[9]The dataset used in Porru et al. [23] is not publicly available, so Choetkiertikul et al. [26] re-collected it by closely following the approach described in their paper [23]. This dataset slightly differs from the original dataset [23] in the number of issues.

Planning Poker cards set.

Choetkiertikul et al. made their version of the Porru dataset publicly available [101]. We refer to this dataset as the Porru dataset. We used this dataset to answer RQ2.1.

4.3.2.3. The Tawosi Dataset

In our empirical study, we also used issues extracted from the TAWOS dataset [27], which contains over 500,000 issues coming from 44 open-source projects found in 13 Jira repositories. Specifically, we sampled from TAWOS all those issues having an SP value specified and then filtered out all those that do not meet the criteria recommended by Porru et al. [23] (see Section 4.3.2.2) as they are more restrictive than those used by Choetkiertikul et al. [26], and could be more effective in removing noisy data points [23]. Our sample resulted in a total of 31,960 issues coming from 26 different projects (each having more than 200 issues). We will refer to this sample as the *Tawosi* dataset herein. The *Tawosi* data is available in our online replication package [123], and we used it in our empirical study to answer RQs 1.2, 2.2, 3.2, and 4.

The 26 projects in the Tawosi dataset belong to 13 different Jira repositories. Eleven of these 13 repositories were used by either Choetkiertikul et al. [26] or Porru et al. [23] and other previous work [24], [25], and the two additional ones, Hyperledger and MongoDB, were used in related work [88].

Twelve of the 26 projects in the Tawosi dataset are in common with the Choet dataset; however, they have a different number of issues, as we collected the data more recently and applied Porru's filtration criteria.[10] In addition, the Tawosi dataset includes 14 more projects (ALOY, CLI, DAEMON, TIDOC, CONFCLOUD, CONFSERVER, DNN, FAB, STL, COMPASS, SERVER, EVG, NEXUS, and TDP) with a total of 10,506 issues. Four projects in the Choet dataset (with a total of 2,087 issues) are not included in the Tawosi dataset: Three of them (i.e., USERGRID, BAM, and JRESERVER) were left with less than 200 issues after applying the Porru's filter, thus removed from the Tawosi dataset; while the MULESTUDIO project was no longer available on Jira when the TAWOS dataset was collected.

The Tawosi dataset contains all eight projects included in Porru's dataset, with 11,029 more issues. Moreover, the Tawosi dataset includes 18 additional projects with respect to Porru's dataset, with a total number of 16,027 additional issues.

[10]Four projects in our dataset have more issues than the same projects in the Choet dataset (4,652 additional issues in total for the TIMOB, DM, MDL, and MULE projects). This is due to the new issues created in these projects since Choet collected issues for their dataset. On the other hand, eight projects in our dataset have a smaller number of issues than the same projects in the Choet dataset (4,424 fewer issues in total for the MESOS, TISTUD, APSTUD, CLOV, DURACLOUD, XD, TDQ, and TESB projects). This is due to the more restrictive issue filtration policy that we followed in this study.

Table 4.2: Descriptive statistics of the three datasets used in this study.

Repository	Project (Abbreviation used in [26])	Key	Tawosi dataset						Choet dataset [26]						Porru dataset [26]					
			#Issues	Story Point					#Issues	Story Point					#Issues	Story Point				
				Min	Max	Mean	Median	Std		Min	Max	Mean	Median	Std		Min	Max	Mean	Median	Std
Apache	Mesos (ME)	MESOS	1,513	0	13	3.15	3	2.14	1,680	1	40	3.09	3	2.42	354	0	13	2.8	2	2.15
	Usergrid (UG)	USERGRID							482	1	8	2.85	3	1.40						
Appcelerator	Alloy	ALOY	241	0	13	3.71	3	2.32												
	Appcelerator Studio (AS)	TISTUD	2,794	0	40	5.48	5	3.23	2,919	1	40	5.64	5	3.33	1,172	1	144	5.53	5	4.79
	Aptana Studio (AP)	APSTUD	476	0	100	7.93	8	7.19	829	1	40	8.02	8	5.95	236	0	100	7.76	8	7.58
	Command-Line Interface	CLI	293	0	13	3.18	3	2.30												
	Daemon	DAEMON	205	1	13	5.58	5	3.76												
	Documentation	TIDOC	1,005	0	40	3.58	2	3.68												
	Titanium (TI)	TIMOB	3,915	0	20	4.68	5	3.32	2,251	1	34	6.32	5	5.10	600	0	6,765	16.89	5	276.19
Atlassian	Bamboo (BB)	BAM							521	1	20	2.42	2	2.14						
	Clover (CV)	CLOV	336	0	100	5.33	2	11.03	384	1	40	4.59	2	6.55						
	Confluence Cloud	CONFCLOUD	234	0	13	2.91	2	2.24												
	Confluence Server and Data Center	CONFSERVER	456	0	13	3.12	3	1.93												
	Jira Server and Data Center (JI)	JRESERVER							352	1	20	4.43	3	3.51						
DNNSoftware	DNN	DNN	2,064	0	100	2.05	2	2.56							866	0	13	1.91	2	1.25
DuraSpace	Duracloud (DC)	DURACLOUD	310	0	20	1.72	1	1.70	666	1	16	2.13	1	2.03						
Hyperledger	Fabric	FAB	303	0	40	2.69	2	3.20												
	Sawtooth	STL	206	0	5	2.09	2	1.19												
Lsstcorp	Data Management (DM)	DM	5,381	0	100	3.05	2	6.87	4,667	1	100	9.57	4	16.60						
MongoDB	Compass	COMPASS	260	1	8	3.55	3	1.85												
	Core Server	SERVER	519	0	20	2.58	2	2.40												
	Evergreen	EVG	2,824	0	8	1.43	1	0.86												
Moodle	Moodle (MD)	MDL	1,394	0	100	11.80	5	18.81	1,166	1	100	15.54	8	21.65						
MuleSoft	Mule (MU)	MULE	2,935	0	13	3.88	3	3.46	889	1	21	4.90	5	3.61	810	0	13	3.42	3	3.43
	Mule Studio (MS)	MULESTUDIO							732	1	34	6.40	5	5.39						
Sonatype	Sonatype's Nexus	NEXUS	1,425	0	40	1.70	1	1.82							420	0	8	1.13	1	0.86
Spring	SpringXD (XD)	XD	811	0	20	3.16	3	2.56	3,526	1	40	3.70	3	3.23	446	0	8	2.77	2	2.08
Talendforge	Talend Data Preparation	TDP	471	0	13	2.31	2	1.84												
	Talend Data Quality (TD)	TDQ	859	0	40	6.01	5	4.66	1,381	1	40	5.92	5	5.19						
	Talend ESB (TE)	TESB	730	0	13	2.13	2	1.45	868	1	13	2.16	2	1.50						
Total			31,960						23,313						4,904					

4.3.2.4. Validation Approach

Similarly to the original study, to perform within project estimation (i.e., for RQs 1, 2, and 5), for all datasets, we sorted the issues in each project in ascending order of their creation time and used the 60% oldest issues as the training set, 20% as the validation set and the newest 20% as the testing set, as has been done in the previous studies. This is because, in a real-world scenario, estimation for a new issue is made by using knowledge from the estimations of past issues. In particular, for RQs 1.1, 2.1, and 5, which are close replications, we used the same data splits used by Choetkiertikul et al. [26] to avoid any possible bias in the data that might be introduced by using a different sampling.

For RQ3 (cross-project estimation), the original study does not clarify how the data was split. Thus, we apply the same rate they used for RQ1.1 and divided the source project into three-forth (75%) train and one-fourth (25%) validation sets (following the 60% train-20% validation split rate) and used all issues of the target project as the test set.

Finally, for RQ4 (augmented training set), we use the same train-validation-test as RQ1.2, but we augmented the training set with additional issues from different projects within the same repository of the target one.

4.3.3. Benchmarks

As done in the original study, we use three commonly used baseline estimation techniques, i.e., Random Guessing, Mean, and Median estimators to benchmark Deep-SE.

Random Guessing (RG) is a naïve method that simply assigns the story point of a randomly selected issue to the target issue [119]. More formally, Random Guessing predicts a story point value y for the target case $issue_t$ by random sampling (with equal probability) all the remaining $n - 1$ cases and taking $y = r$; where r is the story point of the randomly drawn $issue_r$ from $1...n \mid issue_r \neq issue_t$ [119]. This method does not need any parameter estimation and any prediction system is expected to outperform it over time, otherwise, the prediction system would not be using any information about the target case. The results for RG are in the online appendix [123].

Mean and Median Effort are two baseline benchmarks commonly used for effort estimation techniques [38], [131], [132]. Specifically, the mean or median story point of the past issues is used as the predicted story point for a new issue.

4.3.4. Evaluation Measures and Statistical Analysis

Several measures have been used in the software effort estimation literature to measure the accuracy of the estimation models. These measures are generally built upon the error (or absolute error) between the predicted value and the actual value (i.e., $|Actual.value - Predicted.value|$).

Similarly to the original study, we discuss all results of our study based on the Mean Absolute Error (MAE) measure, while we report the Median Absolute Error (MdAE) and Standard Accuracy (SA) values in our online appendix for completeness [123].

These measures (defined in Equations 4.1, 4.2 and 4.3) are standardised measures which are not biased towards under- or over-estimates and which have been recommended in previous work [38], [119], [120].

Across n issues, the MAE and MdAE are computed as follows:

$$MAE = \frac{1}{n}\sum_{i=1}^{n} |actual_i - predicted_i| \tag{4.1}$$

$$MdAE = Median_{i=1}^{n}\Big\{|actual_i - predicted_i|\Big\} \tag{4.2}$$

where $actual_i$ is the actual effort from the historical data, $predicted_i$ is the predicted effort by the method and n is the number of issues in a given project.

SA is recommended as a standard measure to compare multiple prediction models against each other [119]. It is based on MAE and is defined as follows:

$$SA = \left(1 - \frac{MAE_{p_i}}{MAE_{p_0}}\right) \times 100 \tag{4.3}$$

where MAE_{p_i} is the MAE of the approach p_i being evaluated and MAE_{p_0} is the MAE of a large number (usually $1,000$ runs) of *random guesses*.

For a prediction model p_i which outperforms random guessing in terms of accuracy, SA will produce a number in the range $[0, 1]$. An SA value closer to zero means that the predictor p_i is performing just a little better than random guessing [38], [119]. For a prediction model which is outperformed by random guessing SA will produce a negative value. For a high-performance prediction model MAE and $MdAE$ should be lower and SA should be higher than the competitors.

To check if the difference in the results achieved by the two methods is statistically significant, we performed the Wilcoxon Ranked-Sum test (a.k.a. Mann–Whitney U test) on the distribution of the absolute errors produced by the methods under investigation. Specifically, we used a one-sided Wilcoxon test with a confidence limit of $\alpha = 0.05$ to check the following Null Hypothesis:

Hypothesis 4.3.1: Null Hypothesis

The distribution of absolute errors produced by the two prediction models P_i and P_j are not different.

If the test rejects the Null Hypothesis, the alternative hypothesis would be accepted:

Hypothesis 4.3.2: Alternative Hypothesis

The distribution of absolute errors produced by the P_i are different from those provided by the P_j.

Specifically, we performed a one-sided test since we are interested in knowing if a given model (i.e., Deep-SE) would commit a smaller estimation error than another model. In such a case, the one-sided p-value interpretation would be straightforward. To mitigate the risk of incorrectly rejecting the Null Hypothesis (i.e., Type I error) [100], we also analyse how the results would be when the Bonferroni correction is applied to cater for multiple hypothesis testing (i.e., the confidence limit is set as α/K, where K is the number of hypotheses). Therefore, herein we report the original p-value results of the Wilcoxon test (see Tables 4.4, 4.6, 4.8, 4.9, and 4.10). While, we analyse and discuss the results both using the confidence limit of α = 0.05 and the Bonferroni corrected one (α = 0.025), where applicable.

We also use a standardised non-parametric effect size measure (i.e., the Vargha Delaney's $\hat{A}_{12}$ statistic) to assess the practical magnitude of the difference between two methods, as recommended in previous work [38], [100], [133]. For two algorithms A and B, the $\hat{A}_{12}$ measures the probability of A performing better than B with respect to a performance measure. $\hat{A}_{12}$ is computed using Equation (4.4), where R_1 is the rank sum of the first data group being compared, and m and n are the number of observations in the first and second data sample, respectively.

$$\hat{A}_{12} = \frac{\left(\frac{R_1}{m} - \frac{m+1}{2}\right)}{n} \tag{4.4}$$

Based on Equation (4.4), if two algorithms are equally good, $\hat{A}_{12} = 0.5$. Respectively, $\hat{A}_{12}$ higher than 0.5 signifies that the first algorithm is more likely to produce better predictions. The effect size is considered negligible for $\hat{A}_{12} < 0.6$ (represented by an 'N'), small (S) for $0.6 \leq \hat{A}_{12} < 0.7$, medium (M) for $0.7 \leq \hat{A}_{12} < 0.8$, and large (L) for $\hat{A}_{12} \geq 0.8$, although these thresholds are not definitive [38]. We do not transform the $\hat{A}_{12}$ as we are interested in *any* improvement achieved by the methods [38], [134].

To perform the above analyses, we used the Wilcoxon Rank-Sum test and Vargha Delaney's $\hat{A}_{12}$ effect size available from the `stats` library in `R` v. 4.0.1 [135].

4.4. RESULTS

This section presents the results and discusses the answers to the research questions.

4.4.1. RQ1. Sanity Check

4.4.1.1. RQ1.1. - Replication

Results: In Table 4.3, under the column "Rep", we report the results obtained by Deep-SE and the baseline estimators (i.e., Mean and Median) on the Choet dataset in our replication study. Below we analyse and discuss the results based on the MAE measure.

We can observe that Deep-SE generally outperforms the Mean baseline estimator, obtaining lower MAE values on 14 out of the 16 projects (i.e., 88%) investigated in our replication study. The differences in 11 of them (i.e., 69%) are statistically significant, with two projects having a medium effect size, six having small, and three having negligible ones, as shown in Table 4.4. However, when Deep-SE is compared to the Median estimator, results reveal that the former obtains lower MAE values on 8 out of the 16 projects under study (i.e., 50%), with the differences being statistically significant in only 4 cases (25%) with the effect size being small in one case and negligible in the remaining three cases. All these cases remain statistically significant when considering the Bonferroni correction.

Discussion: These results are not in complete accordance with those obtained in the original study, where it is shown that Deep-SE achieves lower MAE values than both baseline techniques for all 16 projects (see the *Orig* column in Table 4.3) with statistically significant differences in 14 and 15 out of the 16 cases when compared to the Median and Mean estimators, respectively.

Variation in the results due to the use of stochastic techniques, such as deep learning, can be acceptable to some extent [5]. In fact, the variation we observed in the MAE values achieved for Deep-SE in the original study and this replication can be attributed to such a stochastic nature. However, the difference observed between the results of the baseline techniques reported in the original study and this replication suggests unjustifiable discrepancies since the baseline techniques are deterministic and should always yield the same results given the same datasets. Therefore, we further investigated the origin of such discrepancies by analysing the results and code provided in the original study's replication package and by contacting the authors. We found that the likely cause for these discrepancies is a fault in the computation of the MAE and MdAE of the baseline estimators and the original study's use of a possibly different approach for random guessing. Section 4.5 provides a detailed explanation of these causes.

Moreover, while investigating the implementation provided for Deep-SE, we noticed that Deep-SE performs a transformation on the distribution of the SPs in the pre-processing stage (see Section 4.5 for more details). This transformation is applied to the SP distribution of each project before being split into training/validation/test sets;

however, this pre-processing is not mentioned in the paper.

To further investigate the effect of such a transformation on the results, we used Deep-SE with the original SPs (i.e., without transforming its distribution), and we report the results in Table 4.3 under the *!Cut* column. Our results show that, in this case, all estimators generally perform worse. Specifically Deep-SE*!Cut* obtains worse MAE values than those it obtains when the transformation is applied in our replication study (*Rep*) in 15 cases (94%), with the difference being statistically significant in four of them (27%) and the effect size being small in one and negligible in the remaining three. While, the Mean and Median estimators perform worse in 16 (100%) and 14 (88%) cases, respectively with the remaining two cases being equivalent. The difference is statistically significant in 12 cases compared to the Mean estimator with medium effect size in 2, small in 7, and negligible in 2 cases. While, when compared to the Median estimator, the difference is not statistically significant for any of the cases.

We also observe differences between the results shown in Table 4.3 *!Cut* and the original study (*Orig*): Deep-SE performs worse without applying the SP transformation in all 16 cases considered; whereas the Mean and Median estimators obtain worse MAE values in 6 (38%) and 4 (25%) cases, respectively.

When we compare the performance of Deep-SE to that of Mean and Median when no transformation is applied, i.e., *!Cut*), we observe that Deep-SE outperforms the Mean and Median in 15 and 6 cases, respectively. Out of the 15 cases, the difference between Deep-SE*!Cut* and Mean is statistically significant in 12 cases, with a medium effect size in 3 and a small one in the remaining 9 cases. As for the Median estimator, out of the 6 cases, only 2 cases show statistically significant differences with a small effect size in one and a negligible effect size in the other. If we consider the Bonferroni correction, all 12 cases for the Mean estimator are still statistically significant, while the number of statistically significant cases for the Median estimator drops to just one having a small effect size.

We conclude that the transformation induces the techniques to produce a lower absolute error by reducing the range of the SP distribution of the entire dataset (including the test set), therefore suggesting more optimistic results than the ones that could be achieved in practice, where the test set is not available at prediction time. Indeed, when we train Deep-SE while correctly applying the transformation on the training set only (column *CutTrain* in Table 4.3), we find that in the majority of the cases (12 out of 16), its performance deteriorates; however, the difference is statistically significant in only one of these 12 cases with a negligible effect size.

In this study, we apply the transformation on the SP distributions for research questions RQ1.1 and RQ5 for replication purposes, while we use the original SP values to answer the remaining RQs (i.e., Deep-SE!Cut is the variant used in the remaining RQs).

4.4.1.2. RQ 1.2 - Extension

To answer RQ 1.2, we benchmark Deep-SE against the Mean and Median estimators using the Tawosi dataset.

The results reported in Table 4.5 show that Deep-SE outperforms Mean in 20 out of 26 cases (77%). However, the difference in absolute error is statistically significant in 16 (62%), with four cases having a negligible effect size, seven cases having a small effect size, two medium ones, and the remaining three having a large effect size as shown in Table 4.6. When compared to the Median estimator, Deep-SE obtains lower (better) MAE values in 10 out of the 26 cases investigated (38%), and only four (15%) show statistically significant differences, with one having a large effect size, one small, and the other two having a negligible effect size. As for the remaining cases, we observe that the Median estimator outperforms Deep-SE in 11 out of 26 projects (42%) and obtains the same MAE values in five other cases (20%). When considering the Bonferroni correction, the number of statistically significant differences does not change for the Mean estimator, while for the Median estimator, this number drops to two cases only, one with negligible and one with a large effect size.

Overall, the results obtained for RQs 1.1-RQ1.2 show:

Answer to RQ1: *Deep-SE statistically significantly outperform the baseline estimators in only 42% of the cases for within-project estimation and therefore does not pass the sanity check.*

4.4.2. RQ2. Deep-SE vs TF-IDF-SE

4.4.2.1. RQ2.1 - Replication

The original study evaluated Deep-SE and TF-IDF-SE on the Porru dataset and found that Deep-SE outperformed TF-IDF-SE for all eight projects under investigation.

To answer RQ2.1, we replicate the same experiment with the same data used in the original study.

Table 4.7 shows the results we obtained in our replication study using Deep-SE (i.e., *Deep-SE (Rep)*) and *TF-IDF-SE (Rep)*), together with the results obtained by the original study (*Deep-SE (Orig)* and *TF-IDF-SE (Orig)*, shaded in grey).

First of all, we observe that in our replication of TF-IDF-SE, we obtain exactly the same results as Choetkiertikul et al. [26] (i.e., the TF-IDF-SE results are fully reproducible), whereas the results we obtained for Deep-SE are different from those obtained in the original study [26]. Deep-SE outperforms TF-IDF-SE in six out of eight projects (75%) and, among these, the difference is statistically significant in four projects (i.e., MULE, XD, DNN and NEXUS), all with a small effect size (Table 4.8).

Table 4.3: RQ1.1 and RQ5. Results obtained for the Choet dataset in RQ1.1.: The column "Rep" shows the replication results, and the column "Orig" presents the original study results [26], both obtained by using Deep-SE with the transformed SPs as done in the original study. The column "CutTrain" shows the results achieved by applying the transformation only on the training set, while the column "!Cut" shows the results of our replication without transforming the SPs. We also include in this table the results for RQ5 "Deep-SE!pre-train", which investigates Deep-SE without pre-training its lower layers (i.e., word embedding and LSTM). The best results are in bold.

Project	Method	Rep	CutTrain	!Cut	Orig	Project	Method	Rep	CutTrain	!Cut	Orig
MESOS	Deep-SE	1.05	**1.15**	**1.12**	**1.02**	DURACLOUD	Deep-SE	**0.64**	**0.71**	**0.82**	**0.68**
	Deep-SE!pre-train	**1.02**					Deep-SE!pre-train	0.68			
	Mean	1.11	1.22	1.41	1.64		Mean	0.73	0.82	1.00	1.30
	Median	1.11	1.22	1.22	1.73		Median	0.76	0.82	0.82	0.73
USERGRID	Deep-SE	1.06	1.18	1.18	**1.03**	DM	Deep-SE	**3.70**	**5.88**	**5.86**	**3.77**
	Deep-SE!pre-train	1.11					Deep-SE!pre-train	3.91			
	Mean	1.09	1.21	1.19	1.48		Mean	4.89	7.14	8.66	5.29
	Median	**1.03**	**1.15**	**1.15**	1.60		Median	4.28	6.19	6.19	4.82
TISTUD	Deep-SE	1.41	1.43	1.42	**1.36**	MDL	Deep-SE	6.89	6.89	7.89	**5.97**
	Deep-SE!pre-train	1.42					Deep-SE!pre-train	8.05			
	Mean	1.52	1.55	1.91	2.08		Mean	10.19	10.19	12.63	10.90
	Median	**1.28**	**1.30**	**1.30**	1.84		Median	**6.59**	**6.59**	**6.59**	7.18
APSTUD	Deep-SE	3.57	4.09	4.14	**2.71**	MULE	Deep-SE	2.26	2.53	2.59	**2.18**
	Deep-SE!pre-train	3.15					Deep-SE!pre-train	2.30			
	Mean	**2.95**	**3.48**	**3.59**	3.15		Mean	2.22	2.49	2.60	2.59
	Median	3.08	3.61	3.61	3.71		Median	**2.21**	**2.47**	**2.47**	2.69
TIMOB	Deep-SE	2.10	2.19	2.09	**1.97**	MULESTUDIO	Deep-SE	**3.12**	**3.66**	3.67	**3.23**
	Deep-SE!pre-train	2.04					Deep-SE!pre-train	3.24			
	Mean	2.53	2.62	3.02	3.05		Mean	3.22	3.70	3.74	3.34
	Median	**1.94**	**2.04**	**2.04**	2.47		Median	3.18	3.66	**3.66**	3.30
BAM	Deep-SE	0.80	0.80	0.81	**0.74**	XD	Deep-SE	1.66	**1.63**	**1.70**	**1.63**
	Deep-SE!pre-train	0.77					Deep-SE!pre-train	**1.58**			
	Mean	1.03	1.03	1.22	1.75		Mean	1.88	1.91	2.05	2.27
	Median	**0.75**	**0.75**	**0.75**	1.32		Median	1.68	1.71	1.71	2.07
CLOV	Deep-SE	2.46	3.75	**3.39**	**2.11**	TDQ	Deep-SE	**2.88**	**2.90**	3.61	**2.97**
	Deep-SE!pre-train	**2.37**					Deep-SE!pre-train	2.94			
	Mean	2.97	4.26	4.57	3.49		Mean	4.08	4.25	4.56	4.81
	Median	2.42	**3.71**	3.71	2.84		Median	3.15	3.31	**3.31**	3.87
JSWSERVER	Deep-SE	**1.57**	**1.77**	**1.70**	**1.38**	TESB	Deep-SE	**0.61**	**0.85**	**0.90**	**0.64**
	Deep-SE!pre-train	1.58					Deep-SE!pre-train	0.63			
	Mean	1.86	2.07	2.40	2.48		Mean	0.71	1.00	1.04	1.14
	Median	2.10	2.31	2.31	2.93		Median	0.70	0.92	0.92	1.16

Table 4.4: RQ1.1 and RQ5. Results of the Wilcoxon test ($\hat{A}_{12}$ effect size in parentheses) comparing Deep-SE vs baselines (Mean, Median), vs Deep-SE!pre-train, and vs Deep-SE!cut, on the Choet dataset.

Project	Deep-SE vs.			
	Mean	Median	Deep-SE!pre-train	Deep-SE!Cut
MESOS	0.139 (0.52) _	0.182 (0.52) _	0.592 (0.49) _	0.528 (0.50) _
USERGRID	0.440 (0.51) _	0.955 (0.43) _	0.318 (0.52) _	0.419 (0.51) _
TISTUD	0.015 (0.54) N	1.000 (0.39) _	0.734 (0.49) _	0.622 (0.49) _
APSTUD	0.979 (0.44) _	0.960 (0.45) _	0.953 (0.45) _	0.352 (0.51) _
TIMOB	<0.001 (0.60) S	0.999 (0.44) _	0.893 (0.48) _	0.947 (0.47) _
BAM	<0.001 (0.64) S	0.886 (0.45) _	0.617 (0.49) _	0.568 (0.49) _
CLOV	<0.001 (0.68) S	0.063 (0.57) _	0.782 (0.46) _	0.678 (0.48) _
JSWSERVER	0.018 (0.60) S	0.003 (0.63) S	0.368 (0.52) _	0.637 (0.48) _
DURACLOUD	0.012 (0.58) N	0.936 (0.45) _	0.690 (0.48) _	0.134 (0.54) _
DM	<0.001 (0.66) S	0.013 (0.53) N	0.007 (0.53) N	0.019 (0.53) N
MDL	<0.001 (0.75) M	0.517 (0.50) _	<0.001 (0.58) N	<0.001 (0.59) N
MULE	0.572 (0.49) _	0.658 (0.49) _	0.532 (0.50) _	0.160 (0.53) _
MULESTUDIO	0.428 (0.51) _	0.442 (0.50) _	0.362 (0.51) _	0.370 (0.51) _
XD	<0.001 (0.58) N	0.266 (0.51) _	0.991 (0.46) _	0.399 (0.50) _
TDQ	<0.001 (0.70) M	0.009 (0.56) N	0.219 (0.52) _	<0.001 (0.60) S
TESB	<0.001 (0.60) S	0.012 (0.57) N	0.233 (0.52) _	0.019 (0.56) N

Table 4.5: RQs 1.2, 2.2, and 5. Results of Deep-SE, Deep-SE!pre-train, TF-IDF-SE, and baseline estimators (Mean, Median) on the Tawosi dataset. The best results are in bold.

Project	Deep-SE	Deep-SE!pre-train	TF-IDF-SE	Mean	Median
MESOS	**1.34**	1.43	**1.34**	1.37	**1.34**
ALOY	1.51	1.71	**1.44**	2.23	**1.44**
TISTUD	1.63	1.68	**1.51**	2.01	**1.51**
APSTUD	4.31	4.15	**3.99**	4.00	**3.99**
CLI	1.76	**1.58**	2.98	2.14	1.77
DAEMON	3.29	3.00	**2.74**	2.75	**2.74**
TIDOC	**2.72**	3.26	3.03	2.99	2.77
TIMOB	**2.41**	2.49	2.53	2.55	2.53
CLOV	**3.78**	3.89	4.04	5.93	4.01
CONFCLOUD	1.48	1.44	**1.33**	1.49	**1.33**
CONFSERVER	**0.91**	0.96	0.96	1.35	0.96
DNN	0.72	0.72	0.79	0.80	**0.71**
DURACLOUD	0.68	0.74	0.68	**0.67**	0.68
FAB	0.86	0.75	1.10	1.19	**0.67**
STL	1.18	1.20	**0.84**	0.97	0.95
DM	1.61	1.65	**1.49**	2.60	1.61
COMPASS	1.63	1.66	**1.38**	1.48	**1.38**
SERVER	0.89	0.87	0.93	1.56	**0.85**
EVG	0.63	**0.62**	0.69	0.68	0.69
MDL	**3.55**	5.08	6.31	14.54	6.31
MULE	2.24	**2.22**	3.58	2.79	2.24
NEXUS	1.08	**1.05**	1.17	1.11	1.17
XD	1.45	**1.42**	2.01	1.65	1.55
TDP	0.99	**0.98**	0.99	1.17	0.99
TDQ	**2.47**	2.92	5.05	4.20	2.88
TESB	1.15	1.09	**0.97**	0.99	0.98

Table 4.6: RQs 1.2, 2.2, and 5. Results of the Wilcoxon test ($\hat{A}_{12}$ effect size in parentheses) comparing Deep-SE vs Deep-SE!pre-train, Deep-SE vs TF-IDF-SE, Deep-SE vs baseline estimators (Mean, Median), and TF-IDF-SE vs baseline estimators (Mean, Median) on the Tawosi dataset.

Project	Deep-SE vs				TF-IDF-SE vs	
	Deep-SE!pre-train	TF-IDF-SE	Mean	Median	Mean	Median
MESOS	0.242 (0.52) _	0.826 (0.48) _	0.441 (0.50) _	0.826 (0.48) _	0.003 (0.56) N	0.500 (0.50) _
ALOY	0.253 (0.54) _	0.910 (0.42) _	<0.001 (0.69) S	0.910 (0.42) _	<0.001 (0.70) M	0.501 (0.50) _
TISTUD	0.034 (0.53) N	1.000 (0.40) _	<0.001 (0.62) S	1.000 (0.40) _	<0.001 (0.63) S	0.500 (0.50) _
APSTUD	0.645 (0.48) _	0.814 (0.46) _	0.763 (0.47) _	0.814 (0.46) _	0.370 (0.51) _	0.501 (0.50) _
CLI	0.779 (0.46) _	<0.001 (0.74) M	0.016 (0.61) S	0.249 (0.54) _	1.000 (0.31) _	1.000 (0.26) _
DAEMON	0.520 (0.50) _	0.659 (0.47) _	0.618 (0.48) _	0.659 (0.47) _	0.629 (0.48) _	0.502 (0.50) _
TIDOC	0.008 (0.57) N	0.521 (0.50) _	<0.001 (0.60) S	0.141 (0.53) _	<0.001 (0.60) S	0.746 (0.48) _
TIMOB	0.272 (0.51) _	0.303 (0.51) _	0.050 (0.52) _	0.303 (0.51) _	0.032 (0.53) N	0.500 (0.50) _
CLOV	0.070 (0.57) _	0.817 (0.46) _	<0.001 (0.85) L	0.026 (0.60) S	<0.001 (0.81) L	0.030 (0.59) N
CONFCLOUD	0.427 (0.51) _	0.822 (0.45) _	0.497 (0.50) _	0.822 (0.45) _	0.029 (0.61) S	0.501 (0.50) _
CONFSERVER	0.359 (0.52) _	0.869 (0.45) _	<0.001 (0.65) S	0.869 (0.45) _	<0.001 (0.65) S	0.501 (0.50) _
DNN	0.642 (0.49) _	0.848 (0.48) _	<0.001 (0.57) N	0.370 (0.51) _	<0.001 (0.67) S	0.614 (0.49) _
DURACLOUD	0.341 (0.52) _	0.949 (0.42) _	0.125 (0.56) _	0.949 (0.42) _	0.181 (0.55) _	0.501 (0.50) _
FAB	0.882 (0.44) _	0.220 (0.54) _	0.013 (0.61) S	0.995 (0.37) _	<0.001 (0.68) S	0.997 (0.37) _
STL	0.405 (0.52) _	0.997 (0.33) _	0.857 (0.43) _	0.892 (0.42) _	<0.001 (0.70) M	0.050 (0.59) _
DM	0.684 (0.49) _	1.000 (0.39) _	<0.001 (0.80) L	0.687 (0.49) _	<0.001 (0.83) L	<0.001 (0.58) N
COMPASS	0.331 (0.52) _	0.898 (0.43) _	0.665 (0.48) _	0.898 (0.43) _	0.229 (0.54) _	0.501 (0.50) _
SERVER	0.844 (0.46) _	0.593 (0.49) _	<0.001 (0.75) M	0.322 (0.52) _	<0.001 (0.80) L	0.483 (0.50) _
EVG	0.767 (0.49) _	0.025 (0.53) N	0.019 (0.54) N	0.025 (0.53) N	0.989 (0.46) _	0.500 (0.50) _
MDL	<0.001 (0.67) S	<0.001 (0.83) L	<0.001 (1.00) L	<0.001 (0.83) L	<0.001 (1.00) L	0.500 (0.50) _
MULE	0.564 (0.50) _	<0.001 (0.61) S	<0.001 (0.65) S	0.079 (0.52) _	0.674 (0.49) _	1.000 (0.39) _
NEXUS	0.853 (0.47) _	0.847 (0.48) _	0.539 (0.50) _	0.847 (0.48) _	0.573 (0.50) _	0.500 (0.50) _
XD	0.800 (0.47) _	0.055 (0.55) _	0.014 (0.57) N	0.691 (0.48) _	0.218 (0.52) _	0.990 (0.43) _
TDP	0.735 (0.47) _	0.355 (0.52) _	0.012 (0.59) N	0.355 (0.52) _	0.008 (0.60) S	0.501 (0.50) _
TDQ	0.003 (0.59) N	<0.001 (0.81) L	<0.001 (0.77) M	0.006 (0.58) N	1.000 (0.35) _	1.000 (0.21) _
TESB	0.640 (0.49) _	0.989 (0.42) _	0.961 (0.44) _	0.976 (0.43) _	0.002 (0.59) N	0.406 (0.51) _

These results contradict those reported in the original study, where Deep-SE was found to be significantly better than TF-IDF-SE for all eight projects investigated.

For completeness, although not included in the original work, we also compare the results of Deep-SE and TF-IDF-SE on the Porru dataset to the Mean and Median baseline estimators. We can observe that there is a difference in the results of the original study and those we obtain in our replication for all eight projects considered. When compared to the baseline estimators, Deep-SE(Rep) outperforms Mean in seven out of eight (88%) cases with the differences in four of them being statistically significant and the effect size being negligible in one case, small in one case and large in two other cases. Deep-SE obtains better MAEs than Median in four projects (50%), out of which two show statistical significance with a small effect size. When we consider the Bonferroni correction, the number of cases where the difference with Mean remains significant drops to three (one with small and two with large effect size). The same observation holds instead for the comparison with the Median estimator under the Bonferroni correction.

4.4.2.2. RQ2.2 - Extension

Table 4.5 shows the results obtained by Deep-SE and TF-IDF-SE on the Tawosi dataset.

We observe that Deep-SE performs better than TF-IDF-SE on 14 out of 26 projects (54%) and worse on nine projects (34%). Both perform equally well on the remaining three projects (12%). However, in the cases where Deep-SE outperforms TF-IDF-SE, the difference in absolute errors is statistically significant in only five out of 14 cases (19% of all cases). Among those, the effect size is large in two cases, medium in one, small in one and negligible in one (see Table 4.6). For the cases where Deep-SE performs worse than TF-IDF-SE, the difference in absolute errors is significant in four out of nine cases (15%), one having a negligible effect size and the remaining three having a small one.

Overall, these results strengthen our confidence in the conclusion of RQ2.1 by confirming a small difference between the results of these two techniques.

Answer to RQ2: *Deep-SE statistically significantly outperforms TF-IDF-SE in only five out of 26 cases (19%), whereas it provides statistically significantly worse results in four cases (15%). Therefore, we cannot conclude that Deep-SE outperforms TF-IDF-SE for all projects herein.*

Table 4.7: RQ2.1. Results of the Deep-SE and TF-IDF-SE replication (Rep), original study [26] (Orig), and the baselines on the Porru dataset. Best results (among all methods but Deep-SE (Orig) and TF-IDF-SE (Orig)) in bold.

Project	Deep-SE (Rep)	TF-IDF-SE (Rep)	Mean	Median	Deep-SE (Orig)	TF-IDF-SE (Orig)
TIMOB	7.36	**1.76**	20.08	**1.76**	1.44	1.76
TISTUD	1.36	**1.28**	1.87	**1.28**	1.04	1.28
APSTUD	**5.52**	5.69	5.59	5.69	2.67	5.69
MESOS	1.08	1.23	1.24	**0.84**	0.76	1.23
MULE	3.27	3.37	3.22	**3.07**	2.32	3.37
XD	**1.23**	1.86	1.24	1.34	1.00	1.86
DNN	**0.69**	1.08	0.72	1.08	0.47	1.08
NEXUS	**0.30**	0.39	0.72	0.39	0.21	0.39

Table 4.8: RQ2.1. Results of the Wilcoxon test ($\hat{A}_{12}$ effect size in parentheses) comparing Deep-SE vs TF-IDF-SE, Deep-SE vs baselines (Mean, Median), and TF-IDF-SE vs baselines (Mean, Median) on the Porru dataset.

Project	Deep-SE vs.			TF-IDF-SE vs.	
	TF-IDF-SE	Mean	Median	Mean	Median
TIMOB	0.968 (0.43) _	<0.001 (0.99) L	0.968 (0.43) _	<0.001 (1.00) L	0.500 (0.50) _
TISTUD	1.000 (0.38) _	<0.001 (0.67) S	1.000 (0.38) _	<0.001 (0.64) S	0.500 (0.50) _
APSTUD	0.366 (0.52) _	0.413 (0.52) _	0.366 (0.52) _	0.666 (0.48) _	0.501 (0.50) _
MESOS	0.486 (0.50) _	0.087 (0.57) _	0.996 (0.37) _	<0.001 (0.66) S	0.989 (0.40) _
MULE	0.043 (0.55) S	0.039 (0.56) N	0.080 (0.55) _	0.990 (0.43) _	0.997 (0.41) _
XD	0.005 (0.61) S	0.280 (0.53) _	0.238 (0.53) _	0.987 (0.41) _	0.995 (0.39) _
DNN	<0.001 (0.62) S	0.150 (0.53) _	<0.001 (0.62) S	1.000 (0.38) _	0.500 (0.50) _
NEXUS	<0.001 (0.63) S	<0.001 (0.91) L	<0.001 (0.64) S	<0.001 (0.76) M	0.501 (0.50) _

4.4.3. RQ3. Cross-project Estimation

4.4.3.1. RQ3.1 - Replication

Following the original study, we performed two different types of experiments in order to investigate the effectiveness of Deep-SE for cross-project SP estimation.

The first experiment is characterised by the fact that Deep-SE is trained by using SP contained only in projects belonging to the same repository of the target project (i.e., within-repository; for example, Deep-SE trained on MESOS and tested on USERGRID, where both belong to Apache repository), while in the second experiment, we train Deep-SE by using SPs contained only in projects belonging to a different repository (i.e., cross-repository; for example Deep-SE trained on MESOS from Apache repository and tested on MULE, which belongs to MuleSoft repository).

We observe that the results we obtained by using Deep-SE both for the within-repository and cross-repository experiments are different from those recorded in the original study. Specifically, the MAE values obtained in our within-repository replication are higher (worse) than those reported in the original study in seven out of the eight (88%) projects (see Table 4.9a). Whereas the MAE values we obtain in the cross-repository replication are higher than those reported in the original study only for four out of eight (50%) projects (see Table 4.9b). The difference between Rep and Orig might have two reasons. The first might be the use of the SP transformation in the original study, as further explained in Section 4.5. The second reason can be the train-validation-test split rates, as two projects are involved in train, validation, and testing, and the original study did not mention how they split the data for this RQ. We kept the same splitting ratio for train and validation as explained in Section 4.3.2.4.

When we compare the results obtained by Deep-SE for cross-project estimation when it is trained with within-repository data (Table 4.9a) vs training it with cross-repository data (Table 4.9b), we can observe that the latter (i.e. training Deep-SE with data from projects belonging to the same repository of the target project) generally provides lower MAE values. Specifically, the MAE values of Deep-SE (Rep) within-repository are lower than those of Deep-SE (Rep) cross-repository in six out of eight cases, whereas, in the remaining two, they are very close. These results corroborate the observation made in the original study, where the within-repository training was found more beneficial than the cross-repository for all cases (i.e., Deep-SE (Orig) within-repository provides always lower MAE than Deep-SE (Orig) cross-repository). This can be explained by the fact that different organisations may apply different policies for SP estimation [18].

While the original study did not benchmark Deep-SE with the Mean and Median baselines for this RQ, we believe these are, as well, necessary benchmarks for this scenario. The results of this comparison are reported in Tables 4.9a and 4.9b. In the within-repository scenario, Deep-SE performs better than the Mean estimator in four out of the eight cases (50%) studied with statistically significant differences with a negligible effect size in three and a small one in one case (see the last column of Table 4.9a). Whereas, Deep-SE outperforms Median in three cases (38%), with the difference being statistically significant in two of these cases and the effect size being small for one and negligible for the other. In the cross-repository training scenario (Table 4.9b), Deep-SE statistically significantly outperforms the Mean estimator in six out of eight cases (75%), with a large effect size in three of them, medium in one, and small in two cases. This is due to the different distributions of the SP values in the source and target projects. The difference between the MAE values achieved by the Mean estimator and those of Deep-SE are higher for the cases in which the difference in the mean SP values of the source and target projects is larger (see Table 4.2). While compared to the Median estimator, our results show that Deep-SE obtains better MAE values in two cases out of eight (25%), with the difference being statistically significant and the effect size being small in both cases. If we consider the Bonferroni correction, these observations still hold. Overall, Deep-SE outperforms the Mean and Median estimators for both scenarios, with a statistically significant difference in 10 and 4 out of 16 cases (63% and 25%), respectively.

Finally, we comment on the use of Deep-SE for within-project estimation versus its use for cross-project estimation by observing that, overall, Deep-SE is more effective for the former. In fact, the estimation accuracy achieved by Deep-SE on the same set of target projects (i.e., USERGRID, MESOS, APSTUD, TIMOB, TISTUD, MULESTUDIO, MULE when trained based on within-project – see Table 4.3) is always higher than the accuracy of Deep-SE when trained with cross-project with both within- or cross-repository (see Table 4.9). This confirms the original study's findings: Deep-SE is more suitable for within-project estimation than for cross-project estimation.

4.4.3.2. RQ3.2 - Extension

For this research question, we exploit issues from projects belonging to the same repository as the target project to form a training/validation set. Therefore, we can only use those repositories in the Tawosi dataset that contain more than one project (i.e., Appcelerator, Atlassian, Hyperledger, MongoDB, and Talendforge). These repositories contain 18 projects in total. For each of these 18 projects (as target projects), we combine all the issues from all the other projects belonging to the same repository to form the training/validation set (as the training source). We then remove the issues created after the target project's start date from the training/validation set. For 13 out of 18 target projects, this leaves us with a source issue set containing less than 200 issues. We remove these target projects from our experiments, as previous work showed training with less than 200 issues might not result in a stable model [23]. Thus, we investigate five projects for this research question: ALOY with 1,620 issues as the training source, CLI with 3,383 issues, DAEMON with 5,587 issues, TIDOC with 335 issues, and TDP with 1,410 issues. We split these source issue sets into 75%-25% train-validation sets and use them for training Deep-SE. The resulting model is used to estimate all the issues in the target projects.

Table 4.10 reports the results of our experiments. Based on the MAE, we can observe that Deep-SE always provides better results than the Mean estimator but with a statistically significant difference only in three of them. Whereas, Deep-SE performs worse than the Median estimator in all five cases, with the difference being statistically significant in two cases (TIDOC and TDP) and the effect size being negligible. If we consider the Bonferroni correction, the same observations hold.

These results corroborate the conclusion drawn in RQ3.1: Deep-SE does not always provide more accurate cross-project SP estimations than baseline estimators. Moreover, our results for RQ3.2 highlight that when Deep-SE is used in a realistic scenario, taking into account the chronological order of the issues, its prediction performance worsens.

Answer to RQ3: *Deep-SE is less effective for cross-project SP estimation with respect to within-project SP estimation.*

4.4.4. RQ4. Augmented Training Set

To augment the training sets used in RQ1.2, we exploit issues from projects belonging to the same repository as the target project in the Tawosi dataset. Similarly to RQ3.2, this limits us to 18 projects in the Appcelerator, Atlassian, Hyperledger, MongoDB, and Talendforge repositories.

Applying the augmentation resulted in an increase in the size of the training set for all 18 target projects, ranging between 63.2 times for TIDOC and 1.4 times for SERVER (10.5 times on average over 18 projects). This experiment mimics a real-world scenario in which a company uses issues from all its projects (past and present) to train Deep-SE and use it for story point estimation of new issues.

Note that this is different from RQ3.2, where Deep-SE did not use any issues from the target project for training, while herein, we keep the original 60%-20% train-validation sets from the target project and only augment the training set with available issues from other projects[11]. This is to examine whether Deep-SE's estimation performance will increase if more training data become available to it.

Table 4.11 shows the results we obtained after augmenting the training set for 18 projects (indicated by "AUG" as the column header). This table also shows the results of Deep-SE estimating SP for the same projects by only using within-project issues (under the WP column). We also report the results of the baseline estimators to verify whether

[11]Note that we only include those issues from other projects that are resolved before the creation of the first issue in the test set.

Table 4.9: RQ3.1. Comparing Deep-SE cross-project ((a) within-repository and (b) cross-repository) SP estimation replication results (Rep) to the original study results (Orig) [26], and to the baselines. The results of the Wilcoxon test ($\hat{A}_{12}$ effect size in parentheses) for Deep-SE (Rep) vs Mean and Median baselines are shown in the last column. The best results (among all approaches but Deep-SE (Orig)) per project are highlighted in bold.

(a) Within-Repository Training					(b) Cross-Repository Training				
Source	Target	Method	MAE	Deep-SE (Rep) vs.	Source	Target	Method	MAE	Deep-SE (Rep) vs.
		Deep-SE (Orig)	1.07				Deep-SE (Orig)	1.57	
MESOS	USERGRID	Deep-SE (Rep)	1.16		TISTUD	USERGRID	Deep-SE (Rep)	3.47	
(ME)	(UG)	Mean	1.02	1.000 (0.42) _	(AS)	(UG)	Mean	3.08	1.000 (0.42) _
		Median	**0.89**	1.000 (0.37) _			Median	**2.30**	1.000 (0.27) _
		Deep-SE (Orig)	1.14				Deep-SE (Orig)	2.08	
USERGRID	MESOS	Deep-SE (Rep)	1.51		TISTUD	MESOS	Deep-SE (Rep)	3.18	
(UG)	(ME)	Mean	1.52	0.282 (0.51) _	(AS)	(ME)	Mean	3.28	0.011 (0.52) N
		Median	**1.50**	0.802 (0.49) _			Median	**2.58**	1.000 (0.39) _
		Deep-SE (Orig)	2.75				Deep-SE (Orig)	5.37	
TISTUD	APSTUD	Deep-SE (Rep)	4.37		MDL	APSTUD	Deep-SE (Rep)	5.03	
(AS)	(AP)	Mean	**4.27**	0.918 (0.48) _	(MD)	(AP)	Mean	9.84	<0.001 (0.81) L
		Median	4.38	0.573 (0.50) _			Median	**3.97**	1.000 (0.43) _
		Deep-SE (Orig)	1.99				Deep-SE (Orig)	6.36	
TISTUD	TIMOB	Deep-SE (Rep)	3.38		MDL	TIMOB	Deep-SE	**3.34**	
(AS)	(TI)	Mean	3.45	<0.001 (0.54) N	(MD)	(TI)	Mean	11.19	<0.001 (0.92) L
		Median	**3.17**	1.000 (0.45) _			Median	4.19	<0.001 (0.63) S
		Deep-SE (Orig)	2.85				Deep-SE (Orig)	5.55	
APSTUD	TISTUD	Deep-SE (Rep)	**2.70**		MDL	TISTUD	Deep-SE (Rep)	**2.64**	
(AP)	(AS)	Mean	3.38	<0.001 (0.59) N	(MD)	(AS)	Mean	11.45	<0.001 (0.97) L
		Median	3.17	<0.001 (0.56) N			Median	3.17	<0.001 (0.58) N
		Deep-SE (Orig)	3.41				Deep-SE (Orig)	2.67	
APSTUD	TIMOB	Deep-SE (Rep)	**3.51**		DM	TIMOB	Deep-SE (Rep)	3.81	
(AP)	(TI)	Mean	4.36	<0.001 (0.64) S	(DM)	(TI)	Mean	5.61	<0.001 (0.72) M
		Median	4.19	<0.001 (0.62) S			Median	**3.46**	1.000 (0.45) _
		Deep-SE (Orig)	3.14				Deep-SE (Orig)	4.24	
MULE	MULESTUDIO	Deep-SE (Rep)	3.64		USERGRID	MULESTUDIO	Deep-SE (Rep)	3.95	
(MU)	(MS)	Mean	3.34	0.775 (0.49) _	(UG)	(MS)	Mean	4.04	0.008 (0.54) N
		Median	**3.26**	0.997 (0.46) _			Median	**3.91**	0.917 (0.48) _
		Deep-SE (Orig)	2.31				Deep-SE (Orig)	2.70	
MULESTUDIO	MULE	Deep-SE (Rep)	2.77		MESOS	MULE	Deep-SE (Rep)	3.20	
(MS)	(MU)	Mean	3.05	0.004 (0.54) N	(ME)	(MU)	Mean	**2.89**	0.999 (0.46) _
		Median	**2.60**	0.997 (0.46) _			Median	2.92	1.000 (0.45) _

Table 4.10: RQ3.2. Comparing the cross-project prediction accuracy (in terms of MAE) of Deep-SE and the baselines Mean and Median. The last column shows the result of the Wilcoxon statistical test ($\hat{A}_{12}$ Effect size in parentheses) for Deep-SE. The best results are in bold.

Source	Target	Method	MAE	Deep-SE vs
APSTUD, TIDOC, TIMOB, TISTUD	ALOY	Deep-SE	2.15	
		Mean	2.83	<0.001 (0.67) S
		Median	**2.10**	0.770 (0.48) _
ALOY, APSTUD, TIDOC, TIMOB, TISTUD	CLI	Deep-SE	2.72	
		Mean	2.80	0.089 (0.53) _
		Median	**2.38**	0.940 (0.46) _
ALOY, CLI, APSTUD, TIDOC, TIMOB, TISTUD	DAEMON	Deep-SE	2.95	
		Mean	3.04	0.166 (0.53) _
		Median	**2.89**	0.526 (0.50) _
APSTUD, TIMOB, TISTUD	TIDOC	Deep-SE	2.53	
		Mean	2.91	<0.001 (0.64) S
		Median	**2.45**	0.006 (0.53) N
TDQ, TESB	TDP	Deep-SE	1.60	
		Mean	2.46	<0.001 (0.75) M
		Median	**1.53**	0.021 (0.54) N

augmenting the training set helps Deep-SE outperform the baselines on projects where Mean and Median had previously performed better in within-project estimation. Results show that Deep-SE performs better on only five out of 18 projects (28%) when the training set is augmented. On the other hand, its performance deteriorates on 12 projects (67%) and remains the same on a single project (CONFCLOUD).

We looked for common features among the projects that benefited from the augmentation or those which did not, but we could not find any emerging pattern. Specifically, we looked into their application domain, project size, amount of augmented data with respect to the project's original size, and the respective repository.

The analysis of the Wilcoxon test on Deep-SE for within-project vs augmentation showed that the difference in the distribution of the errors produced by the two variants is statistically significant in five projects (ALOY, TIDOC, TISTUD, EVG, TDQ) with a medium effect size in ALOY and negligible with the others.

Deep-SE's performance with respect to Mean and Median remains almost unchanged when the augmented data is used. In fact, on the one hand, Deep-SE (AUG) outperforms the baselines on five projects (i.e., ALOY, COMPASS, SERVER, TDP, and TESB), while Deep-SE (WP) was outperformed by the baselines on these five projects (see RQ2.2). However, on the other hand, the performance of Deep-SE (AUG) becomes worse than that of the baseline techniques on four projects (i.e., TISTUD, CONFSERVER, EVG, TDQ), while Deep-SE (WP) performs better than the baselines on these same four projects (see RQ2.2).

Answer to RQ4: *Augmenting the training set with issues from within-company projects had no steady positive or negative effect on Deep-SE's accuracy.*

4.4.5. RQ5. Pre-Training Effectiveness

Tables 4.3 and 4.5 show the performance of Deep-SE with and without pre-training (Deep-SE vs Deep-SE!pre-train) on the Choet and the Tawosi datasets, respectively. We can observe that pre-training does not always improve Deep-SE's estimation accuracy. Specifically, Deep-SE with random initialization (i.e., Deep-SE!pre-train) obtained better MAE values, although with a small difference, for six out of 16 (38%) projects from the Choet dataset (TIMOB, APSTUD, BAM, CLOV, MESOS, XD).

The results of the Wilcoxon test (Table 4.4) show that the difference in the estimation accuracy of Deep-SE with and without pre-training is statistically significant (in favour of pre-training) for two of the projects (DM and MDL) with a negligible effect size.[12] On the Tawosi dataset (Table 4.5), Deep-SE without pre-training performs similarly or better than Deep-SE in 13 out of 26 projects (50%), among which the difference in errors produced by the two variants is statistically significant on four projects only (TISTUD, TIDOC, MDL, TDQ) with a negligible/small effect size (Table 4.6).

Having seen the trivial effect of pre-training in improving the accuracy of Deep-SE for SP estimation, we checked whether it at least helps Deep-SE converge faster to the best solution. To this end, we compare the running time and the number of epochs (Table 4.12) required by the two variants to converge to the best solution on the validation set before the early stopping criterion is met (i.e., ten consecutive epochs with no improvements in the validation loss function).

We can observe that the running time of Deep-SE with random initialization (i.e., Deep-SE!pre-train) is slightly smaller than its pre-trained variant in 33 out of the 42 cases (79%) studied. When checking for statistical differences between the distribution of the running time of the two variants, the results of the Wilcoxon test show $p-value =$0.956 for the Choet dataset and $p-value =$0.791 for the Tawosi dataset, indicating that we cannot accept the alternative hypothesis that these results are statically significant different. Similarly, the comparison between the number of epochs does not reveal any significant difference ($p-value = 0.830$), although on average, Deep-SE without pre-training took slightly fewer epochs to converge (see Table 4.12).

Overall, the results show that using the pre-trained embedding weights and LSTM layer does not enhance Deep-SE's accuracy or convergence speed; thus, it can be

[12] Note that on the other hand, the difference in the estimation accuracy is statistically significant in favour of random initialization (without pre-training) for two projects (APSTUD and XD), however, with a negligible effect size.

Table 4.11: RQ4. Results achieved by Deep-SE on the Tawosi dataset when the training set is augmented by using older issues from the repository that the project belongs to (AUG), compared to Deep-SE's within-project results from RQ1.2 (WP) and to baseline estimators. The best results are in bold.

Project	Method	MAE		Project	Method	MAE	
		AUG	WP			AUG	WP
ALOY	Deep-SE	**2.59**	1.51	CONFSERVER	Deep-SE	1.04	**0.91**
	Mean	3.13	2.23		Mean	1.95	1.35
	Median	2.80	**1.44**		Median	**0.96**	0.96
TISTUD	Deep-SE	1.73	1.63	FAB	Deep-SE	0.87	0.86
	Mean	1.91	2.01		Mean	1.00	1.19
	Median	**1.51**	**1.51**		Median	**0.67**	**0.67**
APSTUD	Deep-SE	4.49	4.31	STL	Deep-SE	1.10	1.18
	Mean	**4.02**	4.00		Mean	1.24	0.97
	Median	4.02	**3.99**		Median	**0.95**	**0.95**
CLI	Deep-SE	**2.04**	**1.76**	COMPASS	Deep-SE	**1.43**	1.63
	Mean	3.19	2.14		Mean	1.89	1.48
	Median	2.98	1.77		Median	1.81	**1.38**
DAEMON	Deep-SE	3.10	3.29	SERVER	Deep-SE	**0.83**	0.89
	Mean	2.76	2.75		Mean	0.85	1.56
	Median	**2.74**	**2.74**		Median	0.85	**0.85**
TIDOC	Deep-SE	**3.35**	**2.72**	EVG	Deep-SE	0.68	**0.63**
	Mean	3.47	2.99		Mean	0.66	0.68
	Median	3.42	2.77		Median	**0.65**	0.69
TIMOB	Deep-SE	**2.46**	**2.41**	TDP	Deep-SE	**1.07**	**0.99**
	Mean	2.62	2.55		Mean	2.27	1.17
	Median	2.53	2.53		Median	1.47	**0.99**
CLOV	Deep-SE	**3.71**	**3.78**	TDQ	Deep-SE	2.83	**2.47**
	Mean	5.35	5.93		Mean	2.82	4.20
	Median	4.01	4.01		Median	**2.22**	2.88
CONFCLOUD	Deep-SE	1.48	1.48	TESB	Deep-SE	**1.19**	1.15
	Mean	2.32	1.49		Mean	2.52	0.99
	Median	**1.33**	**1.33**		Median	1.29	**0.98**

skipped to save the high cost required by such a pre-training procedure.

Answer to RQ5: *Pre-training the lower layer of Deep-SE with user stories from other projects has a negligible effect in improving its SP estimation accuracy or convergence speed; thus, it can be skipped.*

4.5. DISCUSSION

In some cases, the replication of the work by Choetkiertikul et al. [26] provided different results from the original study. In RQ1.1, we found that the results achieved for Deep-SE and the baselines differ from that of the original study. Moreover, in RQ2.1, although the results obtained in this study for TF-IDF-SE matched those reported in the original study, we found differences in the results of Deep-SE between the two studies. Finally, in RQ3.1, we observed differences in the results of Deep-SE between the replication and the original study. However, they support the same conclusion as the one made in the original study.

In order to investigate the causes of the discrepancies: (i) an independent reviewer checked the modifications and fixes performed on the original source code and results to make sure it is correct, (ii) checked the source code and results included in the replication package provided by the authors [101]; (iii) contacted the authors of the original study. In the following, the possible reasons for observing different results are discussed.

First of all, we notice a discrepancy in the results reported for the baseline estimators, namely Mean and Median. These naïve baselines are deterministic, and therefore we do expect them to achieve the same results when applied to the same data. However, this was not the case, and further investigation suggests that there may have been a misuse of these estimators in the original study. Indeed, the results of the original study are reproducible only by adding one (1.0) to the Mean and Median story point estimates. Since this addition modifies the original estimates of the baselines, it leads to an incorrect absolute error computation on the test set. This modification of the original estimates is not described or justified in the original paper nor in the more general effort estimation literature. As such, we conclude that it should not be used. Therefore, for all the RQs investigated in this study, we report the correct results for the Mean and Median estimators based on the definition given in the literature as described in Section 4.3.4.

By investigating the source code made publicly available by the authors [101], we also found a discrepancy between the description of Deep-SE provided in the paper and its implementation as included in their replication package. In fact, the Python implementation for Deep-SE applies a transformation on the SP distribution before using it for training, validation, and testing. Through this transformation, all the original story points falling above a certain threshold (i.e., 90^{th} percentile of the distribution)

Table 4.12: RQ5. Comparing running time and the number of epochs that each of the methods (i.e., Deep-SE when used with initialization of weights through pre-training (Deep-SE) and Deep-SE with random initialization (Deep-SE!pre-train)) needs to converge on the (a) Choet and (b) Tawosi datasets. The best results are in bold.

(a) Choet dataset

Project	Running Time (Seconds)		Epoch	
	Deep-SE	Deep-SE!pre-train	Deep-SE	Deep-SE!pre-train
MESOS	757	**702**	115	**109**
USERGRID	**240**	260	**114**	129
TISTUD	**1,212**	1,268	**109**	111
APSTUD	464	**363**	133	**110**
TIMOB	950	**905**	109	**106**
BAM	243	**243**	**107**	108
CLOV	210	**204**	123	**115**
JSWSERVER	**186**	279	**113**	144
DURACLOUD	329	**320**	120	**117**
DM	2,128	**1,973**	119	**110**
MDL	566	**522**	121	**116**
MULE	429	**415**	117	**111**
MULESTUDIO	320	**310**	110	**107**
XD	**1,433**	1,475	**108**	109
TDQ	673	**629**	**111**	112
TESB	**435**	452	124	**123**
Total	10,583	**10,326**	1,853	**1,837**
Mean	-	-	115.81	**114.81**

(b) Tawosi dataset

Project	Running Time (Seconds)		Epoch	
	Deep-SE	Deep-SE!pre-train	Deep-SE	Deep-SE!pre-train
MESOS	844	**750**	**111**	116
ALOY	**144**	159	**111**	119
TISTUD	1,641	**1,376**	**106**	**106**
APSTUD	317	**251**	122	**111**
CLI	226	**164**	130	**108**
DAEMON	198	**149**	131	**128**
TIDOC	601	**486**	**108**	115
TIMOB	2,178	**1,793**	**110**	112
CLOV	239	**217**	125	**111**
CONFCLOUD	179	**150**	117	**108**
CONFSERVER	303	**261**	112	**109**
DNN	1,193	**892**	117	**106**
DURACLOUD	272	**217**	**137**	147
FAB	214	**172**	118	**107**
STL	161	**133**	116	**115**
DM	2,942	**2,711**	**110**	**110**
COMPASS	259	**166**	171	**117**
SERVER	**276**	307	**107**	129
EVG	2,421	**1,967**	**173**	175
MDL	822	**719**	**112**	126
MULE	1,586	**1,495**	**111**	114
NEXUS	**604**	626	**106**	**106**
XD	466	**380**	110	**109**
TDP	312	**239**	**110**	**110**
TDQ	**409**	434	**115**	117
TESB	466	**350**	119	**110**
Total	19,273	**16,564**	3,115	**3,041**
Mean	-	-	119.81	**116.96**

are replaced with a smaller cap value (specifically, with the value falling in the 90^{th} percentile). With this transformation in place, for example, in the DM project, 319 issues with SP values between 25 and 100 are set to 21 in the training set. The same happens for 68 issues from the test set.

The following caveats arise using such a transformation: (i) it is not mentioned in the original paper [26], which can render future replications and/or adoption of the method non-consistent or unreliable; (ii) the replication results (RQ1) show that without applying this pre-processing step, an estimation technique tends to perform generally worse. (iii) the current source code applies the pre-processing step to the entire dataset (i.e., before splitting it into training, validation, and test sets). However, while applying such a transformation on the training set might be acceptable, it should not be applied on the test set since it can undermine the validation process by inflating its accuracy, given that in real-world settings, the test set is unknown and such a transformation cannot be applied at prediction time. The corresponding author of the original study explained to us that they performed this pre-processing to investigate the effect of eliminating outliers from the training set, but it was not their intention to apply it to the test set.

Therefore, we conclude that using the code as provided in the replication package might lead users, unaware of the application of such a transformation in the code, to unjustly/unfairly compare Deep-SE with other approaches which do not include this pre-processing step.

This may be the case of recent work by Abadeer and Sabetzadeh [63] that used Deep-SE with a dataset of industrial projects. Adopting the Deep-SE's implementation, as provided in the replication package, will automatically apply the transformation to the entire dataset before training, validation, and testing. It will replace all those SPs with a value of 13 (4% of the issues) with a value of 8. Therefore, Deep-SE may produce a lower (better) MAE than the baselines in case they are used on non-transformed data. The same issue appears in Fu and Tantithamthavorn's work [21], where they compare their proposed model to Deep-SE.

Based on the above observations, for replication purposes, we reported the result of Deep-SE both with and without transformed SPs for RQ1.1 and with transformation for RQ5. Whereas, for RQ1.2, RQ2, RQ3, and RQ4, we report the results of Deep-SE without transformation.

4.6. THREATS TO VALIDITY

Our study is a close replication and extension of previous work on task-level software effort estimation[26]. Thus, it shares some of the threats to the validity of the original study but also has implemented further mitigation. We followed best practices for designing the replication and reporting findings in a sound way [110]–[112], [136], [137].

In the original study, the authors expressed their concern about basing their ground

Table 4.13: Semantically related user stories with different SP values for Spring XD. Related concepts are highlighted in bold.

Issue Key	Type	Title	Description	Story Point
XD-2347	Technical Task	**Document Kafka message bus**	As a user, I'd like to refer to **document**ation in wiki so that I can setup and **configure Kafka** as a **message bus** as recommended.	2
XD-2361	Story	Pre-allocate **partition**s for **Kafka message bus**	As a user, I want Spring XD's **message bus** to be able to pre-allocate **partition**s between nodes when a stream is deployed, so that rebalancing doesn't happen when a container crashes and/or it's redeployed.	8
XD-3164	Story	**Kafka bus** defaults **configurable** at producer/consumer level	As a developer, I want to be able to override **Kafka bus** defaults for module consumers and producers, so that I can finely tune performance and behaviour. Such **properties** should include -autoCommitEnabled, queueSize, **maxWait**, fetchSize for consumers- batchSize, batchTimeout for producers.	3
XD-3516	Story	**Document partition**ing through deployment **properties**	As an s-c-d user, I'd like to have **document**ation on deployment manifest, so I could refer to the relevant bits on **partition**s. I'd like to understand how stream with e...	2
XD-3740	Bug	**Kafka message bus maxWait property** is not set up	The **maxWait property** from server.yml in the **message bus** section for **kafka** is not propagated through the code, it is ignored.	1

truth on the most likely biased human-estimated story points [30]. However, they argued that Deep-SE is trained to imitate human beings with respect to estimations by reproducing an estimate that human engineers would be able to derive. For this aim, story points sufficiently serve the purpose. Moreover, whenever an unbiased ground truth is available, Deep-SE can be trained on the new target variable.

To minimize threats to conclusion validity, the original study, and similarly, our study, used unbiased accuracy performance measures and applied statistical tests by checking all the required assumptions. Besides, in our study, we checked by performing a peer-code-review that the implementation of all estimation approaches and the computation of the accuracy measures adhere to their original definitions.

To mitigate the external validity, like the original study, we collected data from real-world open-source projects to evaluate the methods. Although these projects differ greatly in size, complexity, and developer community, we cannot claim that they are representative of all kinds of software projects. Especially there are differences between open-source and commercial software projects. A key difference that may affect the estimation of story points is the behaviour of contributors, developers, and the project's stakeholders. It is expected that in an industrial setting for a commercial software project, the user stories are written in a more cohesive and disciplined manner, thus, providing more useful information and containing less noise. Hence, further investigation of commercial projects from industrial software companies is needed to strengthen the conclusions made in this study.

Santos et al. [112] showed that sampling error and (un)intentional contextual modifications in experimental settings can produce different results, leading to an unsuccessful reproduction of results. To avoid such errors and modifications, we made sure that for the replicated RQs, we use the same data and train-validation-test as the ones used in the original study. We also made sure that all the projects have more than 200 observations, thus mitigating the threat of obtaining unstable p-values and effect sizes due to undersampling [112]. To ensure that we keep the experimental setting consistent with the original study, we followed the procedures outlined in their paper. Furthermore, for both Deep-SE and TF-IDF-SE, we used the implementation provided by Choetkiertikul et al. [26]. The only changes we made are to amend and fix errors found in the implementation of the baseline techniques and evaluation metrics according to the definition recommended in the literature (see Section 4.5).

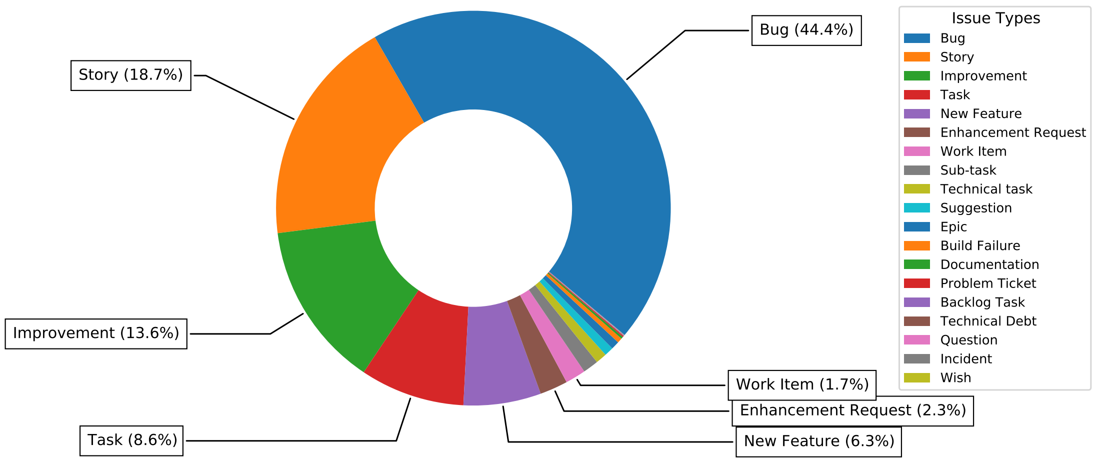
Our scripts and dataset are publicly available at [123].

4.7. CONCLUSION

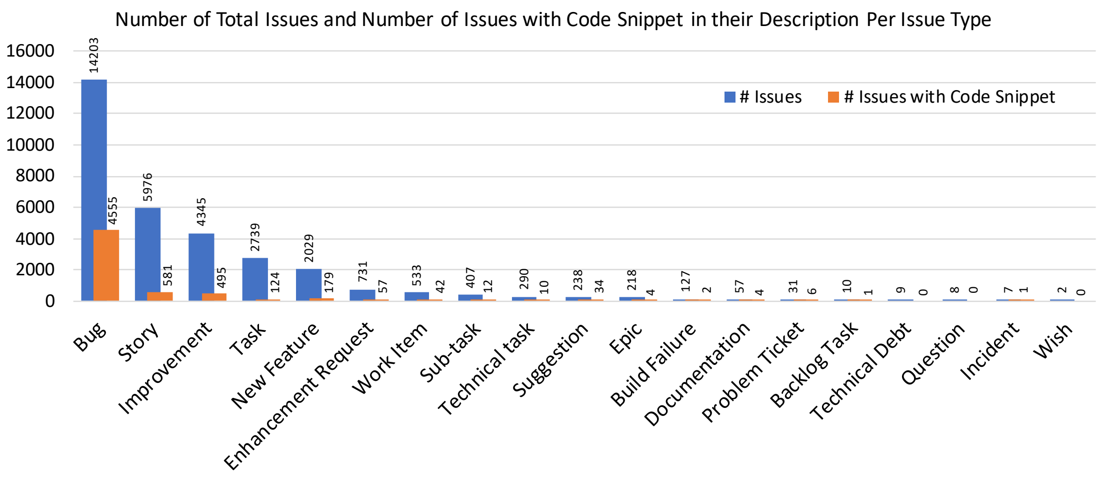
Considering the variations observed between the replication and the original study [26], we cannot confirm all conclusions made in the original study for RQs1-2. We found that Deep-SE does not outperform the Median baseline for all projects (RQ1) and is not always statistically significantly better than TF-IDF-SE (RQ2). Moreover, the statistical test performed on the distribution of the absolute errors obtained by Median estimator vs Deep-SE and TF-IDF-SE, respectively, showed that the null hypothesis cannot be rejected (i.e. the MAEs are different) for over half of the projects investigated (RQ2.1). A broader experiment with the Tawosi dataset (RQ2.2), which contains a larger number of projects and issues, showed that the results of these methods are significantly different in only one-fifth of the cases.

Furthermore, we observed that Deep-SE is not so effective for cross-project estimation (RQ3), thus confirming the conclusion made in the original study. Using other additional projects for training Deep-SE or augmenting the training set with issues from other projects developed within the same repository does not improve its prediction performance statistically significantly (RQ4). We also showed that pre-training the lower layers of Deep-SE, which demands a large number of issues and takes a long time to execute, does not significantly affect its accuracy or convergence (RQ5).

Overall, our study reveals that current approaches to Agile Software Effort Estimation, which in essence try to find semantic similarities between user stories to estimate effort, very often fail to provide statistically significantly better estimations than a naïve baseline technique like the Median estimator. This suggests that semantic similarity between user stories might not be sufficient, or even effective, for issue-level effort estimation. For instance, it is possible that two user stories that discuss the same concept (thus are semantically related) demand different amounts of effort to be resolved (e.g., one asks for a new feature to be implemented, and the other requires a small change in an already implemented feature). Table 4.13 shows an example of five semantically related user stories from the Spring XD project. As we can see, these semantically similar user



(a)



(b)

Figure 4.1: Proportion of Issue Types in the Tawosi Dataset (a), and the number of total issues and the number of issues with code snippet(s) in their description, grouped by issue type, in the Tawosi dataset (b). Issue types are ordered by their frequency in descending order from left to right.

stories are estimated to have different story points. It means that the distribution of the story points in the groups of semantically related user stories is not necessarily different from the distribution of the story points in the set of all user stories in the project. In other words, using semantic similarity does not seem effective in discriminating user stories with regard to their story point. That is a possible explanation for why methods like Deep-SE and TF-IDF-SE, which rely on semantic similarities between user stories to make a mapping between the groups of similar user stories and their story points, fail to perform better than the Median estimator. Future work might devise and experiment with additional effort drivers extracted from issue reports in order to achieve more accurate effort estimates.

Moreover, future work can investigate the effect of pre-processing the textual input (i.e., title and description), for example, to remove non-natural text like links and code snippets. In fact, the current version of Deep-SE indeed does not perform any pre-processing; however, we observed that 6,107 issues in the Tawosi dataset (which accounts for 19.11% of all issues in this dataset) contain code snippets or stack traces in their description (see our online appendix [123] for a break down of this information per issue type). The presence of such information may hinder Deep-SE's learning ability and could be investigated in future work.

We also observe that Deep-SE is trained using issues of different types such as Bug, Story, Improvement, etc., (see in Figure 4.1a the proportion of the different issue types in the Tawosi dataset). Therefore, the structure and content of the issues used as training data in both the original and replication study are heterogeneous, which can hinder the learning ability of the model. For instance, we observe that there is a statistically significant difference between the length of the description of Bug and Story issue types (two-sided Wilcoxon test $p - value = 2.2\text{e}{-16}$ with a medium (0.72) effect size). As mentioned earlier, 19.11% of the issues in the Tawosi dataset have code snippets or stack traces in their description (see Figure 4.1b for the number of issue reports including code snippet in their description, along with the total number of issues for each issue type group). Although in both studies, Deep-SE uses a limited number of tokens (specifically, 100 tokens) from the beginning of the issue context (i.e., title+description) to train and test, it would be interesting to investigate and compare in future work the performance of the model trained on all issues vs a model trained on one issue type at a time (e.g., bug report).

We hope that the results of our study will encourage further work aiming at improving methods for Agile Software Effort Estimation, as well as further prove the importance of replication studies and making replication packages publicly available in order to support reproduction, replication and extension of previous work: *"If I have seen further [than others], it is by standing on the shoulders of giants."*[13]

[13]Newton, Isaac. "Letter from Sir Isaac Newton to Robert Hooke". Historical Society of Pennsylvania. Retrieved 7 June 2018.

In the next chapter, I will investigate a clustering-based method for SP estimation, which is not only simpler but its decision can also be explained to the end user, something that most Deep Learning-based models lack.

Chapter 5

Clustering Approach to Estimate Story Points

Published as: “Investigating the Effectiveness of Clustering for Story Point Estimation”
By: Vali Tawosi[1], Afnan Al-Subaihin[1], and Federica Sarro[1]
[1] Department of Computer Science, University College London, United Kingdom
At: The 29th IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER’22)

Abstract – In the previous chapter, we saw that Deep-SE, a Deep-learning based model for SP estimation was not able to outperform the much simpler Median baseline method for all the projects investigated statistically significantly. This chapter presents a new approach for SP estimation based on analysing textual features of software issues by employing latent Dirichlet allocation (LDA) and clustering. LDA is first used to represent issue reports in a new space of generated topics. Then, hierarchical clustering is used to agglomerate issues into clusters based on their topic similarities. Next, estimation models are built using the issues in each cluster. Finally, the closest cluster to the new coming issue is identified, and the model from that cluster is used to estimate the SP. This approach is evaluated on the Tawosi dataset from the previous chapter (see Section 4.3.2.3), a dataset of 26 open source projects with a total of 31,960 issues, and compared against both baselines and Deep-SE.

The results show that the estimation performance of our proposed approach is as good as the state-of-the-art. However, none of these approaches is statistically significantly better than more naive estimators in all cases, which does not justify their additional complexity.

5.1. INTRODUCTION

In agile development, Story Point (SP) is a commonly used measure of the complexity and required effort of completing a software development task, be it an implementation, perfective or corrective maintenance task [1], [59]. Teams typically carry out assigning story points to these tasks in order to plan for the content of upcoming sprints. To this end, teams mainly rely on expert estimation methods like Planning Poker and Delphi [18]. However, expert judgment has been shown to be prone to bias due to its reliance on subjective assessment [5], [100], [109]. This motivated several research endeavours to find automated ways to predict story points of a task given its features with the aim of avoiding inaccurate estimations by human judgement, in addition to, and more importantly to an agile team, producing consistent estimations throughout the project's lifecycle.

Task descriptions (referred to as *user stories* in agile development) are a convenient information source for both humans and automated SP estimators. This information, which is usually conveyed via a few sentences written in natural language by product owners, developers, or users, is available upon the creation of a new task. Most SP estimation techniques study the similarities between the task at hand and the previously completed tasks to decide on the SP value of the new task [22]–[26], [53].

On the other hand, previous studies in software effort estimation showed that software engineering data often contain a large amount of variability [65], [66]; as previously shown in the literature for traditional software effort estimation [65], [67]–[72].

Therefore, in order to help reduce such variability, this chapter proposes and investigates the suitability of a novel clustering-based model to estimate the SP values of new issues. We dubbed this approach **L**DA[1]-based **H**ierarchical **C**lustering for **S**tory point **E**stimation (**LHC-SE**), hereafter.

The proposed approach, LHC-SE, relies on the similarities of historical issue descriptions by grouping them into coherent clusters that maximize their prediction power. These clusters are formed by representing issues as vectors of their LDA-extracted topics; the estimation of SP for a new issue is then inferred from the SP scores of issues in its assigned cluster. The results of three SP assignments (estimation models) are reported based on the resulting clusters: first is assigning the issue at hand the mean SP of the issues in its assigned cluster; second, the median of the aforementioned; and finally, assigning it the SP of the issue deemed most similar.

To the best of our knowledge, this study is the first to investigate whether clustering can help improve SP estimation accuracy by reducing the variance in the issue descriptors. This is motivated by the observations made in previous work on traditional software effort estimation where clustering techniques are employed to reduce the variability, which often leads to the construction of more accurate effort estimation

[1]Latent Dirichlet allocation

models [65], [67]–[71], [124], [126]. Previous work that uses clustering mainly uses it to group projects according to attributes that are pertinent to the task of effort estimation (i.e., cost drivers), such as size measures or manager/team experience. Whereas this study uses clustering to group the issue reports according to their topic. This makes the approach independent from the cost-drivers used as the basis for the estimation. Furthermore, this study employs the largest dataset (26 open-source projects and a total of 31,960 issues) used for SP estimation thus far (i.e., the Tawosi dataset, described in Section 4.3.2.3).

Similar to the previous chapter, the SP values estimated by human experts are used as the ground truth to evaluate the accuracy of the estimation model. Accuracy is measured by using robust measures; namely, the mean and median absolute errors of the estimations and the relative improvement over random guessing [119].

The results show that clustering issues based on their topic similarity (i.e., LHC-SE) improves the accuracy of estimation over Random Guessing and Mean baseline method, with statistical significance. While it is comparable to the state-of-the-art SP methods proposed in the literature, we also observed that the Median baseline estimator achieves similar accuracy as our model and the state-of-the-art on this dataset. We discuss these results and our observations in Section 5.5. The scripts and data we used in this study can be found online [138].

5.2. THE PROPOSED METHOD

The proposed method relies on clustering similarly described issues, such that, given an issue with an unknown SP score, its score can be derived from the SP scores of the issues deemed most similar to it (i.e., issues that reside in the same cluster).

In order to carry out the clustering, the similarities among issues need to be measured based on their natural language description. To this end, issues are represented as vectors in a numerical vector space, such that the distance among issues could be used as a proxy for issue similarity. This study uses topic modelling, namely, Latent Dirichlet Allocation (LDA); which uses statistical models to infer a set of topics in textual documents and represents the documents as the set of probabilities of their relevance to each of the inferred topics [139], [140].

Equipped with a numerical representation of issues in a vector space, a clustering algorithm can be employed to group similar issues together. Many clustering techniques require the choice of k (number of clusters) to be known a priori, which is usually unknown for software engineering data [68]. In order to enable the discovery of a suitable k, we use hierarchical clustering, which produces a dendrogram that can be efficiently investigated for the most suitable cut-off point [141]. In this study, this choice of k is guided by estimation accuracy on a validation subset of the dataset. Using the resulting clusters, when a new issue needs to be assigned an SP score, it is first converted to a vector using the pre-existing LDA model. Afterwards, the most suitable cluster for the issue is identified as the one to which the closest issue belongs. Once the cluster of most similar issues is identified, the model reports the results of first assigning the issue mean SP scores of issues in the similar cluster, the median, and finally, simply assigning the issue the SP score of the most similar issue in the historical dataset.

5.2.1. Text Pre-processing and Topic Modelling

To capture the context of issue reports and their purpose, the title and the description of the issue are combined, dubbed the issue-context hereafter. In order to create a vector representation of the issue context, first, basic text cleaning and pre-processing operations are performed on the text. Specifically, the URLs, code snippets, and all non-alphanumeric characters are removed from the issue context, the text is converted to lowercase, punctuation and English stop-words are removed, and finally, the words with less than two characters are removed. No stemming is performed on the tokens, as previous work showed that it is prone to over-stemming, which may lower the accuracy of the results [142].

To generate an LDA topic model, we first join all the pre-processed training issue contexts from a designated training subset of issues for all 26 projects used in this study to build a large corpus. The training corpus was then fed into the LDA topic modelling algorithm (using the `topicmodels` library in `R`). In order to set the number of topics t needed by the generative model, a range of t values is explored to find the t that produces a model with the minimum perplexity. The perplexity was evaluated using the validation subset of the dataset, which was not included in the initial model generation. The used LDA technique employs Gibbs sampling [143] to identify topics in the corpus, the α and δ parameters were set to $1/t$ and 0.1, respectively. Fig. 5.1 shows the perplexity of the models built for different t values. As we can see, the model with the least perplexity is produced with 2,265 topics. The generated topic model is then used to generate posterior probabilities for issues in the testing subset of the dataset; thus representing each issue as the vector of all topics such that each cell in the vector represents the relatedness of the issue to the respective topic.

5.2.2. Clustering

Given the generated vector space of the issues in the dataset, a clustering algorithm is used to cluster similar issues, with regard to their topics, into cohesive clusters. This is done using agglomerative hierarchical clustering (using Ward's linkage criterion [144] and *cosine* as a distance measure). This generates a dendrogram of the clustering options for each k. This dendrogram can be explored at various cut-off points. A sample dendrogram for the COMPASS project is shown in Fig. 5.2.

In order to discover the most suitable k, we perform a simple greedy search to find

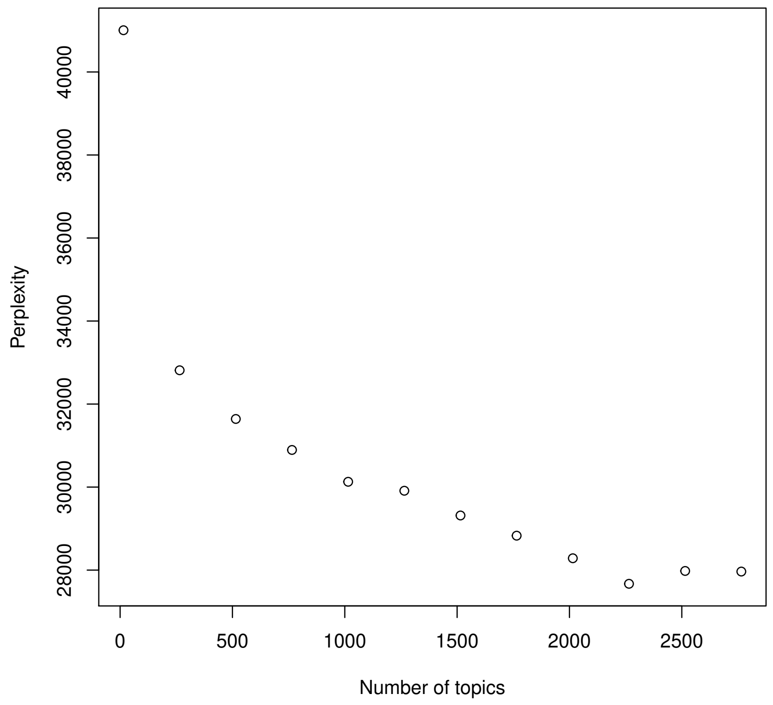


Figure 5.1: Perplexity of the LDA topic model per number of topics (i.e., t-values).

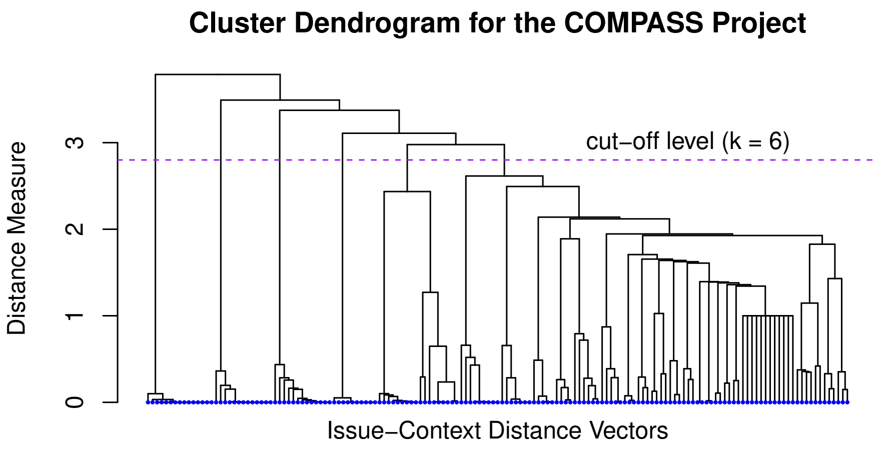


Figure 5.2: A sample dendrogram of agglomerative hierarchical clustering of issues (COMPASS project). A sample cut-off line is shown on the plot, which cuts the dendrogram at level 6, thus producing 6 clusters.

the cut-off point that generates the clustering solution that would produce the most accurate estimation models.

We investigate three strategies for selecting the most suitable cut-off k: (a) a k which when used, the estimation models of the resulting clustering produce the lowest Mean Absolute Error (i.e., *MAE-based* strategy), (b) a k which when used, the estimation models produce the lowest Median Absolute Error (i.e., *MdAE-based* strategy), and (c) a k which when used, the resulting clusters have the highest silhouette index (i.e., *Silhouette-based* strategy). The silhouette index is an internal measure of cluster quality by calculating how similar are the issues to each other in their own cluster (cohesion) compared to other clusters (separation) [145]. To evaluate the first two strategies, MAE and MdAE are calculated over the validation subset of the dataset. A range of different k values is examined per project, starting from 3 to $0.9 \times l$ with $\frac{l}{10}$ increments, where l is the size of the training set. One k value is selected per project using each of the three strategies, and experiments are performed with all three strategies.

5.2.3. Estimation Models

Once a clustering solution is selected, an SP estimation model is built using issues in each cluster. Three models are investigated: (a) *Cluster Mean-based* and (b) *Cluster Median-based* estimators, which return the mean/median SP of all the issues in the cluster, respectively; and (c) *Closest Point-based* estimator, which returns the SP value of the closest issue to the queried issue, as the estimated SP.

Given a new issue, the previously generated topic model is used to compute the posterior probabilities of the issue-context. Then, its $cosine$ distance from all the issues used in the training phase is computed to determine the closest cluster to the issue at hand. Then, the estimation model of that cluster is used to estimate the SP value for the new issue.

5.3. EMPIRICAL STUDY DESIGN

In order to evaluate the performance of LHC-SE, we investigate the accuracy of the SP estimation models it produces. Additionally, we explore whether additional features help improve the estimation accuracy of the clustering approach. Finally, we compare the resulting performance with other state-of-the-art techniques in SP effort estimation using natural language. To answer these questions, the Tawosi dataset from the previous chapter (see Section 4.3.2.3) is used to train, evaluate and validate the models. Following is a detailed report of the empirical study design.

5.3.1. Research Questions

We investigate three research questions to assess the effectiveness of LHC-SE, against the baseline methods and previous work.

RQ1. Sanity Check *Does clustering of issue reports based on their textual similarities help accurately estimate story points?*

To see whether the proposed approach is a suitable method for estimating story points, we compare it against baseline estimators. Specifically, we compare LHC-SE to Random guessing, Mean and Median baselines. Mean and Median baselines are simple models used for sanity checks; they mainly involve assigning the issue at hand the mean and median SP over all previous issues, respectively.

To be accepted as a suitable estimation method, LHC-SE should be able to outperform these baseline techniques.

RQ2. Additional Features *Can additional features help improve the estimation accuracy of the clustering approach?*

In a further investigation, the LDA-generated topic probabilities extracted from the issue context are augmented with additional features in the vector space. In particular, we add two features that are available when the issue is created: the issue type (e.g., *story*, *improvement*, *bug*, etc.) and the component(s) from which the issue rose (e.g., *UI*, *Runtime*, *DSL*, etc.). Since type and component are categorical variables, we use a one-hot encoding to convert them to numerical features to be able to exploit them with our approach. We also add issue report length, which is the number of characters used to describe the issue. This can serve as an indicator of the complexity of the issue. Adding these features creates a new variant of our model, we call it LHC$_{TC}$-SE to distinguish it from the base LHC-SE model.

Furthermore, we add TF-IDF features to LHC$_{TC}$-SE features to see if they help the clustering approach achieve more coherent clusters, thus improving its estimation accuracy. We refer to this variant as LHC$_{TC+TFIDF}$-SE.

RQ3. Comparison to the Previous Work *How does the clustering approach compare to the existing SP estimation approaches?*

To answer this question, the best variant of the clustering approach-based model is compared to two previous works (introduced in the previous chapter), including a state-of-the-art deep-learning-based model for SP estimation. Specifically, we compare the estimation accuracy of our approach to TF-IDF-SE and Deep-SE (see description of these approaches in Section 4.2).

5.3.2. Data

A large number of issues were extracted from the TAWOS dataset [27] (Chapter 3) to evaluate the proposed approach. Specifically, we use the same sample of issues used in the previous chapter, to be able to compare the proposed approach to TF-IDF-SE and Deep-SE on a common dataset. For a description of how this sample is extracted from the TAWOS dataset see Section 4.3.2.3. Table 4.2 shows the descriptive statistics of the Tawosi dataset.

Similar to the validation method used in the previous chapter, for each project, the issues are ordered in ascending order with respect to their creation time, and split into three subsets (namely, training, validation, and testing) with a ratio of 60%:20%:20%, thereby using the older issues for training and newer issues for testing.

5.3.3. Evaluation Measures

Similar to previous studies on software effort estimation, we use measurements that are built upon the absolute error between the predicted value and the actual value. These measures are the Mean Absolute Error (MAE), the Median Absolute Error (MdAE), and the Standard Accuracy (SA) (see Section 4.3.4 for description and definition).

5.3.4. Statistical Analysis

To check if the difference in the results achieved by two methods is statistically significant, a non-parametric statistical test is performed. Specifically, the Wilcoxon Ranked-Sum test (a.k.a. Mann–Whitney U test) [146] with confidence limit at $\alpha = 0.05$, corrected with Bonferroni, is applied on the distribution of the absolute errors produced by the methods under investigation. We tested the hypothesis:

Hypothesis 5.3.1: Null Hypothesis

The distribution of absolute errors produced by two prediction models P_i and P_j are not different.

A one-way Wilcoxon test is performed; hence, if the test rejects the null hypothesis, the alternative hypothesis is accepted:

Hypothesis 5.3.2: Alternative Hypothesis

The distribution of absolute errors produced by the P_i are lower than those produced by the P_j.

As done in previous work [40], [49], [147], [148], we use the win-loss-tie counting for summarising the results of the Wilcoxon test, as follows: if the distribution i is statistically significantly better (less) than j according to the Wilcoxon test we update win_i and $loss_j$, otherwise we increment tie_i and tie_j.

To measure the effect size of the difference, the Vargha Delaney's $\hat{A}_{12}$ measure is used [133], which is a standardised non-parametric effect size measurement, to assess the effect size of the difference between two methods [38], [133]. For two algorithms

1 and 2, $\hat{A}_{12}$ measures the probability of 1 performing better than 2 with respect to a performance measure (see Section 4.3.4 for definition). Similar to the boundaries set in the previous chapter, the effect size is considered negligible for $0.6 < \hat{A}_{12}$ (indicated by an ‘N’ beside its value), small (S) for $0.6 \leq \hat{A}_{12} < 0.7$, medium (M) for $0.7 \leq \hat{A}_{12} < 0.8$, and large (L) for $\hat{A}_{12} \geq 0.8$, although these thresholds are not definitive [5].

5.4. RESULTS

This section presents the results of the empirical study for each of the proposed research questions.

5.4.1. RQ1. Sanity Check

Identifying the best LHC-SE model: As described in Sections 5.2.2 and 5.2.3, this study explores the use of LHC-SE with three different cluster-forming strategies and three estimators, for a total of nine different LHC-SE estimation models. Before comparing LHC-SE to the baselines, we study which combination of cluster forming and estimation strategies works best with LHC-SE. To this end, these nine strategies are compared against each other based on the Wilcoxon Rank-Sum test and the results are summarised using the win-loss-tie approach explained in Section 5.3.4. The results are shown in Table 5.1, where for each method, the rows and columns represent the nine strategies, and each cell contains the number of cases (out of 26 projects) the strategy in the row won/lost/tied against the strategy in the column. Specifically, a win is counted if the strategy in the row produces statistically significantly lower absolute errors than the strategy in the column.

Based on the results shown in Table 5.1 (RQ1), we can observe that the MAE-based k-selection strategy with Cluster Median estimation model wins most of the times (117 wins, 1 loss, and 90 ties). Thus, we select this combination of strategy and estimation model for LHC-SE to compare with the baselines.

Sanity Check: Table 5.2 shows the MAE and SA values achieved by LHC-SE and the baselines. The MdAE values are also reported for completeness.

We can observe that 24 out of 26 SA values for LHC-SE are positive, which means that LHC-SE outperforms the Random Guessing (RG) baseline in 24 cases (exceptions are the STL and DURACLOUD projects). For all these 24 cases, the difference between the absolute errors produced by LHC-SE and RG is statistically significant in favour of LHC-SE, and 12 cases also showed a large or medium effect size, while the remaining showed a small or negligible one.

LHC-SE achieves a good performance against the Mean baseline as well. It outperforms the Mean estimator in 20 cases while underperforming in only 6 cases. From these 20 cases, the improvement is statistically significant in 18 cases, with a large effect size in 5 cases and a small or negligible one in the rest. However, against

Table 5.1: RQ1 and RQ2: Win-Loss-Tie results comparing the nine different combinations of three cluster-building methods and three estimation strategies for each of the three LHC-SE-based variants. The best strategy for each variant is highlighted.

RQ	Method	k-selection strategy	Estimator	MAE-based Closest Point	MAE-based Cluster Mean	MAE-based Cluster Median	MdAE-based Closest Point	MdAE-based Cluster Mean	MdAE-based Cluster Median	Silhouette-based Closest Point	Silhouette-based Cluster Mean	Silhouette-based Cluster Median	Summary
RQ1	LHC-SE	MAE-based	Closest Point		3 | 8 | 15	0 | 20 | 6	0 | 0 | 26	3 | 9 | 14	0 | 20 | 6	0 | 0 | 26	2 | 6 | 18	0 | 14 | 12	8 | 77 | 123
			Cluster Mean	8 | 3 | 15		0 | 17 | 9	8 | 3 | 15	0 | 5 | 21	0 | 17 | 9	8 | 3 | 15	0 | 5 | 21	0 | 13 | 13	24 | 66 | 118
			Cluster Median	20 | 0 | 6	17 | 0 | 9		20 | 0 | 6	16 | 0 | 10	1 | 1 | 24	20 | 0 | 6	18 | 0 | 8	5 | 0 | 21	**117 | 1 | 90**
		MdAE-based	Closest Point	0 | 0 | 26	3 | 8 | 15	0 | 20 | 6		3 | 9 | 14	0 | 20 | 6	0 | 0 | 26	2 | 6 | 18	0 | 14 | 12	8 | 77 | 123
			Cluster Mean	9 | 3 | 14	5 | 0 | 21	0 | 16 | 10	9 | 3 | 14		0 | 16 | 10	9 | 3 | 14	0 | 5 | 21	0 | 13 | 13	32 | 59 | 117
			Cluster Median	20 | 0 | 6	17 | 0 | 9	1 | 1 | 24	20 | 0 | 6	16 | 0 | 10		20 | 0 | 6	16 | 0 | 10	3 | 0 | 23	113 | 1 | 94
		Silhouette-based	Closest Point	0 | 0 | 26	3 | 8 | 15	0 | 20 | 6	0 | 0 | 26	3 | 9 | 14	0 | 20 | 6		2 | 6 | 18	0 | 14 | 12	8 | 77 | 123
			Cluster Mean	6 | 2 | 18	5 | 0 | 21	0 | 18 | 8	6 | 2 | 18	5 | 0 | 21	0 | 16 | 10	6 | 2 | 18		0 | 12 | 14	28 | 52 | 128
			Cluster Median	13 | 0 | 13	13 | 0 | 13	0 | 5 | 21	13 | 0 | 13	13 | 0 | 13	0 | 3 | 23	13 | 0 | 13	12 | 0 | 14		77 | 8 | 123
	Method			LHC$_{TC}$-SE									
		k-selection strategy	Estimator	MAE-based Closest Point	MAE-based Cluster Mean	MAE-based Cluster Median	MdAE-based Closest Point	MdAE-based Cluster Mean	MdAE-based Cluster Median	Silhouette-based Closest Point	Silhouette-based Cluster Mean	Silhouette-based Cluster Median	Sum
RQ2	LHC$_{TC}$-SE	MAE-based	Closest Point		3 | 6 | 17	0 | 16 | 10	1 | 0 | 25	4 | 6 | 16	0 | 17 | 9	1 | 0 | 25	3 | 5 | 18	0 | 17 | 9	12 | 67 | 129
			Cluster Mean	6 | 3 | 17		0 | 13 | 13	5 | 5 | 16	2 | 3 | 21	1 | 14 | 11	5 | 4 | 17	5 | 2 | 19	1 | 14 | 11	25 | 58 | 125
			Cluster Median	16 | 0 | 10	13 | 0 | 13		16 | 0 | 10	16 | 0 | 10	3 | 1 | 22	15 | 0 | 11	18 | 0 | 8	4 | 1 | 21	**101 | 2 | 105**
		MdAE-based	Closest Point	0 | 1 | 25	5 | 6 | 15	0 | 16 | 10		7 | 6 | 13	1 | 17 | 8	0 | 0 | 26	5 | 6 | 15	0 | 16 | 10	18 | 68 | 122
			Cluster Mean	6 | 4 | 16	3 | 2 | 21	0 | 16 | 10	5 | 7 | 14		1 | 15 | 10	5 | 6 | 15	7 | 3 | 16	1 | 15 | 10	28 | 68 | 112
			Cluster Median	17 | 0 | 9	14 | 1 | 11	1 | 3 | 22	17 | 1 | 8	15 | 1 | 10		17 | 1 | 8	17 | 1 | 8	2 | 1 | 23	100 | 9 | 99
		Silhouette-based	Closest Point	0 | 1 | 25	4 | 5 | 17	0 | 15 | 11	0 | 0 | 26	6 | 5 | 15	1 | 17 | 8		4 | 5 | 17	0 | 17 | 9	15 | 65 | 128
			Cluster Mean	5 | 3 | 18	2 | 5 | 19	0 | 18 | 8	5 | 5 | 16	3 | 7 | 16	1 | 17 | 8	5 | 4 | 17		1 | 17 | 8	22 | 76 | 110
			Cluster Median	17 | 0 | 9	14 | 1 | 11	1 | 4 | 21	16 | 0 | 10	15 | 1 | 10	1 | 2 | 23	17 | 0 | 9	17 | 1 | 8		98 | 9 | 101
	Method			LHC$_{TC+TFIDF}$-SE									
		k-selection strategy	Estimator	MAE-based Closest Point	MAE-based Cluster Mean	MAE-based Cluster Median	MdAE-based Closest Point	MdAE-based Cluster Mean	MdAE-based Cluster Median	Silhouette-based Closest Point	Silhouette-based Cluster Mean	Silhouette-based Cluster Median	Sum
RQ2	LHC$_{TC+TFIDF}$-SE	MAE-based	Closest Point		4 | 13 | 9	0 | 20 | 6	0 | 0 | 26	4 | 12 | 10	1 | 21 | 4	0 | 0 | 26	5 | 12 | 9	1 | 20 | 5	15 | 98 | 95
			Cluster Mean	13 | 4 | 9		0 | 17 | 9	13 | 4 | 9	5 | 5 | 16	1 | 18 | 7	13 | 4 | 9	5 | 2 | 19	2 | 17 | 7	52 | 71 | 85
			Cluster Median	20 | 0 | 6	17 | 0 | 9		20 | 0 | 6	17 | 0 | 9	0 | 1 | 25	20 | 0 | 6	17 | 0 | 9	2 | 2 | 22	113 | 3 | 92
		MdAE-based	Closest Point	0 | 0 | 26	4 | 13 | 9	0 | 20 | 6		4 | 12 | 10	1 | 21 | 4	0 | 0 | 26	5 | 12 | 9	1 | 20 | 5	15 | 98 | 95
			Cluster Mean	12 | 4 | 10	5 | 5 | 16	0 | 17 | 9	12 | 4 | 10		1 | 18 | 7	12 | 4 | 10	7 | 4 | 15	2 | 17 | 7	51 | 73 | 84
			Cluster Median	21 | 1 | 4	17 | 1 | 8	1 | 0 | 25	21 | 1 | 4	18 | 1 | 7		21 | 1 | 4	18 | 1 | 7	2 | 1 | 23	**119 | 7 | 82**
		Silhouette-based	Closest Point	0 | 0 | 26	4 | 13 | 9	0 | 20 | 6	0 | 0 | 26	4 | 12 | 10	1 | 21 | 4		5 | 12 | 9	1 | 20 | 5	15 | 98 | 95
			Cluster Mean	12 | 5 | 9	2 | 5 | 19	0 | 17 | 9	12 | 5 | 9	4 | 7 | 15	1 | 18 | 7	12 | 5 | 9		1 | 19 | 6	44 | 81 | 83
			Cluster Median	20 | 1 | 5	16 | 2 | 8	2 | 2 | 22	20 | 1 | 5	17 | 2 | 7	1 | 2 | 23	20 | 1 | 5	19 | 1 | 6		115 | 12 | 81

the Median estimator, LHC-SE performs rather poorly, albeit with a negligible effect size. Although LHC-SE outperforms the Median estimator in 10 cases, the Median estimator outperforms LHC-SE in the remaining 16 cases. Nonetheless, the results of the Wilcoxon test reveal that the difference in the estimation performance of these two methods is statistically significant in only three cases, one in favour of LHC-SE and two in favour of the Median baseline.

These results show that LHC-SE outperforms RG and Mean baselines in the majority of the cases but emerges shoulder-to-shoulder with the Median baseline. This motivates checking whether augmenting the feature set of LHC-SE increases its accuracy (RQ2).

Answer to RQ1: *LHC-SE easily outperforms Random Guessing and Mean baselines, and performs similarly to the Median baseline.*

5.4.2. RQ2. Additional Features

To answer this question, the base LHC-SE model from RQ1 is compared to two other variants (i.e., LHC$_{TC}$-SE which incorporates issue length, type, and components, and LHC$_{TC+TFIDF}$-SE which incorporates the aforementioned in addition to TF-IDF scores for each issue).

Identifying the best strategies: Similar to RQ1, we first identify which combination of cluster forming and estimation strategies works best with each of the two additional variants. The middle and last rows of Table 5.1 show the win-loss-tie scores of the two variants (i.e., LHC$_{TC}$-SE and LHC$_{TC+TFIDF}$-SE) for different k-selection and estimation strategies, with respect to their Wilcoxon test results. We can observe that the best combination for LHC$_{TC}$-SE is the MAE-based k-selection with Cluster Median estimation, achieving the highest score (101 wins, 2 losses, and 105 ties). Whereas, for LHC$_{TC+TFIDF}$-SE the MdAE-based k-selection with Cluster Median estimator scores highest.

Comparing LHC-SE variants: Using the strategy that works the best for each variant, we compare the three LHC-based variants (each with their best performing strategy) in Table 5.3. As we can see, the three models score very close to one another (they draw a tie in almost all the cases). Specifically, LHC-SE and LHC$_{TC}$-SE perform similarly (each beats the other on two projects, and are tie on the others). However, LHC$_{TC}$-SE is better than LHC$_{TC+TFIDF}$-SE in more cases, and therefore it scores the highest number of wins among the three. So, we select LHC$_{TC}$-SE for the comparison against existing SP estimation approaches (RQ3).

Answer to RQ2: *Using type, component(s) and report length of issues, in addition to their LDA topics, help LHC$_{TC}$-SE perform better.*

Table 5.2: RQ1 and RQ3: MAE, MdAE and SA values achieved by LHC-SE, LHC$_{TC}$-SE, Deep-SE, TF-IDF-SE, and the Mean and Median baselines. The best values per method and per project are printed in bold face.

Project	Method	MAE	MdAE	SA	Project	Method	MAE	MdAE	SA	Project	Method	MAE	MdAE	SA
MESOS	LHC-SE	1.34	**1.00**	34.38	CONFCLOUD	LHC-SE	1.34	1.00	40.41	SERVER	LHC-SE	**0.85**	1.00	**59.47**
	LHC$_{TC}$-SE	**1.33**	**1.00**	**34.63**		LHC$_{TC}$-SE	1.37	1.00	39.04		LHC$_{TC}$-SE	**0.85**	1.00	**59.47**
	Deep-SE	1.34	1.12	34.07		Deep-SE	1.48	**0.93**	33.89		Deep-SE	0.89	**0.71**	57.60
	TF-IDF-SE	1.34	**1.00**	34.38		TF-IDF-SE	**1.33**	1.00	40.86		TF-IDF-SE	0.93	1.00	55.88
	Mean	1.37	1.08	32.72		Mean	1.49	1.23	33.65		Mean	1.56	1.86	25.99
	Median	1.34	**1.00**	34.38		Median	**1.33**	1.00	**40.87**		Median	**0.85**	1.00	59.46
ALOY	LHC-SE	1.84	2.00	26.57	CONFSERVER	LHC-SE	0.96	1.00	49.64	MDL	LHC-SE	6.31	7.00	57.30
	LHC$_{TC}$-SE	2.28	2.00	9.01		LHC$_{TC}$-SE	0.96	1.00	49.64		LHC$_{TC}$-SE	6.31	7.00	57.30
	Deep-SE	1.51	**1.28**	39.67		Deep-SE	**0.91**	**0.64**	**52.28**		Deep-SE	**3.55**	**2.77**	**76.00**
	TF-IDF-SE	**1.44**	2.00	**42.53**		TF-IDF-SE	0.96	1.00	49.64		TF-IDF-SE	6.31	7.00	57.30
	Mean	2.23	2.17	10.84		Mean	1.35	1.45	29.17		Mean	14.54	15.23	1.58
	Median	**1.44**	2.00	**42.53**		Median	0.96	1.00	49.64		Median	6.31	7.00	57.30
APSTUD	LHC-SE	4.14	3.00	30.20	DNN	LHC-SE	**0.71**	1.00	**42.60**	MULE	LHC-SE	2.27	2.00	37.11
	LHC$_{TC}$-SE	**3.99**	3.00	**32.81**		LHC$_{TC}$-SE	**0.71**	1.00	**42.60**		LHC$_{TC}$-SE	2.60	3.00	28.16
	Deep-SE	4.31	2.70	27.37		Deep-SE	0.72	**0.69**	41.69		Deep-SE	**2.24**	**1.68**	37.95
	TF-IDF-SE	**3.99**	3.00	**32.81**		TF-IDF-SE	0.79	1.00	36.13		TF-IDF-SE	3.58	2.00	0.81
	Mean	4.00	**2.49**	32.72		Mean	0.80	0.88	35.28		Mean	2.79	3.18	22.68
	Median	**3.99**	3.00	**32.81**		Median	**0.71**	1.00	**42.60**		Median	**2.24**	2.00	**38.05**
CLI	LHC-SE	1.87	2.00	29.32	FAB	LHC-SE	0.67	1.00	69.75	NEXUS	LHC-SE	1.14	1.00	22.52
	LHC$_{TC}$-SE	**1.76**	2.00	33.35		LHC$_{TC}$-SE	**0.65**	1.00	**70.47**		LHC$_{TC}$-SE	1.22	1.00	16.88
	Deep-SE	**1.76**	**1.30**	**33.44**		Deep-SE	0.86	**0.71**	61.06		Deep-SE	**1.08**	0.88	**26.56**
	TF-IDF-SE	2.98	3.00	-12.84		TF-IDF-SE	1.10	1.00	50.31		TF-IDF-SE	1.17	1.00	20.68
	Mean	2.14	2.61	18.93		Mean	1.19	1.10	46.21		Mean	1.11	**0.58**	24.69
	Median	1.77	2.00	33.04		Median	0.67	1.00	69.75		Median	1.17	1.00	20.68
DAEMON	LHC-SE	2.81	3.00	32.09	STL	LHC-SE	1.28	1.00	-6.77	XD	LHC-SE	1.54	**1.00**	39.53
	LHC$_{TC}$-SE	**2.74**	3.00	**33.81**		LHC$_{TC}$-SE	0.95	1.00	20.41		LHC$_{TC}$-SE	1.50	**1.00**	40.85
	Deep-SE	3.29	**2.00**	20.55		Deep-SE	1.18	1.12	1.91		Deep-SE	**1.45**	1.16	**43.06**
	TF-IDF-SE	**2.74**	3.00	**33.81**		TF-IDF-SE	**0.84**	**0.00**	**30.12**		TF-IDF-SE	2.01	2.00	20.82
	Mean	2.75	2.75	33.53		Mean	0.97	1.02	19.32		Mean	1.65	1.72	34.89
	Median	**2.74**	3.00	**33.81**		Median	0.95	1.00	20.41		Median	1.55	**1.00**	39.05
TIDOC	LHC-SE	2.79	**1.00**	23.48	DM	LHC-SE	1.56	1.00	53.87	TDP	LHC-SE	1.00	1.00	37.08
	LHC$_{TC}$-SE	3.65	2.00	-0.30		LHC$_{TC}$-SE	1.52	1.00	54.94		LHC$_{TC}$-SE	1.03	1.00	35.44
	Deep-SE	**2.72**	1.19	**25.35**		Deep-SE	1.61	**0.89**	52.41		Deep-SE	**0.99**	**0.81**	37.69
	TF-IDF-SE	3.03	**1.00**	16.69		TF-IDF-SE	**1.49**	1.00	**55.71**		TF-IDF-SE	**0.99**	1.00	**37.74**
	Mean	2.99	2.59	18.00		Mean	2.60	2.43	22.83		Mean	1.17	1.38	26.26
	Median	2.77	**1.00**	24.03		Median	1.61	1.00	52.19		Median	**0.99**	1.00	**37.74**
TIMOB	LHC-SE	2.53	2.00	30.70	DURACLOUD	LHC-SE	1.25	1.00	-9.65	TDQ	LHC-SE	3.52	3.00	27.60
	LHC$_{TC}$-SE	2.48	2.00	32.12		LHC$_{TC}$-SE	0.68	1.00	39.94		LHC$_{TC}$-SE	2.92	3.00	40.01
	Deep-SE	**2.41**	**1.81**	**33.90**		Deep-SE	0.68	**0.58**	39.90		Deep-SE	**2.47**	**2.23**	**49.14**
	TF-IDF-SE	2.53	2.00	30.70		TF-IDF-SE	0.68	1.00	39.94		TF-IDF-SE	5.05	5.00	-3.95
	Mean	2.55	**1.81**	30.23		Mean	**0.67**	0.85	**41.13**		Mean	4.20	3.82	13.65
	Median	2.53	2.00	30.70		Median	0.68	1.00	39.94		Median	2.88	3.00	40.72
TISTUD	LHC-SE	**1.51**	2.00	**51.89**	COMPASS	LHC-SE	1.38	2.00	28.54	TESB	LHC-SE	0.99	1.00	32.56
	LHC$_{TC}$-SE	**1.51**	2.00	**51.89**		LHC$_{TC}$-SE	**1.30**	**1.00**	**32.46**		LHC$_{TC}$-SE	1.04	1.00	29.31
	Deep-SE	1.63	**1.38**	48.08		Deep-SE	1.63	1.34	15.25		Deep-SE	1.15	**0.73**	21.36
	TF-IDF-SE	**1.51**	2.00	**51.89**		TF-IDF-SE	1.38	2.00	28.54		TF-IDF-SE	**0.97**	1.00	**33.95**
	Mean	2.01	2.16	35.93		Mean	1.48	1.63	23.05		Mean	0.99	0.99	32.71
	Median	**1.51**	2.00	**51.89**		Median	1.38	2.00	28.54		Median	0.98	1.00	33.02
CLOV	LHC-SE	3.88	2.00	46.35	EVG	LHC-SE	**0.60**	1.00	**22.69**					
	LHC$_{TC}$-SE	4.23	1.50	41.55		LHC$_{TC}$-SE	0.62	1.00	19.97					
	Deep-SE	**3.78**	1.05	**47.73**		Deep-SE	0.63	**0.54**	19.39					
	TF-IDF-SE	4.04	**1.00**	44.15		TF-IDF-SE	0.69	1.00	10.67					
	Mean	5.93	5.30	18.06		Mean	0.68	0.56	12.98					
	Median	4.01	2.00	44.55		Median	0.69	1.00	10.67					

Table 5.3: RQ2: Win-Loss-Tie summary of the Wilcoxon test results comparing the LHC-SE variants with their respective best strategies from Table 5.1. The best variant is highlighted.

	Win/Loss/Tie			
Method	LHC-SE	LHC$_{TC}$-SE	LHC$_{TC+TFIDF}$-SE	Summary
LHC-SE		2 | 2 | 22	1 | 2 | 23	3 | 4 | 45
LHC$_{TC}$-SE	2 | 2 | 22		2 | 1 | 23	**4 | 3 | 45**
LHC$_{TC+TFIDF}$-SE	2 | 1 | 23	1 | 2 | 23		3 | 3 | 46

5.4.3. RQ3. Comparison to the Previous Work

Since LHC$_{TC}$-SE was found to be the best performing variant among the models investigated (see RQ2), we compare it against the state-of-the-art.

Table 5.2 shows the MAE and SA values achieved by LHC$_{TC}$-SE, Deep-SE, TF-IDF-SE, and the Mean and Median baselines (the MdAE values are also reported, for completeness). We observe that LHC$_{TC}$-SE achieves a better (lower) MAE than Deep-SE in 14 out of 26 cases, while Deep-SE achieves a better MAE in 12 cases. LHC$_{TC}$-SE outperforms TF-IDF-SE in 15 cases, whereas it is outperformed in the remaining 11 cases. LHC$_{TC}$-SE achieves better MAE values than those achieved by Mean in 21 cases. It achieves better MAEs than the Median estimator, in 14 cases, and slightly worse in 12.

It is worth noting that in some cases the MAE values are very close (e.g., the MULE project in Table 5.2), showing that achieving a lower MAE does not guarantee that a method performs statistically significantly better than the other. For this reason, we also provide the p-values and effect sizes of the statistical tests performed on LHC$_{TC}$-SE against the other methods per project in Table 5.4 and, then summarise these results as win-loss-tie in Table 5.5.

Based on the results reported in Table 5.4, we observe that LHC$_{TC}$-SE performs statistically significantly better than Deep-SE for five projects (i.e., TISTUD, FAB, DM, COMPASS, and EVG); however, the effect size for all five cases is negligible or small. Similarly, LHC$_{TC}$-SE performs statistically significantly better than the Median estimator in two projects (i.e., DM and EVG), but the effect size for both cases is negligible. Compared to TF-IDF-SE, LHC$_{TC}$-SE showed statistically significant improvement for five projects (i.e., CLI, FAB, EVG, MULE, and TDQ), in two cases with a medium effect size, in one case with a small one, and in the remaining two cases with a negligible one. Against the Mean estimator, LHC$_{TC}$-SE shows a significant improvement. Particularly, from 19 projects for which LHC$_{TC}$-SE produces statistically significantly better estimations, the difference shows a large effect size in four cases, a medium effect size in two, a small effect size in seven, and a negligible effect size in the remaining six cases. Finally, compared to the RG baseline, LHC$_{TC}$-SE shows a statistically significant difference for all projects but one. Among the 25 projects in which LHC$_{TC}$-SE outperforms RG, 11 cases show a large effect size, eight cases show a medium, four cases a small, and two cases show a negligible effect size.

Based on the win-loss-tie summary (Table 5.5), we can conclude that LHC$_{TC}$-SE scores are very close to Deep-SE and Median estimator, though it is ahead by two and one wins, respectively. Considering the scores achieved by LHC$_{TC}$-SE against the other methods (see first row Table 5.5), we can also observe that LHC$_{TC}$-SE never wins less than it loses to the other methods.

Overall, these results show that our proposed method outperforms RG and Mean baselines statistically significantly and matches the accuracy of the state-of-the-art, while slightly enhancing it in some cases, it does not perform worse in most of the cases. It also performs as good as the Median estimator.

Answer to RQ3: *LHC$_{TC}$-SE matches the accuracy of the state-of-the-art, while slightly enhancing it in some cases, it does not perform worse in most of the cases.*

Table 5.4: RQ3: Wilcoxon Test results (with Vargha-Delaney effect size in brackets) comparing LHC$_{TC}$-SE against each of the previous work and the baseline methods.

Project	LHC$_{TC}$-SE vs.				
	Deep-SE	TF-IDF-SE	Mean	Median	Random
MESOS	0.147 (0.52) _	0.420 (0.50) _	0.001 (0.57) N	0.420 (0.50) _	<0.001 (0.73) M
ALOY	0.992 (0.36) _	0.999 (0.33) _	0.435 (0.51) _	0.999 (0.33) _	0.167 (0.56) _
APSTUD	0.186 (0.54) _	0.501 (0.50) _	0.370 (0.51) _	0.501 (0.50) _	<0.001 (0.73) M
CLI	0.736 (0.47) _	<0.001 (0.74) M	0.030 (0.60) S	0.466 (0.50) _	<0.001 (0.75) M
DAEMON	0.344 (0.53) _	0.502 (0.50) _	0.629 (0.48) _	0.502 (0.50) _	<0.001 (0.77) M
TIDOC	0.826 (0.47) _	0.869 (0.47) _	0.002 (0.58) N	0.873 (0.47) _	<0.001 (0.70) M
TIMOB	0.563 (0.50) _	0.121 (0.52) _	0.048 (0.52) N	0.121 (0.52) _	<0.001 (0.73) M
TISTUD	<0.001 (0.60) S	0.500 (0.50) _	<0.001 (0.63) S	0.500 (0.50) _	<0.001 (0.84) L
CLOV	0.920 (0.43) _	0.949 (0.42) _	<0.001 (0.75) M	0.301 (0.53) _	<0.001 (0.81) L
CONFCLOUD	0.497 (0.50) _	0.552 (0.49) _	0.036 (0.60) S	0.552 (0.49) _	<0.001 (0.81) L
CONFSERVER	0.132 (0.55) _	0.501 (0.50) _	<0.001 (0.65) S	0.501 (0.50) _	<0.001 (0.77) M
DNN	0.630 (0.49) _	0.386 (0.51) _	<0.001 (0.69) S	0.500 (0.50) _	<0.001 (0.82) L
FAB	0.003 (0.64) S	0.002 (0.64) S	<0.001 (0.83) L	0.435 (0.51) _	<0.001 (0.95) L
STL	0.110 (0.58) _	0.951 (0.41) _	0.007 (0.65) S	0.502 (0.50) _	0.003 (0.67) S
DM	<0.001 (0.56) N	1.000 (0.46) _	<0.001 (0.83) L	<0.001 (0.54) N	<0.001 (0.93) L
DURACLOUD	0.052 (0.58) _	0.501 (0.50) _	0.181 (0.55) _	0.501 (0.50) _	<0.001 (0.69) S
COMPASS	0.043 (0.60) S	0.403 (0.51) _	0.112 (0.57) _	0.403 (0.51) _	0.004 (0.65) S
EVG	0.034 (0.53) N	0.008 (0.54) N	0.045 (0.53) S	0.008 (0.54) N	0.015 (0.54) N
SERVER	0.413 (0.51) _	0.481 (0.50) _	<0.001 (0.80) L	0.422 (0.51) _	<0.001 (0.91) L
MDL	1.000 (0.17) _	0.500 (0.50) _	<0.001 (1.00) L	0.500 (0.50) _	<0.001 (1.00) L
MULE	1.000 (0.41) _	<0.001 (0.57) N	<0.001 (0.59) N	1.000 (0.41) _	<0.001 (0.77) M
NEXUS	0.721 (0.49) _	0.892 (0.47) _	0.897 (0.47) _	0.892 (0.47) _	<0.001 (0.59) N
XD	0.626 (0.49) _	0.056 (0.55) _	0.005 (0.58) N	0.774 (0.48) _	<0.001 (0.84) L
TDP	0.613 (0.49) _	0.564 (0.49) _	0.016 (0.59) N	0.564 (0.49) _	<0.001 (0.82) L
TDQ	0.995 (0.42) _	<0.001 (0.79) M	<0.001 (0.67) M	0.516 (0.50) _	<0.001 (0.84) L
TESB	0.105 (0.54) _	0.817 (0.47) _	0.056 (0.55) _	0.753 (0.48) _	<0.001 (0.67) S

Table 5.5: RQ3: Win-Loss-Tie summary of the Wilcoxon test results comparing LHC$_{TC}$-SE against each of the previous work and the baseline methods. The best method is highlighted

	Win/Loss/Tie						
Method	LHC$_{TC}$-SE	Deep-SE	TFI/DF-SE	Mean	Median	Random	Summary
LHC$_{TC}$-SE		5 | 4 | 17	5 | 3 | 18	19 | 0 | 7	2 | 2 | 22	25 | 0 | 1	**56 | 9 | 65**
Deep-SE	4 | 5 | 17		5 | 4 | 17	16 | 1 | 9	4 | 3 | 19	25 | 0 | 1	54 | 13 | 63
TF-IDF-SE	2 | 5 | 19	4 | 5 | 17		16 | 3 | 7	2 | 5 | 19	23 | 3 | 0	47 | 21 | 62
Mean	0 | 19 | 7	1 | 16 | 9	3 | 16 | 7		1 | 20 | 5	24 | 0 | 2	29 | 71 | 30
Median	2 | 2 | 22	3 | 4 | 19	5 | 3 | 18	20 | 1 | 5		25 | 1 | 0	55 | 11 | 64
Random	0 | 25 | 1	0 | 25 | 1	3 | 23 | 0	0 | 24 | 2	1 | 25 | 0		4 | 122 | 4

5.5. DISCUSSION

Our results show that LDA is able to capture information latent in the issue context to enable the clustering algorithm to form useful clusters for story point estimation.

In RQ1, we analysed LHC-SE, which solely uses LDA-generated posterior topic probabilities. This approach outperforms random guessing in all cases and the Mean baseline in 77% of the cases, based on the MAE values. However, the Median baseline performs as good as LHC-SE. We should note that the Median baseline is also performing better than all other methods investigated in this study (see Table 5.5), including the two previous works (TF-IDF-SE and Deep-SE), though both are more sophisticated methods.

RQ2 results show that augmenting LDA-generated posterior topic probabilities with extra features from issue reports helps the clustering algorithm to form a better clustering solution, thus improving the estimation accuracy, though marginally (see Table 5.3). In fact, the addition of TF-IDF weights to the feature set (i.e., LHC$_{TC+TFIDF}$-SE) showed improvements over LHC-SE; however, the accuracy of LHC$_{TC+TFIDF}$-SE was slightly lower than those achieved by LHC$_{TC}$-SE. These results suggest that adding more discriminating attributes to the feature set can help the clustering algorithm form even higher-quality clusters.

Finally, RQ3 results reveal that LHC$_{TC}$-SE never performs worse than the other benchmarks (e.g., the Median baseline and Deep-SE). However, considering the level of complexity of the model, the time and resources consumed to build it, and the interpretability of the models built, the Median estimator can be viewed as the more favourable model so far that can be used in practice.

We note that the fact that a naive estimation approach, such as the Median one, provides comparable and, in some cases, even better results than much more sophisticated techniques like deep learning and LDA strongly indicates that the research advances made so far are unsatisfactory. This also suggests that future research on story point estimation might need to pay more attention to the data rather than the estimation technique when building prediction models [69]. In fact, issue reports, especially in open-source projects, are not usually written in a structured or formal way; thus, they can be very noisy and it is possible that by using tailored text pre-processing and data cleaning, the accuracy of the proposed model can be improved. On the other hand, improving the quality of the user stories written by the authors of the issue report (for example, by providing them with accurate guidelines or training) could yield less noisy data for model building, thus, improving the estimation accuracy. Besides, additional features can be extracted in order to aid prediction models in seeking more accurate estimations.

We believe that sharing these findings provides the research community with the knowledge needed to develop alternative strategies and evolve better solutions for story point estimation.

5.6. THREATS TO VALIDITY

Like previous studies, we use human-estimated story points as the ground truth, which might be biased. On the one hand, this is mitigated by the clustering of similar issues based on their description, hence augmenting the estimations of several human estimators. On the other hand, these values can be viewed as a placeholder that is used to test the model's ability to estimate SP based on historical issues' scores (wherever their origin might be). The model, therefore, can be trained on an unbiased target value when it is available (for example, the real-time spent on issue development). Currently, given the available dataset, our model can imitate human experts assisting them in their estimation at its best.

To minimize threats to conclusion validity, we carefully selected unbiased accuracy performance measures and applied statistical tests to rule out small differences.

The dataset we used represents a wide range of real-world projects. However, we cannot claim that our dataset is representative of all software projects. All our projects are collected from open-source repositories, which can differ from industrial projects in many aspects. A key difference that may affect the estimation of story points is the behaviour of contributors, developers, and project stakeholders. It is also expected that in a commercial project setting, issue reports may be written in a more disciplined environment, thus, providing more useful information and containing less noise. Therefore, further investigation of commercial projects from industrial software companies is needed to validate the conclusions made in this study.

5.7. CONCLUSION AND FUTURE WORK

In this study, we investigate a novel clustering-based model to estimate Story Point (SP), dubbed LHC-SE.

The idea behind LHC-SE is to leverage the similarity of issues, measured using the similarity of LDA-generated topic space of issue descriptions and agglomerative hierarchical clustering, to estimate the SP of a new issue based on the past most similar issues. This model works on the premise that clustering similar data points together helps reduce variance and, thus, increases the accuracy of any model built upon them.

To assess the effectiveness of our proposal we have carried out a thorough empirical study benchmarking LHC-SE's performance against those of both baselines and state-of-the-art approaches for SP estimation on the largest corpus of open-source projects used in the literature to date.

The results showed that the use of LHC-SE allows us to achieve comparable results with the state-of-the-art (i.e., based on the Wilcoxon test results, it is statistically significantly better in 5 cases, worse in 4 and tie in the remaining 17 cases). On the other end, our results also surprisingly reveal that both LHC-SE and the state-of-the-art are comparable to some naive estimators, such as simply assigning the median SP of previous issues; therefore, their additional complexity does not seem warranted.

We hope that these findings encourage researchers to develop alternative strategies and evolve better ideas for story point estimation. In future work, we suggest investigating:

- More advanced data analysis and cleaning prior to model building.
- Utilizing other contextual text representation models recently introduced in NLP research instead of LDA.
- Collecting and using additional effort-informative features available in, or derivable from, issue reports.
- Training machine learning methods on each cluster instead of baseline estimators used in this study.
- Exploring other distance measures for clustering instead of cosine similarity; and/or other clustering techniques instead of agglomerative hierarchical clustering.

Chapter 6

Effectiveness of Story Points in Estimating Effort

Published as: “On the Relationship Between Story Point and Development Effort in Agile Open-Source Software”
By: Vali Tawosi[1], Rebecca Moussa[1], and Federica Sarro[1]
[1] Department of Computer Science, University College London, United Kingdom
At: The 16th International Symposium on Empirical Software Engineering and Measurement (ESEM’22)

Abstract – Previous work has provided some initial evidence that Story Point (SP) estimated by human experts may not accurately reflect the effort needed to realise agile software projects. In this study, we aim to shed further light on the relationship between SP and agile software development efforts to understand the extent to which human-estimated SP is a good indicator of user story development effort expressed in terms of time needed to realise it. To this end, we carry out a thorough empirical study involving a total of 37,440 unique user stories from 37 different open-source projects publicly available in the TAWOS dataset. For these user stories, we investigate the correlation between the issue development time[1] (or its approximation when the actual time is not available) and the SP estimated by human experts by using three widely-used correlation statistics (i.e., Pearson, Kendall, and Spearman). Furthermore, we investigate SP estimations made by the human experts in order to assess the extent to which they are consistent in their estimations throughout the project, i.e., we assess whether the development time of the issues is proportionate to the SP assigned to them. The average results across the three correlation measures reveal that the correlation between the human-expert estimated SP and the approximated development time is strong for only 7% of the projects investigated and medium (58%) or low (35%) for the remaining ones. Similar results are obtained when the actual development time

[1]Equivalent to effort in person-hour.

is considered. Our empirical study also reveals that the estimation made is often not consistent throughout the project, and the human estimator tends to misestimate in 78% of the cases. Our empirical results suggest that SP might not be an accurate indicator of open-source agile software development effort expressed in terms of development time. The impact of its use as an indicator of effort should be explored in future work, for example, as a cost driver in automated effort estimation models or as the prediction target.

6.1. INTRODUCTION

Software Effort Estimation (SEE) is a crucial activity for managing, planning, and monitoring software projects [1]. Without an accurate estimation of the effort required to develop software, budget and schedule overrun seem inevitable [8], [149]. SEE research has mainly focused on estimating the effort required to develop a whole project (i.e., project-level estimation). To this end, Functional Size Measures (FSM), such as Function Point (FP) [10], or COSMIC Function Point (CFP) [11], have been usually used as a cost driver to estimate traditional software development effort [1], [12]–[15].

The advent of Agile Software Development (ASD) methodologies [51] has shifted the focus towards estimating the effort of developing smaller units of software, like a new feature or change. In these cases, FSM methods are not easy to use [16] and another measure, namely Story Point (SP), has become popular in the context of ASD [18]. SP is a relative unit that represents an intuitive mixture of complexity and required effort of a user story (a.k.a. issue) [1], [17]. [2]

However, previous studies have shown that the accuracy of the SP estimate is sensitive to the practitioners' expertise and, thus, prone to bias. According to Usman et al. [150], who surveyed 60 engineers experienced in Agile Effort Estimation, the estimates of around half of the agile teams were inaccurate by a factor of 25% or more. Using inaccurate SP could result in iteration mismanagement and wrong prioritization of tasks, which in turn can lead to customer dissatisfaction or even project failure. Moreover, since human-expert estimated SP has been used as a cost driver to train automated estimation models [73]–[77], [151] or as a prediction target [21], [23]–[26], [28], [29], [62], [63], researchers and practitioners need to be aware if they are using inaccurate SP as this might impact the accuracy of these models.

Previous case studies have provided discordant results on whether SP can accurately capture software size and effort [16], [26], [78], [79], and to date, there is not enough empirical evidence on this matter. This work aims to fill this gap by carrying out a thorough large-scale empirical study investigating the extent to which using Story Point reflects the effort needed to develop a user story (i.e., issue development time).

[2] A user story is a user-valued functionality which is specified in the form of one or two sentences in the everyday language of the user.

To this end, we analyse 37,440 user stories coming from 37 agile software projects tracked with Jira [82], which are available in the TAWOS dataset [27]. To the best of our knowledge, this is the largest empirical study to date to investigate the relationship between SP and effort in agile open-source projects. In particular, we aim to answer the question *What is the relationship between the SP estimated for a given issue and its actual development time?* Since developers do not always record the actual time they spent on the development of an issue [27] in the issue tracking system, we compare three different proxies for the development time computed using the issue changelog[3] in order to answer the question: *To what extent can we approximate the actual development time as reported by the developers?* Furthermore, since one would expect issue development time to be proportionate to the story point assigned to a certain issue, we also aim to answer *How consistent is the assignment of SP throughout a project?*

The results of our empirical study show that among the three proxies, there is one which more closely reflects the development time as recorded by the developers, namely the InProgress development time. Moreover, we found that the correlation between this issue development time and human-expert estimated SP is medium or low for 93% of the projects we investigated. These results are in line with those we obtained using the recorded development time rather than the proxy, and they highlight that SP is not an accurate indicator of the software development effort. Moreover, we found that the human-expert estimation is not consistent throughout the projects. Although SP can remain useful for agile teams to organize and plan their iterations, these results raise awareness that the inaccuracy observed in the SP might be carried out into those automated effort estimation models that use SP as a cost driver to predict issue development time. Moreover, recent studies have proposed the use of machine learning approaches to predict SP for issues based on historical human-estimated SP. This means that such approaches learn to imitate human-expert estimations at best which in itself might be misleading of the actual effort needed to realise an issue. Further work is needed to understand the extent to which the use of inaccurate SP impact automated effort estimation models, and whether the use of development time (actual or proxies for it) can provide the engineers and managers with more reliable and accurate models.

The rest of this chapter is organised as follows. Section 6.2 provides some background for those readers who are not familiar with software size and effort measures, and issue tracking systems. Sections 6.3 and 6.4 present the design and results of our empirical study, respectively. Final remarks and future work are discussed in Section 6.6.

[3]The history of changes in the issue's attributes.

6.2. BACKGROUND

In this section, we briefly introduce the most common software functional size and effort measures proposed in the literature. We also give some background on Jira, the issue tracking system used by the projects analysed in this study, and describe three proxy measures for issue development time [27].

6.2.1. Software Size Measures

Albrecht was the first to introduce a disciplined method for measuring software product size, called Function Point Analysis (FPA), based on the functionality the software product is built to deliver to the customer [56]. Soon after, he showed that there is a strong correlation between Function Points and the final effort of a software [10]. Although FPA was designed to measure software from the domain of business applications [57], it is still widely applied in the software production industry [58].

COSMIC[4] Function Point (CFP) method belongs to the second generation of software functional size methods [11]. CFP also takes non-functional requirements into consideration and is suitable for a broader range of application domains including, but not limited to, business applications, web applications, mobile applications, real-time software, and service-oriented software [14], [15], [57], [152].

In the context of agile software development, practitioners have introduced and used Story Point (SP) as an agile-specific software size measurement unit [59]. Unlike FPA and CFP, SP does not follow a method of measurement; therefore, developers use them as a relative measure to keep the relative difference of stories in size by assigning a point value to each user story. One common approach to determine the story point value of a user story is to select one of the smallest stories in the Backlog[5] and assign it with one story point. Then the more complex and larger user stories get more points considering their size [59]. So any user story that is assigned two SPs is twice as large as a user story that is assigned one SP. SP estimations need to be consistent throughout the project.

6.2.2. Jira Workflow and Issue Development Time

Jira [82] is a widely-used issue-tracking system that supports agile development [153]. Using Jira, the development teams can record their estimated story point and the time taken for the development of the issue.

Although Jira Software has provided teams with specific fields to record the actual effort (i.e., time) spent on an issue, usually the developers do not use this feature to log their work. Thus, identifying the actual time spent to develop an issue might be

[4]Common Software Measurement International Consortium

[5]Backlog is a breakdown of work containing an ordered list of user stories that an agile team maintains for a project.

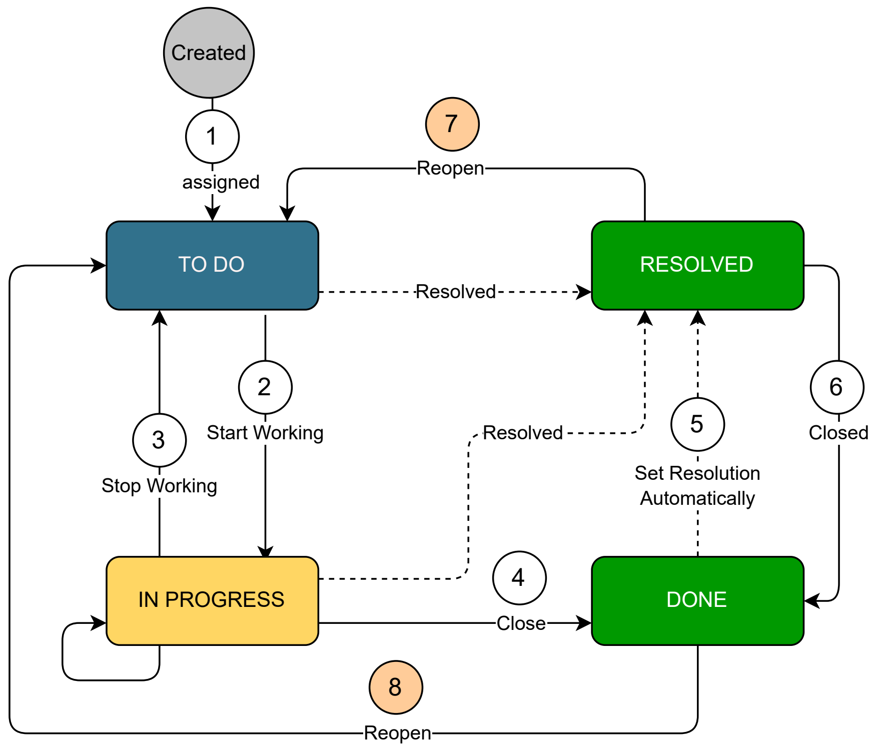


Figure 6.1: A generic Jira workflow.

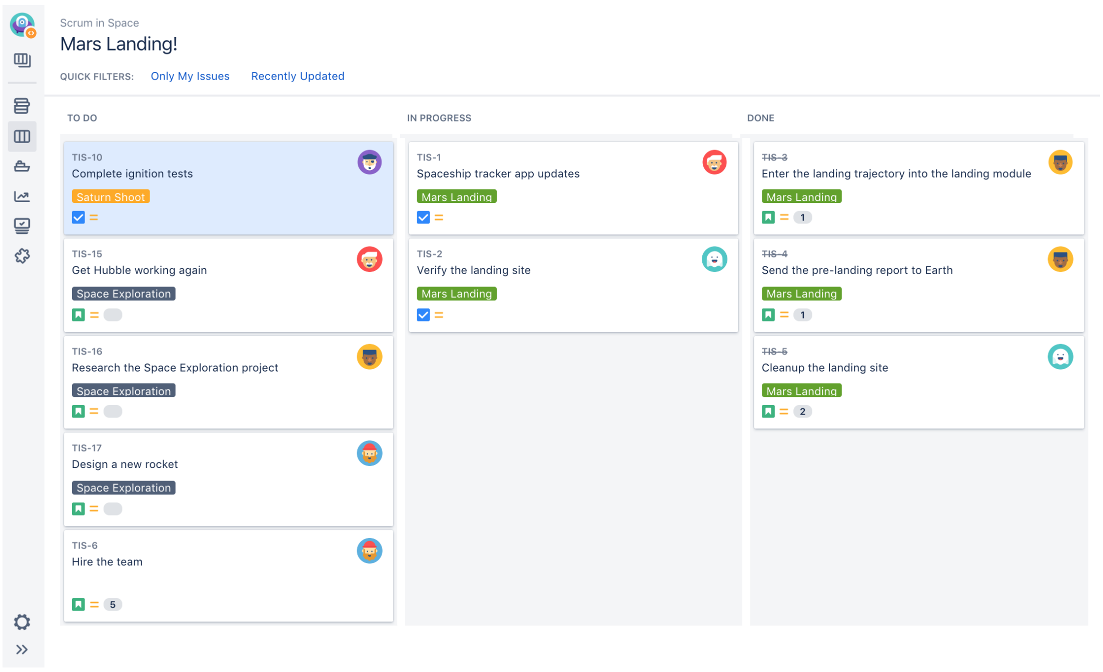
challenging. Nonetheless, previous work [26], [27] derived an approximation of the actual development time from the transitions recorded in the change log of the issues in the Jira repository.

Jira Workflow: In Jira projects, issues transition through stages of work —from creation to completion— following a path. This path is called workflow. Figure 6.1 shows a generic Jira workflow that could be used to track issue transitions and calculate approximate time spent on issue development.

The life cycle of an issue starts at the time of its creation (the grey circle in the workflow). When the issue is considered to be developed, its status changes into a *To Do* state (1). This is when the issue gets assigned to a developer. When the developer starts working on an issue, he/she changes its status to *In-Progress* (2). The developer is given the option to stop working (3) and restart it again (2) at any time.

When the development is finished, the developer changes the issue's status to *Done* (4), and Jira automatically populates the Resolution field (5). Also, the developer can set the Resolution field at any time during the life cycle of the issue (indicated by the dashed lines). The Resolution field can be populated with one of the several labels predefined in the workflow (usually but not necessarily with statuses defined within the *Done* category), e.g., Fixed, Completed, Closed, Delivered, Invalid, Duplicate, Won't Fix, Rejected, and Cancelled, depending on the project and issue type.

Jira recognises an issue as *Resolved* if the Resolution field is populated. By default


Scrum in Space

Mars Landing!

QUICK FILTERS: Only My Issues Recently Updated

TO DO

TIS-10
Complete ignition tests
Saturn Shoot

TIS-15
Get Hubble working again
Space Exploration

TIS-16
Research the Space Exploration project
Space Exploration

TIS-17
Design a new rocket
Space Exploration

TIS-6
Hire the team
5

IN PROGRESS

TIS-1
Spaceship tracker app updates
Mars Landing

TIS-2
Verify the landing site
Mars Landing

DONE

~~TIS-3~~
Enter the landing trajectory into the landing module
Mars Landing
1

~~TIS-4~~
Send the pre-landing report to Earth
Mars Landing
1

~~TIS-5~~
Cleanup the landing site
Mars Landing
2

Figure 6.2: A sample Jira task Board showing issues organised according to their status (i.e., To Do, In-Progress, or Done).

Resolved means that the issue is in a closed state and no more work is needed to be done. But in many workflows *Resolved* is not an ultimate state. For instance, in a custom workflow, once the issue is resolved there might be an inspection which decides if the solution provided is sufficient and/or corrects —the review process— before the issue can be closed (6). If the solution is not accepted, the issue will then be reopened and it would need to be addressed again (7). Ultimately, an issue might be reopened after it has been closed (8), although this is rare.

Issue Development Time: The workflow is typically specific to the work processes within an organization/team. Indeed, Jira provides organizations with the ability to create customised workflows and statuses for each project and issue type. This makes it difficult to create a general method to calculate the time by observing issue transitions. However, a custom status defined in a custom workflow has to belong to one of these three categories: *To Do*, *In-Progress*, and *Done*. Jira mandates the use of these categories and employs them internally to identify the column under which each issue should be listed in the software task board (Figure 6.2). Therefore, one can base the time calculation on these three categories. Specifically, using the status categories, one can identify the transitions of the issues between the time they were set to be in progress, stopped progress, or accomplished.

Based on the above workflow, we have defined the following three proxies for issue development time.

In-Progress Time is defined as *the duration in which an issue has been in the*

"In-Progress" status. In most projects, the "*In-Progress*" status is used by developers to mark the time that they spend on the implementation of an issue. Hence, In-Progress Time might not include any time spent on testing, reviewing or discussion.[6]

Effort Time is defined as *the duration in which an issue has been in any of the statuses categorised as* In-Progress. This definition can be interpreted as a more realistic proxy for the effort since it includes time spent for implementation, testing, reviewing, discussions, etc., as the Effort Time considers all the time that an issue spends under any status from the *In-Progress* category.[7]

Resolution Time is defined as *the duration required for an issue to be resolved* (equivalent to the elapsed time). As we can see in Figure 6.1, an issue status can be set to *Resolved* at any point in its life cycle. This definition aims at capturing the amount of time it takes for an issue to be resolved. To this end, we consider the duration between the time an issue was created until it is *Resolved* [27]. This is the definition used in the literature to measure the time to fix an issue [95]–[97]. This proxy slightly differs from the one used in the work of Choetkiertikul et al. [26], as it also takes into account the time between the creation of the issue and the first time it was set to an In-Progress status. Whereas the proxy used by Choetkiertikul et al. considers the duration between the time an issue was first set to an In-Progress status and the time that it was resolved.

6.3. EMPIRICAL STUDY DESIGN

In this section, we describe the research questions posed in our study and the dataset, methods and statistical tests used to answer these questions.

6.3.1. Research Questions

Story point, as a measure of effort, is expected to have a positive correlation with the actual time needed to realise a software.

In this study, we aim to investigate the correlation between the estimated story point and the actual effort. As the actual effort is rarely recorded in an issue report, we analysed three proxies for the development time based on the Jira workflow as described in Section 6.2.2. Therefore, our first research question assesses which of these proxies is a good approximation of the actual effort.

RQ1. Approximating Issue Development Time: *To what extent can proxies be*

[6]Note that there is always a status named *In-Progress* within the *In-Progress* category, teams can add other statuses into this category based on their issue ecosystem. For instance, in a project which has *In-Progress*, *Test*, and *Review* statuses defined in the *In-Progress* category, the issue may transition from one status to another until it passes all the required stages before it is closed (i.e., set to *Done*). This is showed by a recursive arrow for the *In-Progress* status in Figure 6.1.

[7]To identify the category of each status in each project, we queried the metadata of each project's repository by using the REST API provided by Jira [27].

used to approximate the development time logged by the developers?

Once we assess whether these proxies provide a satisfactory approximation, we move to investigate the correlation between the story point and the actual effort (i.e., each of the three proxies proposed):

RQ2. Correlation: *What is the relationship between an issue's story point and its development time?*

To answer this question, we use three widely known correlation statistics to verify the relationship between the SP and each of the three proxies we used for approximating the development time (see Section 6.2.2).

SP is a relative measure by definition, and relativeness refers to the amount of work that one story point represents. It can differ from project to project and from team to team. This rate (i.e., one story point ratio with respect to the amount of effort in person-hour) might be affected by aspects such as the experience of the team, the programming language and the technology used for development. This rate is used to compute the team's productivity and make the story point scale specific to each team. However, within a project, the estimation team should remain consistent throughout the project with respect to the unit of work that a story point represents. In other words, the amount of work considered for a story point should be kept the same until the end of the project, and all the issues should be measured with that same unit. Nevertheless, keeping this rate consistent is challenging for any team. This phenomenon justifies the rationale for our third and last research question, which emphasises the variance of the time for each SP:

RQ3. SP Consistency: *How consistent is the assignment of story points throughout a project?*

To answer this question, we rely on a visual representation to identify any deviation between the actual data and an ideal trajectory of consistency in SP estimation derived from the data itself, as further explained in Section 6.3.2.

6.3.2. Methodology

To answer RQ1, we compare the *Timespent* value with the time as measured by each of the proxies for each issue and compute the absolute error one would commit had a proxy been used rather than the actual value. Specifically, to measure the resemblance of the three proxy measures to *Timespent*, we compute the Sum of Absolute Errors (SAE) between each of the proxies and the *Timespent* for the issues contained in each project. SAE is computed as follows: $SAE = \sum_{i=1}^{n} |P_i - TS_i|$, where n is the number of issues in the project with reported *Timespent* values, TS_i is the *Timespent* value for issue i, and P_i is each of the values of the development time proxies for issue i, obtained from the TAWOS dataset [27]. The proxy with the minimum SAE is the most representative of the *Timespent*. Then we apply statistical significance tests on the distribution of each of the proxies against *Timespent* to verify whether the difference between them is statistically significant. Specifically, we used the Wilcoxon Rank-Sum test (a.k.a. Mann–Whitney U test) [154] to check for statistical differences. The confidence limit is initially set to $\alpha = 0.05$ and is corrected for multiple hypotheses using the standard Bonferroni correction (α/K, where K is the number of hypotheses). To answer RQ1, we tested the following null hypothesis:

Hypothesis 6.3.1: Null Hypothesis

The distribution of the *Timespent* is not different from that of the proxy P_i.

We perform a two-sided hypothesis test; thus, for those cases where the null hypothesis is rejected, the following alternative hypothesis is accepted:

Hypothesis 6.3.2: Alternative Hypothesis

The distribution of the *Timespent* is different from that of the proxy P_i.

To measure the effect size of the difference, we use Vargha Delaney's $\hat{A}_{12}$ measure, which is a standardised non-parametric effect size measurement, to assess how meaningful the difference between the two distributions is [133]. According to Vargha Delaney's effect size, if the two distributions are very similar $\hat{A}_{12} = 0.5$. Respectively, an $\hat{A}_{12}$ closer to 1 means that the two distributions are not similar. The effect size is considered negligible for $0.6 < \hat{A}_{12}$ (indicated by an 'N' beside its value), small (S) for $0.6 \leq \hat{A}_{12} < 0.7$, medium (M) for $0.7 \leq \hat{A}_{12} < 0.8$, and large (L) for $\hat{A}_{12} \geq 0.8$ (the same boundaries used in the previous chapters).

To answer RQ2, we apply three correlation statistics to our data: the Pearson r correlation coefficient [155], the Spearman's ρ rank correlation [156], and Kendall's τ rank correlation [157]. The Pearson correlation test measures the linear correlation between two variables, while Spearman's and Kendall's correlation tests are statistics used to measure the ordinal association between two samples and assess how well the relationship between two variables can be described using a monotonic function [158]. Unlike Pearson's r which considers the value of the data points, Spearman's ρ and Kendall's τ work with ranks of data points which makes them less sensitive to strong outliers that lie in the tails of both samples [159]. All three correlation statistics range from $+1$ to -1, where $+1$ indicates a perfect correlation and -1 indicates a perfect inverse correlation. A non-correlation is indicated by a 0. Although both Spearman's ρ and Kendall's τ measure rank correlation, they cannot be compared directly with one another since they have different scales. Gilpin [160] describes the ratio of ρ to τ to be almost 1.5 for most of the range. The two get close to each other as their magnitude increases towards the limits (i.e., both approaching $+1$ or -1) and when they both approach zero. We use Cohen's standard [146], [161] for interpreting the correlation coefficients to determine the strength of the relationship. Based on this, correlation coefficients between 0.10 and 0.29 represent a small association, coefficients between 0.30 and 0.49 represent a medium association, and coefficients equal to or greater than 0.50 represent a large association. To perform the correlation, we used the `cor.test` method available in `R` version $4.0.1$. Both Spearman's and Kendall's correlation statistic implementations used in this study can handle ties in the data points.

To answer RQ3, we group all the issues that have been assigned the same story point of value <*X*> (i.e., story point <*class X*>) in the same class. Then we analyse the boxplots of the time spent on each issue for the distinct classes to understand if the time distribution in story point classes is normal, which would indicate a normal distribution of the error for story point estimation in each class. To investigate consistency, we observe the median point in the distribution of development time per SP class. We use the median point as it is not affected by extreme values. For a project with inconsistent SP estimations, the median development time will be affected (misplaced from ideal trajectory) due to many miss-classifications of smaller or bigger tasks in a specific SP class; or, from another point of view, issues estimated to have the same SP do not agree on the same (or similar) development time. In an ideal scenario, the median of distribution of the development time in story point classes should have a linear relationship with the value of the story point. For instance, the median of the development time for issues assigned with story point five should be five times larger than the median of the development time for issues assigned with story point one. Should this linear relationship hold for all of the issues in story point classes, we can assert that story point estimations are consistent throughout a project. To test this, we show the trajectory of this linear relation to visualise the degree of consistency.

6.3.3. Data

We sample data from the TAWOS dataset version 1.0 [27] (Chapter 3. We used SQL queries to sample issues from this database[8]. Below we describe, in detail, how we sample the set of projects investigated in this study.

To answer our first research question (i.e., RQ1), we analyse those projects from the TAWOS dataset that have recorded the actual development time in the *Timespent* field of Jira. Hence, we selected all the resolved issues (i.e., we filter out all those that are not addressed) and have the *Timespent* field populated. Then, we removed all issues having a *Timespent* value lower than two minutes as done in previous work [163], to reduce noise in the data. After applying such filtering, we retained all those projects with at least 100 issues each. This resulted in a sample of 9,806 issues from 15 projects. Descriptive statistics of this set, which is used to answer RQ1, are provided in Table 6.1a.

As most of the issues recorded either one of the *Timespent* or SP, in order to answer

[8]The queries used to sample the data are publicly available in our online appendix [162].

RQ2 and RQ3, we also sampled another set of issues.[9] To this end, we filtered out from the TAWOS data all those issues that are not addressed and those that have been assigned with SP less than 1 and greater than 100, as done in previous work [26], to reduce the presence of data that is not relevant to the purpose of our empirical study. Moreover, we noticed that a considerable number of the issues in some of the projects have a proxy time equal to zero. After a careful manual inspection, we found that there are issues which never transitioned to an *In-Progress* status. Thus they had been closed immediately after being created or opened. These cases may correspond to issues where developers had already worked on the issue before tracking the corresponding record in Jira created for the mere purpose of recording issues. In order to reduce the bias introduced by these cases, and in accordance with the filter used for RQ1, we removed all the issues with *In-Progress time* in less than two minutes, which corresponded to a total of 34.47% of the issues sampled from the TAWOS dataset. Furthermore, we filtered out issues with outlying values of *In-Progress time* to minimise the effect of extreme values in our results.[10] We, therefore, retained projects with at least 100 issues after filtering out unwanted ones, which left us with 58.33% of the initially sampled data, corresponding to a total of 28,608 issues from 32 projects (equal to 44.11% of all the issues with recorded SP in 32 projects under investigation). To identify the outliers, we used the Interquartile Range (IQR). The IQR, which is equal to the difference between the 75th and 25th percentiles of the distribution of the data points, is multiplied by 1.5. The resulting value is subtracted from and added to the first and third quartiles, respectively, to get the lower and upper fences (a.k.a. Tukey fences). The data points falling outside the lower and upper fences are considered outliers and removed from the dataset. The resulting data has been used to answer RQ2 and RQ3. Descriptive statistics of this sampled dataset can be found in Table 6.1c. Note that a total of 621 issues from the open-source data sampled for RQ1 are also in the sample used for RQ2 and RQ3. Therefore, the total number of unique user stories sampled from the TAWOS dataset is equal to 37,440 extracted from 37 different open-source projects.

6.4. RESULTS

This section presents the results we have obtained in answering the research questions described in Section 6.3.1.

[9]Since RQ1 aimed at examining the approximation of the three proxies to the recorded *Timespent* values, and there, the analysis is independent of the SP values; therefore, we can use a different sample for RQ2 and RQ3 without loss of generality.

[10]Note that we did not filter out issues with regards to their development time proxy values from the sample used in RQ1, since the aim of RQ1 is to examine the approximation of the proxies to the recorded *Timespent* values.

Table 6.1: List of projects we analysed for RQ1 (a, and b) and RQs 2-3 (c). Each project's total number of issues is shown in the *Total Issues* column. *Before Filter* shows the original number of issues extracted from the TAWOS dataset [27] and *After Filter* shows the number of issues remaining after the filtering process as explained in Section 6.3.3. The other columns show summary statistics for SP, Timespent and its proxies.

(a)

Repository	Project	Key	Total Issues	Issues with Timespent		Timespent (minutes)				
				Before Filter	After Filter	Min	Max	Mean	Median	SD
Atlassian	Crowd	CWD	4,311	222	220	2	4,800	468.36	240	729.33
	Jira Software Cloud	JSWCLOUD	11,702	255	244	30	2,220	383.70	300	368.79
	Jira Software Server	JSWSERVER	12,862	262	257	30	3,600	394.01	180	535.28
	Jira Server	JRASERVER	44,165	990	981	5	24,622	298.01	120	941.64
	Bamboo	BAM	14,252	524	521	5	8,460	392.57	240	694.73
	Clover	CLOV	1,501	106	106	2	4,801	605.87	240	906.33
	FishEye	FE	5,533	634	612	2	8,782	265.36	112	638.94
Appcelerator	Titanium Mobile Platform	TIDOC	3,059	714	711	5	11,040	307.01	120	626.55
Lsstcorp	Data management	DM	26,506	191	190	5	24,000	934.08	480	1,847.11
Sonatype	Nexus	NEXUS	9,912	1,356	1,348	2	4,560	189.26	90	327.95
Talendforge	Talend Data Quality	TDQ	15,315	2,054	2,053	10	18,960	764.44	480	1,144.12
	Talend Data Preparation	TDP	5,670	219	193	10	6,660	723.85	300	1,027.93
	Talend Data Management	TMDM	9,137	1,650	1,648	3	5,760	541.33	360	625.69
	Talend Big Data	TBD	4,624	193	191	5	5,700	712.36	360	943.00
	Talend Enterprise Service Bus	TESB	15,985	436	178	60	2,760	268.66	180	289.30
Total			184,534	9,806	**9,453**					

(b)

Repository	Project	Key	In-Progress Time (minutes)					Effort Time (minutes)					Resolution Time (minutes)				
			Min	Max	Mean	Median	SD	Min	Max	Mean	Median	SD	Min	Max	Mean	Median	SD
Atlassian	Crowd	CWD	0	1,401,053	13,258.70	44.5	101,509.06	0	1,401,053	17,341.50	2,122.5	102,618.99	7	2,724,297	185,866.38	20,465.5	436,488.49
	Jira Software Cloud	JSWCLOUD	0	219,603	2,873.85	118.5	14,352.33	0	220,762	2,878.60	118.5	14,424.37	0	1,968,722	52,691.43	5,750	253,296.60
	Jira Software Server	JSWSERVER	0	192,687	4,030.39	0	17,679.50	0	192,687	4,030.39	0	17,679.50	0	1,434,770	46,633.84	5,998	127,654.65
	Jira Server	JRASERVER	0	216,201	2,167.55	0	12,139.01	0	248,570	3,345.93	0	18,248.28	0	4,387,661	145,363.86	11,644	416,982.46
	Bamboo	BAM	0	2,278,726	11,284.45	288	103,415.46	0	455,765	12,845.10	4,253	30,649.89	1	2,635,279	78,653.98	20,072	222,346.01
	Clover	CLOV	0	259,569	15,619.25	191.5	42,533.41	0	260,743	21,726.25	7,364.5	43,420.66	0	1,350,621	85,252.67	30,289.5	171,444.39
	FishEye	FE	0	4,771,423	14,706.04	94.5	195,220.17	0	4,771,423	38,411.01	11,178.5	257,383.61	6	4,797,823	167,097.20	34,210	536,889.66
Appcelerator	Titanium Mobile Platform	TIDOC	0	579,203	7,839.01	276	32,121.50	0	579,203	10,971.92	1,399	38,699.95	0	2,463,988	131,868.77	25,699	303,406.05
Lsstcorp	Data management	DM	0	1,695,993	29,921.76	1,445	145,880.95	0	1,849,990	54,467.54	8,724.5	222,795.99	0	1,850,092	72,405.15	19,166.5	223,627.89
Sonatype	Nexus	NEXUS	0	495,244	2,310.59	0	21,501.77	0	495,244	2,370.53	0	21,529.01	0	5,068,853	42,930.46	4,658	202,515.39
Talendforge	Talend Data Quality	TDQ	0	165,637	5,119.09	1,174	11,622.38	0	916,623	42,361.35	11,562	94,348.85	1	4,002,454	179,687.61	42,085	438,753.36
	Talend Data Preparation	TDP	0	137,132	7,632.41	1,604	15,791.27	0	162,678	25,879.64	16,936	29,052.18	1	903,339	106,585.19	60,215	141,153.18
	Talend Data Management	TMDM	0	1,031,320	5,019.47	1,459.5	27,701.47	0	1,231,065	43,492.23	19,125	87,717.66	0	2,902,016	98,187.55	27,414	228,291.61
	Talend Big Data	TBD	0	152,267	4,539.72	1,131	12,736.24	0	1,751,027	103,679.79	56,685	192,330.37	263	1,751,054	119,209.97	53,074	227,411.31
	Talend Enterprise Service Bus	TESB	0	2,391,610	16,848.47	1,472	179,097.87	0	2,391,610	31,680.99	1,490	241,500.85	0	2,480,801	25,164.78	5,380.5	186,888.34

(c)

Repository	Project	Key	Total Issues	# Issues with SP		Story Point					In-Progress Time (minutes)				
				Before Filter	After Filter	Min	Max	Mean	Median	StD	Min	Max	Mean	Median	StD
Atlassian	Jira Software Cloud	JSWCLOUD	11,702	318	185	1	20	4.19	3	3.58	2	21,931	5,212.30	2,945	5,267.85
	Confluence Server	CONFSERVER	42,324	662	362	1	13	3.03	3	1.73	2	24,847	4,511.35	1,559.5	5,922.60
	Jira Software Server	JSWSERVER	12,862	351	208	1	20	4.19	3	3.51	2	18,864	4,696.80	2,831	4,749.70
	Bamboo	BAM	14,252	528	302	1	20	2.47	2	2.18	2	20,524	4,104.67	1,455.5	5,048.14
	Clover	CLOV	1,501	387	146	1	20	3.48	2	4.09	2	30,293	5,700.54	2,848	6,655.53
Apache	Mesos	MESOS	10,157	3,272	1,157	1	13	3.32	3	2.08	2	39,861	6,311.93	1,616	8,922.99
	Usergrid	USERGRID	1,339	487	162	1	8	2.62	3	1.41	2	21,657	4,905.17	2,950.5	4,898.76
Appcelerator	Titanium Mobile Platform	TIDOC	3,059	1,297	628	1	40	4.28	3	4.14	2	46,257	6,771.88	1,553	10,171.09
	Aptana Studio	APSTUD	8,135	890	302	1	40	7.92	8	5.13	3	6,915	1,222.96	340	1,618.00
	Appcelerator Studio	TISTUD	5,979	3,406	1,918	1	34	5.69	5	4.14	2	5,243	817.76	182	1,159.13
	The Titanium SDK	TIMOB	22,059	4,665	1,753	1	21	5.56	5	2.70	2	14,255	1,927.80	240.5	3,137.66
	Appcelerator Daemon	DAEMON	313	242	131	1	99	9.57	8	11.03	2	26,184	3,403.00	379	6,141.23
DNN Tracker	DotNetNuke Platform	DNN	10,060	2,594	1,122	1	14	2.18	2	1.46	2	9,553	1,371.70	199	2,268.66
Hyperledger	Blockchain Explorer	BE	802	373	239	1	13	3.01	3	1.77	2	33,120	8,387.12	5,754	8,061.10
	Fabric	FAB	13,682	636	235	1	24	2.85	2	2.71	2	64,394	11,464.59	7,021	13,734.58
	Indy Node	INDY	2,321	681	438	1	13	3.21	3	1.73	2	38,453	9,357.24	7,190.5	8,693.04
	Sawtooth	STL	1,663	966	646	1	8	2.38	2	1.30	2	40,392	11,375.65	8,799	9,660.83
	Indy SDK	IS	1,531	720	418	1	13	3.91	3	2.13	2	22,384	4,986.39	2,917	5,339.56
Lsstcorp	Lsstcorp Data management	DM	26,506	20,664	9,019	1	100	6.16	3.2	9.58	2	105,126	18,290.56	8,083	24,287.98
Lyrasis	Lyrasis Dura Cloud	DURACLOUD	1,125	666	243	1	13	2.05	2	1.55	2	24,555	4,253.40	1,363	5,881.14
MongoDB	Compass	COMPASS	1,791	499	275	1	8	3.43	3	1.73	3	50,363	10,583.59	4,351	13,403.21
	C++ driver	CXX	2,032	224	105	1	4	1.35	1	0.65	2	14,917	2,458.21	1,098	3,587.64
	MongoDB Core Server	SERVER	48,663	784	418	1	42	2.53	2	2.76	2	18,577	3,081.26	1,056	4,419.22
	Evergreen	EVG	10,299	5,402	1,674	1	8	1.94	2	1.10	2	10,949	2,116.95	1,155	2,742.38
Mulesoft	Mule	MULE	11,816	4,170	2,105	1	21	4.95	4	3.52	2	22,982	5,195.96	3,179	5,394.67
	Mule APIkit	APIKIT	886	473	284	1	13	3.14	3	2.30	2	17,755	3,522.79	1,640	4,242.36
Sonatype	Nexus	NEXUS	9,912	1,845	421	1	15	1.56	1	1.22	2	11,467	2,054.11	518	2,811.87
Spring	XD	XD	3,707	3,705	1,602	1	24	3.37	3	2.48	2	20,942	4,160.57	1,680.5	5,038.44
Talendforge	Talend Data Quality	TDQ	15,315	1,843	1,151	1	40	5.20	5	4.28	2	27,457	6,161.18	3,945	6,663.57
	Talend Data Preparation	TDP	5,670	813	473	1	18	2.24	2	1.72	2	48,922	10,373.10	7,181	11,139.64
	Talend Data Management	TMDM	9,137	297	177	1	8	2.42	2	1.60	3	20,046	4,795.11	2,996	4,781.31
	Talend Enterprise Service Bus	TESB	15,985	1,000	309	1	13	2.28	2	1.51	2	47,937	11,166.68	8,370	11,427.15
Total			326,585	64,860	**28,608**										

Table 6.2: RQ1. Difference between *Timespent* and the three proxy measures for development time (i.e., In-Progress Time, Effort Time, and Resolution Time) in terms of Sum of Absolute Error (SAE) and significance statistical tests (effect size shown in brackets).

Project	SAE with Timespent (minutes)			Timespent vs.		
	In-Progress Time	Effort Time	Resolution Time	In-Progress Time	Effort Time	Resolution Time
CWD	2,909,699	3,796,762	40,790,437	<0.001 (0.40)	<0.001 (0.60)	<0.001 (0.95)
JSWCLOUD	666,291	667,450	12,776,147	**0.285** (0.47)	**0.285** (0.47)	<0.001 (0.80)
JSWSERVER	1,024,283	1,024,283	11,903,893	<0.001 (0.23)	<0.001 (0.23)	<0.001 (0.72)
JRASERVER	2,230,661	3,389,772	142,334,494	<0.001 (0.26)	<0.001 (0.25)	<0.001 (0.92)
BAM	5,784,351	6,565,644	40,777,437	**0.361** (0.52)	<0.001 (0.68)	<0.001 (0.96)
CLOV	1,641,549	2,261,458	8,973,869	**0.872** (0.49)	<0.001 (0.67)	<0.001 (0.93)
FE	8,961,125	23,375,131	102,102,251	**0.968** (0.50)	<0.001 (0.86)	<0.001 (0.99)
TIDOC	5,501,120	7,703,434	93,548,470	**0.085** (0.53)	<0.001 (0.61)	<0.001 (0.96)
DM	5,578,740	10,194,085	13,588,070	**0.043** (0.56)	<0.001 (0.77)	<0.001 (0.91)
NEXUS	3,114,794	3,193,031	57,646,533	<0.001 (0.31)	<0.001 (0.32)	<0.001 (0.85)
TDQ	9,568,980	85,743,427	367,355,182	<0.001 (0.54)	<0.001 (0.82)	<0.001 (0.96)
TDP	1,358,593	4,863,758	20,432,471	<0.001 (0.67)	<0.001 (0.91)	<0.001 (0.98)
TMDM	7,521,187	70,814,422	160,926,609	<0.001 (0.65)	<0.001 (0.92)	<0.001 (0.96)
TBD	784,328	19,667,618	22,633,044	**0.020** (0.57)	<0.001 (0.99)	<0.001 (0.99)
TESB	2,963,686	5,602,799	4,438,478	<0.001 (0.70)	<0.001 (0.72)	<0.001 (0.80)

6.4.1. RQ1. Approximating Issue Development Time

Table 6.2 shows the SAE values computed for the three proxies with respect to the *Timespent* value. We can observe that, among the three proxies under study, In-Progress Time has the smallest error for all projects. However, while this indicates that In-Progress Time is the most representative of *Timespent*, the magnitude of the absolute errors shows that all three proxies have large differences with *Timespent*. This is due to the fact that the *Timespent* field, which stores the aggregated amount of time spent on the development of the issue, is computed as the sum of the work hours logged by the developers on the issue. The proxies obtained from the TAWOS dataset are the aggregation of the duration between points in the timeline for an issue's status change, so they take into account the idle time that a developer might pause working but not change the status. For example, if an issue's status remains unchanged for a week but a developer works five hours a day on the task, they may log 25 hours for *Timespent*, while the proxies would take into account the number of days (in progress, or to resolution) in order to measure development time. Hence, the proxy may be multiple times greater than the actual *Timespent*. However, this difference in magnitude does not affect the correlation results.

Overall, considering all limitations of getting a close approximation of the actual effort in open source projects, these proxies are the most representative we could obtain from the data.

In order to verify whether the differences between *Timespent* and each of the three proxies are statistically significant, we revert to the Wilcoxon test. The results of this test are presented in Table 6.2 (last three columns). Since we are interested in any difference between each pair of distribution sets, we use a two-sided alternative hypothesis, therefore a comparison of P_i vs P_j results in the same p-value as the comparison of P_j vs P_i. As revealed by the results, the difference in the distributions of *Timespent* and In-Progress is significant in eight out of 15 projects, all with small or negligible effect size, except for the TESB project, for which the effect size is medium. However, the difference between *Timespent* and the Effort time proxy is significant in 14 out of 15 cases, with JSWCLOUD being the only exception given that the recorded Effort time is very close to In-Progress time in most of the issues belonging to this project. Moreover, the difference is significant for all the cases when comparing *Timespent* with the Resolution time. Out of 30 cases of statistical tests on Effort Time and Resolution Time, 19 cases show a large effect size, three cases a medium effect size, and 7 cases a small or negligible effect size.

As a result, we only consider In-Progress in our subsequent research questions given that it is the most representative of *Timespent* compared to the other two proxies.

6.4.2. RQ2. Correlation

The results of three correlation statistics (RQ2) are shown in Table 6.3. We also reported the p-value for each correlation coefficient.

For all projects, Kendall's τ is consistent with Spearman's ρ in the scale and confidence level. However, if we consider Gilpin's τ to ρ conversion table [160], we would expect a higher ρ. For example, in the case of the CONFSERVER project (see Table 6.3), Gilpin's table maps a $\tau = 0.26$ to $\rho = 0.38$, while our data lead to a $\rho = 0.34$. The rationale behind this is the fact that Kendall's τ is the proportion of the concordant to discordant pairs while Spearman's ρ considers the variance in the ranks. Hence, as we obtain a ρ smaller than expected (indicated by ρ to τ rate) it shows the high variance in the ranks of the data, to which Spearman is sensitive, but Kendall is not. This high variance in the ranks is a sign of misclassification of many issues by human estimators in wrong SP classes, thus an error in the estimation.

The Pearson correlation coefficient is lower than Spearman's ρ for 24 projects (75% of the cases), indicating that the relationship between the story point and development time is not usually linear.

As we can observe, the correlation denoted by Spearman's ρ for In-Progress time is low in six out of 32 cases, medium in 21 cases and strong for only five cases. The strongest positive correlation appears to be in projects DAEMON, INDY, JSWCLOUD, JSWSERVER, and DURACLOUD. Looking at the p-value of Spearman's ρ, we find that the confidence level is above 99% for 30 out of the 32 projects under study (94% of the cases).

As a subsequent analysis, we computed the three correlation statistics on all the issues from the TAWOS dataset that have reported both the SP and *Timespent* values. This resulted in a total of 697 issues from four projects (specifically, 128 issues from DM, 303 issues from TDQ, 104 issues from TMDM, and 162 issues from MDL). Although

Table 6.3: RQ2. Correlation results between SP and three development time proxies (p-value in brackets). Medium and strong correlations are highlighted in orange and red, respectively.

Project	In-Progress Time Correlation with Story Point			Effort Time Correlation with Story Point			Resolution Time Correlation with Story Point		
	Spearman's ρ	Kendall's τ	Pearson r	Spearman's ρ	Kendall's τ	Pearson r	Spearman's ρ	Kendall's τ	Pearson r
JSWCLOUD	0.54 (<0.001)	0.41 (<0.001)	0.47 (<0.001)	0.30 (<0.001)	0.23 (<0.001)	0.09 (0.238)	0.20 (0.006)	0.14 (0.008)	-0.05 (0.513)
CONFSERVER	0.34 (<0.001)	0.26 (<0.001)	0.26 (<0.001)	0.10 (0.064)	0.07 (0.064)	-0.03 (0.571)	0.12 (0.020)	0.09 (0.023)	-0.02 (0.681)
JSWSERVER	0.53 (<0.001)	0.40 (<0.001)	0.49 (<0.001)	0.29 (<0.001)	0.21 (<0.001)	0.09 (0.190)	0.21 (0.002)	0.15 (0.002)	0.02 (0.775)
BAM	0.35 (<0.001)	0.28 (<0.001)	0.35 (<0.001)	0.21 (<0.001)	0.16 (<0.001)	0.12 (0.036)	0.00 (0.949)	0.00 (0.958)	-0.02 (0.692)
CLOV	0.45 (<0.001)	0.34 (<0.001)	0.44 (<0.001)	0.43 (<0.001)	0.33 (<0.001)	0.39 (<0.001)	0.09 (0.289)	0.07 (0.269)	-0.03 (0.704)
MESOS	0.40 (<0.001)	0.30 (<0.001)	0.35 (<0.001)	0.38 (<0.001)	0.29 (<0.001)	0.10 (<0.001)	0.16 (<0.001)	0.12 (<0.001)	0.04 (0.225)
USERGRID	0.20 (0.013)	0.16 (0.008)	0.12 (0.139)	0.20 (0.013)	0.16 (0.008)	0.12 (0.139)	-0.04 (0.656)	-0.02 (0.722)	0.02 (0.827)
TIDOC	0.48 (<0.001)	0.36 (<0.001)	0.27 (<0.001)	0.49 (<0.001)	0.36 (<0.001)	0.07 (0.078)	0.08 (0.034)	0.07 (0.021)	-0.07 (0.094)
APSTUD	0.38 (<0.001)	0.30 (<0.001)	0.40 (<0.001)	0.38 (<0.001)	0.29 (<0.001)	0.40 (<0.001)	0.18 (0.002)	0.14 (0.001)	0.12 (0.042)
TISTUD	0.42 (<0.001)	0.33 (<0.001)	0.35 (<0.001)	0.35 (<0.001)	0.27 (<0.001)	0.07 (0.005)	0.17 (<0.001)	0.13 (<0.001)	0.05 (0.036)
TIMOB	0.28 (<0.001)	0.22 (<0.001)	0.23 (<0.001)	0.13 (<0.001)	0.10 (<0.001)	0.02 (0.419)	0.13 (<0.001)	0.10 (<0.001)	0.05 (0.019)
DAEMON	0.62 (<0.001)	0.48 (<0.001)	0.66 (<0.001)	0.62 (<0.001)	0.47 (<0.001)	0.21 (0.018)	0.42 (<0.001)	0.31 (<0.001)	0.07 (0.435)
DNN	0.32 (<0.001)	0.25 (<0.001)	0.27 (<0.001)	0.29 (<0.001)	0.22 (<0.001)	0.03 (0.264)	0.00 (0.927)	0.00 (0.964)	-0.06 (0.042)
BE	0.19 (0.003)	0.15 (0.002)	0.21 (0.001)	0.29 (<0.001)	0.22 (<0.001)	0.11 (0.077)	0.28 (<0.001)	0.20 (<0.001)	0.08 (0.246)
FAB	0.49 (<0.001)	0.38 (<0.001)	0.36 (<0.001)	0.50 (<0.001)	0.38 (<0.001)	0.21 (0.001)	0.49 (<0.001)	0.37 (<0.001)	0.20 (0.002)
INDY	0.56 (<0.001)	0.44 (<0.001)	0.53 (<0.001)	0.45 (<0.001)	0.35 (<0.001)	0.28 (<0.001)	0.14 (0.004)	0.10 (0.004)	0.16 (0.001)
STL	0.41 (<0.001)	0.32 (<0.001)	0.39 (<0.001)	0.40 (<0.001)	0.31 (<0.001)	0.08 (0.042)	0.34 (<0.001)	0.26 (<0.001)	0.12 (0.003)
IS	0.49 (<0.001)	0.38 (<0.001)	0.43 (<0.001)	0.38 (<0.001)	0.28 (<0.001)	0.26 (<0.001)	0.25 (<0.001)	0.19 (<0.001)	0.10 (0.046)
DM	0.49 (<0.001)	0.36 (<0.001)	0.42 (<0.001)	0.46 (<0.001)	0.34 (<0.001)	0.18 (<0.001)	0.42 (<0.001)	0.31 (<0.001)	0.17 (<0.001)
DURACLOUD	0.52 (<0.001)	0.41 (<0.001)	0.47 (<0.001)	0.52 (<0.001)	0.41 (<0.001)	0.46 (<0.001)	0.29 (<0.001)	0.23 (<0.001)	0.09 (0.156)
COMPASS	0.30 (<0.001)	0.23 (<0.001)	0.25 (<0.001)	0.24 (<0.001)	0.18 (<0.001)	0.21 (<0.001)	0.14 (0.024)	0.10 (0.023)	0.07 (0.223)
CXX	0.27 (0.005)	0.22 (0.006)	0.36 (<0.001)	0.32 (0.001)	0.26 (0.001)	0.00 (0.998)	0.01 (0.906)	0.01 (0.906)	0.07 (0.457)
SERVER	0.49 (<0.001)	0.37 (<0.001)	0.29 (<0.001)	0.51 (<0.001)	0.39 (<0.001)	0.15 (0.002)	0.43 (<0.001)	0.33 (<0.001)	0.06 (0.218)
EVG	0.36 (<0.001)	0.28 (<0.001)	0.28 (<0.001)	0.34 (<0.001)	0.26 (<0.001)	0.07 (0.007)	0.28 (<0.001)	0.22 (<0.001)	0.11 (<0.001)
MULE	0.48 (<0.001)	0.36 (<0.001)	0.49 (<0.001)	0.48 (<0.001)	0.36 (<0.001)	0.47 (<0.001)	0.10 (<0.001)	0.07 (<0.001)	-0.05 (0.035)
APIKIT	0.37 (<0.001)	0.28 (<0.001)	0.30 (<0.001)	0.35 (<0.001)	0.26 (<0.001)	0.17 (0.005)	0.28 (<0.001)	0.21 (<0.001)	0.07 (0.266)
NEXUS	0.25 (<0.001)	0.19 (<0.001)	0.25 (<0.001)	0.16 (0.001)	0.12 (0.001)	0.08 (0.083)	0.23 (<0.001)	0.18 (<0.001)	0.08 (0.115)
XD	0.41 (<0.001)	0.31 (<0.001)	0.38 (<0.001)	0.36 (<0.001)	0.27 (<0.001)	0.09 (<0.001)	0.28 (<0.001)	0.21 (<0.001)	0.08 (0.003)
TDQ	0.44 (<0.001)	0.32 (<0.001)	0.36 (<0.001)	0.27 (<0.001)	0.19 (<0.001)	0.13 (<0.001)	0.11 (<0.001)	0.08 (<0.001)	-0.04 (0.144)
TDP	0.36 (<0.001)	0.27 (<0.001)	0.23 (<0.001)	0.25 (<0.001)	0.19 (<0.001)	0.13 (0.004)	0.16 (0.001)	0.12 (0.001)	0.06 (0.220)
TMDM	0.18 (0.016)	0.14 (0.015)	0.23 (0.002)	0.16 (0.031)	0.12 (0.029)	0.09 (0.212)	-0.08 (0.267)	-0.06 (0.301)	-0.01 (0.848)
TESB	0.33 (<0.001)	0.26 (<0.001)	0.24 (<0.001)	0.04 (0.507)	0.03 (0.510)	-0.05 (0.414)	0.14 (0.013)	0.11 (0.009)	0.00 (0.965)
Min	0.18	0.14	0.12	0.04	0.03	-0.05	-0.08	-0.06	-0.07
Max	0.62	0.48	0.66	0.62	0.47	0.47	0.49	0.37	0.20
Mean	0.40	0.31	0.35	0.33	0.25	0.15	0.19	0.14	0.05
SD	0.11	0.08	0.11	0.14	0.10	0.13	0.14	0.11	0.07

Table 6.4: RQ2. Correlation results between SP and *Timespent* (p-value in brackets). Medium and strong correlations are highlighted in orange and red, respectively.

Project	*Timespent* Correlation with Story Point		
	Spearman's ρ	Kendall's τ	Pearson r
DM	0.33 (<0.001)	0.28 (<0.001)	0.35 (<0.001)
MDL	0.34 (<0.001)	0.27 (<0.001)	0.11 (0.179)
TDQ	0.64 (<0.001)	0.52 (<0.001)	0.61 (<0.001)
TMDM	0.03 (0.729)	0.03 (0.688)	0.28 (0.004)

the number of such issues is not prevalent, it indicates how much the results of RQ2 can be resembled by actual development time values (i.e., *Timespent*).

The result of this correlation analysis is shown in Table 6.4. As we can see, only one out of four projects showed a strong correlation with respect to all three statistics. Of the other three projects, two show a medium range Spearman's ρ and one a medium range Pearson's r coefficient. For the rest of the cases (i.e., 50%), a low correlation is obtained between SP values and actual *Timespent*.

6.4.3. RQ3. Consistency

Figure 6.3 shows the boxplots of the development time distribution for each story point value for six sample projects.[11]. As previously explained (Section 6.3.2), for an acceptable story point estimation error, each of these boxplots should resemble a normal distribution, and the relation between the median of each box should be proportional to the value of the story point class. The ideal projection of median development time per each SP class is depicted by a line connecting the diamonds in Figure 6.3. This projection is computed by multiplying the SP value with the median development time for all issues estimated to have one story point.

From the boxplots, we can observe that this proportion does not hold for most of the classes. Besides, the distribution of each class tends to be heavily tailed. This observation is confirmed by the Shapiro-Wilk test [164], which has revealed that the data is not normally distributed for any of the projects [162]. Specifically, only for seven projects out of the 32 under study, the median development time per SP class (depicted by the median line inside each boxplot) falls in the vicinity of the ideal projected value (for example, see projects CLOV and JSWCLOUD in Figure 6.3). For four other projects, the projection line falls well above the actual median development time (e.g., the BE project in Figure 6.3), indicating an overestimation. While for the remaining 21 projects, the projection line falls well below the actual median development time (e.g., the APIKIT, XD and MULE projects in Figure 6.3). This high number shows the tendency for human

[11]Due to space, the boxplots for all projects can be found in our online appendix [162].

Table 6.5: RQ3. Angles created between the X axis and the linear regression fit for SP classes against Median In-Progress time when only SP classes ≤ 5 are considered (Angle (SP≤ 5)) and when all the classes are considered (Angle (SP≤ 100)), and the angle between the two (Difference).

Project	Angle (SP≤ 5)	Angle (SP≤ 100)	Difference	Project	Angle (SP≤ 5)	Angle (SP≤ 100)	Difference
JSWCLOUD	40.41°	24.22°	**16.19**°	STL	65.47°	54.82°	10.65°
CONFSERVER	48.35°	20.77°	**27.58**°	IS	46.96°	25.49°	**21.47**°
JSWSERVER	41.12°	24.48°	**16.64**°	DM	75.48°	26.95°	**48.53**°
BAM	57.40°	21.57°	**35.83**°	DURACLOUD	56.49°	46.08°	10.41°
CLOV	57.51°	22.65°	**34.86**°	COMPASS	47.02°	59.73°	**12.71**°
MESOS	54.04°	39.22°	**14.82**°	CXX	64.53°	64.53°	0.00°
USERGRID	26.17°	38.78°	**12.61**°	SERVER	30.39°	7.43°	**22.96**°
TIDOC	23.57°	23.73°	0.16°	EVG	36.15°	24.12°	12.03°
APSTUD	-4.67°	5.71°	10.38°	MULE	29.20°	31.42°	2.22°
TISTUD	0.72°	7.78°	7.06°	APIKIT	19.81°	20.73°	0.92°
TIMOB	1.45°	6.79°	5.34°	NEXUS	17.80°	23.40°	5.60°
DAEMON	1.40°	11.14°	9.74°	XD	31.91°	17.83°	**14.08**°
DNN	26.08°	7.92°	**18.16**°	TDQ	50.19°	11.11°	**39.08**°
BE	48.00°	45.96°	2.04°	TDP	63.45°	-0.90°	**64.35**°
FAB	78.33°	19.39°	**58.94**°	TMDM	32.77°	37.86°	5.09°
INDY	66.94°	63.87°	3.07°	TESB	69.70°	7.97°	**61.73**°

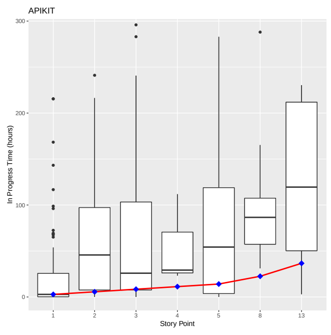
experts to generally underestimate the time required to complete a certain task.

We further analyse this phenomenon by fitting a regression line to each class's median development time against the story point's value. We then fit another regression line considering only the classes of SP with values less than five. As the angle between these two lines widens, the consistency between story point classes becomes lower. In contrast, if these two lines are aligned together for a project, we can say that the consistency of estimation in lower SP classes is maintained for higher SP classes. This is based on the premise that human experts are better at estimating smaller tasks than bigger ones. Plots of the regression fit for all projects can be found in our online appendix [162].

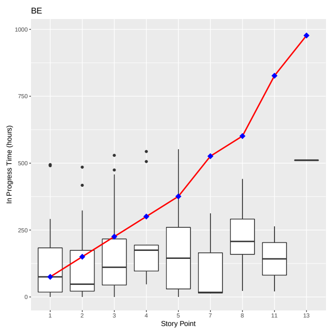
We also report, in Table 6.5, the angles each of these lines creates with the x-axis as well as the deviation in the trajectory (i.e., the difference between the two angles).

We can observe that the angle between the two lines is wider than 12.5° for more than half the projects (53% of the cases). This signifies that there is a notable shift in the trajectory of the regression fit of those projects taking into account the median development time for the higher SP classes. Thus, the scale in which the issues in the higher SP classes are estimated is inconsistent with those in the smaller SP classes. Therefore, based on these observations, we recommend that development teams consider breaking down bigger issues into smaller ones before they attempt to estimate the story point.

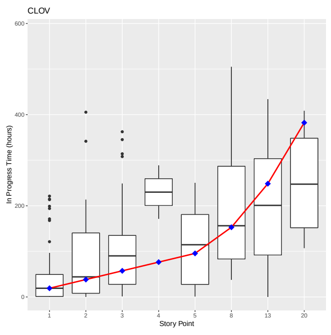
It is also worth noting that using data consisting of estimated SP to train a predictive model would result in that model imitating human expert misestimates and, therefore, possibly achieving biased results.



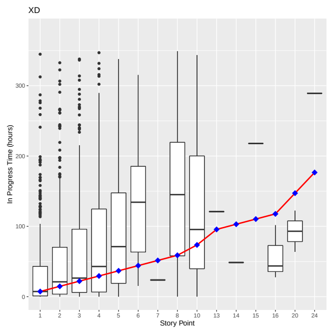
(a) APIKIT



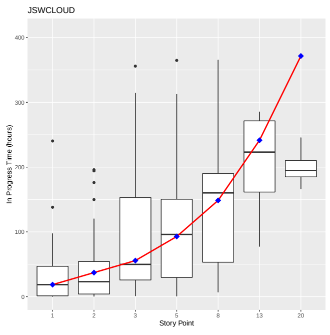
(b) BE



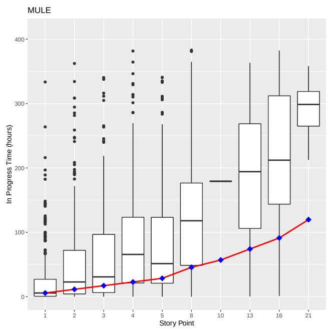
(c) CLOV



(d) XD



(e) JSWCLOUD



(f) MULE

Figure 6.3: Boxplots of the distribution of development time per SP class for (a) APIKIT, (b) BE, (c) CLOV, (d) XD, (e) JSWCLOUD, (f) MULE. The red line depicts a project-specific baseline, drawn based on the median development time for one SP.

6.5. THREATS TO VALIDITY

To mitigate construct validity threats we use data from real-world projects, which have been carefully curated and used in previous work [27]–[29]. The story point values are predicted by human experts and recorded in the Jira issue repository. However, the values we use as the actual time are extracted from the issue change-log, based on the issue transitions recorded in the repository throughout the development process. We are aware that these time values might not accurately represent the actual effort spent on developing an issue, and we mitigated this threat by considering three different approximations (i.e., In-Progress time, Effort time, and Resolution time), each capturing different aspects of the actual effort. We also used the actual development time recorded for issues to investigate the extent to which these proxies resemble the actual development time. Data points which are likely to be noisy, such as issues that have not been fully resolved or issues with less than 2 minutes of recorded development time, were filtered out from the dataset before any analysis, as described in Section 6.3.3.

Using proxies of the development time instead of the actual time might also be a threat to the internal validity of this study. In other words, the low correlation between SP and the proxies might be because of an unrepresentative proxy and not the expert misestimation. We mitigate this threat by conducting the same correlation analysis with the actual development time recorded by the developers where this was available.

With regard to the conclusion validity, we used three well-known correlation statistics and reported the corresponding p-value. To investigate the similarity of the proxies to the actual development time, we used the Sum of Absolute Errors and examined the statistical difference between the absolute error distributions by applying the Wilcoxon Rank-Sum test with all required assumptions checked, following best practice for effort estimation studies [119].

To mitigate external validity threats we used a large set of 37 projects which differ in size, application domain, programming language, and development team. Although we used such a diverse dataset, all the projects are open-source and the results might not be generalizable to other contexts.

6.6. CONCLUSIONS

We have studied the relationship between human-expert estimated Story Point (SP) and the time required by the developers to realise a given issue (i.e., *development time*) on a large sample of open-source user stories sampled from the TAWOS public dataset [27], which consists of 37 software projects for different application domains, diversified in size and characteristics, resulting in a total of 37,440 unique issues.

The results of this empirical study showed that among the three proxies for development time studied herein, In-Progress time is the most representative of the development time recorded by the developers. When considering its correlation with human-expert estimated SP, we found that for the majority of the projects, such a correlation is low (35%) or medium (58%). Analysing the correlation between SP and the actual development time unveiled a similar outcome: SP showed a low (50%) or medium (25%) correlation with *Timespent*. We also found that the majority of the investigated projects (25 out of 32) lack consistent human-expert estimations for SP. The consistency starts to wear when the issues are estimated to be bigger than five points, thereby suggesting that human estimators are not accurate at assessing the size of the issues that need five times or more effort than an issue worth a single story point. To overcome this issue, agile teams can try to break down all tasks/issues estimated to be bigger than five SP into smaller ones.

The above results provide empirical evidence that human-expert estimated SP might not be a good indicator for the issue development effort of agile open-source projects. This might render any machine-learnt effort estimation model, which learns from human-expert estimated SP, vulnerable to the same bias, and its impact should be taken into account in future work. It would be interesting, for example, to assess if more accurate effort estimation models can be obtained by using the development time instead of SP as a cost driver. Moreover, future work could replicate our study by considering industrial projects to expand the understanding beyond the open-source realm investigated herein. Also, future work could involve expert-certified FSM measurers to compute the FP and CFP of the user stories available in the TAWOS dataset so that one could carry out a large-scale empirical study analysing the correlation between SP, FP, CFP, and actual development time.

In order to allow for replication and extension of our work, we make our data and scripts publicly available [162].

Chapter 7

Predicting Issue Development Time

Abstract – In Chapter 6, we saw that story points are not a strong indicator of actual software effort. Therefore, their usage as the ground truth to build predictive models should be dealt with with caution. As an alternative, an AI-based effort estimation model may leverage the development time of resolved tasks instead of their story points. In this chapter, I aim to use the development time of agile tasks to train effort estimation models. I build these models using the same approaches developed for story point estimation and evaluated in previous chapters of this thesis. Specifically, I use GPT2SP, Deep-SE, LHC-SE, and TF-IDF-SE and the three baseline estimators (Median, Mean, and Random Guessing). Results show that all four AI-based methods can estimate the development time with an average absolute error of less than 8.5 hours for 11 out of 15 (73%) projects investigated.

7.1. INTRODUCTION

Although story points may be useful for agile teams to manage sprints and prioritize user stories, they are not a strong indicator of the development time needed to deliver the user stories. Biased with their experience, the human estimators, be it as an individual or in a group, tend to under- or overestimate most of the time [30]. Therefore, relying on a human estimator's output to build automated estimation models will most likely lead to a biased model that imitates human bias.

This chapter investigates the predictive performance of the state-of-the-art effort estimation models when they are trained on and used to estimate the development time (equivalent to effort in person-hour) instead of story points. To this end, four Machine-Learning (ML) based models are adapted to carry out the experiments. Three of the models are those used in previous chapters to estimate story points (namely, TF-IDF-SE [23], and Deep-SE [26] from Chapter 4, and LHC-SE [29] from Chapter 5). The other method included in the experiments is GPT2SP, a transformer-based deep learning method which has been published more recently [21]. Details about this new method are provided in Section 7.2. To be consistent in naming the methods and to reflect the modification perform on GPT2SP, I refer to the modified variant as **GPT2-SE** in this chapter, with SE standing for Software Effort estimation. The three baseline estimators (Random Guessing, Mean and Median baselines) are also included in experiments to measure the performance of the four ML models against simple baselines.

The rest of this chapter is organised as follows. Section 7.2 briefly introduces GPT2SP and its architecture. Section 7.3 describes the study's design, including the research questions, the dataset and the experimental method used to address them. Section 7.4 reports and discusses the results, and Section 7.5 discusses the threats to the validity of the study. Finally, Section 7.6 concludes the chapter and provides suggestions for future work.

7.2. BACKGROUND

This section provides a brief introduction to GPT2SP and its architecture.

Fu and Tantithamthavorn [21] have recently proposed GPT2SP, a Transformer-based deep learning model for SP estimation of user stories. They introduce three structural and design improvements over Deep-SE: First, instead of word-level tokenization, GPT2SP employs a byte- pair-encoding subword tokenization which splits rare words into subword units, reducing the vocabulary size to almost one-fourth. Second, unlike Deep-SE, which needed pre-training[1] on project-specific data, GPT2SP is already a pre-trained model that can generate meaningful embedding for any project. And third, GPT2SP uses GPT-2 architecture [165] with a masked multi-head self-attention mechanism [166], allowing it to capture the relationship among words better while considering the context of a given word and its position in the sequence.

Figure 7.1 shows an overview of GPT2SP's architecture. Given an issue report, GPT2SP performs sub-word tokenisation using a byte-pairs encoding (BPE) approach (Step 1). In this step, GPT2SP leverages the GPT-2 pre-trained language model to produce subword-tokenized issues. Then, GPT2SP performs a word and positional encoding in order to generate an embedding vector of each word and its position in the issue and then uses GPT-2's stacking transformer decoder architecture to output the issue report as a vectorised representation. The vectorised output is then fed to a Multi-Layer Perceptron (MLP) to estimate the story point for the given issue (Step 2).

Fue and Tantithamthavorn [21] empirically evaluated the performance of GPT2SP on Choetkiertikul's dataset [26], including 16 projects with a total of 23,313 issues (details in Section 4.3.2.1). They benchmarked GPT2SP against two baselines (namely the Mean and Median estimators) and Deep-SE for both within- and cross-project

[1]Note that in Chapter 4 we observed that pre-training had a non-significant effect on the estimation performance of Deep-SE.

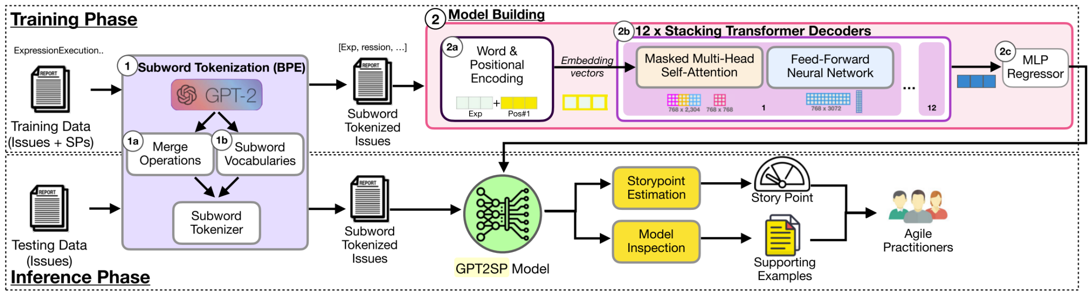


Figure 7.1: Architecture of GPT2SP [21].

estimation scenarios and evaluated the extent to which each component of GPT2SP contributes towards the accuracy of the SP estimates.

Their results show that GPT2SP outperforms Deep-SE with a 6%-47% improvement over MAE for the within-project scenario and a 3%-46% improvement for the cross-project scenarios. However, the attempt to use the GPT2SP source code made available by Fu and Tantithamthavorn to reproduce their experiments resulted in the discovery of a bug in the computation of the Mean Absolute Error (MAE), which may have inflated the GPT2SP's accuracy reported in their work. A pull request issued by the author of this thesis to fix such a bug was accepted and merged into their repository at `https://github.com/awsm-research/gpt2sp/pull/2`. Further details and results about this replication can be found at [64].

7.3. EMPIRICAL STUDY DESIGN

7.3.1. Research Questions

The intelligent agile effort estimation methods are designed to draw an analogy between the given task and similar tasks completed in the past to come to an estimation. Because of the mechanism they work with, these models should be able to be trained on and estimate the effort regardless of its unit. Therefore, by adapting their implementation to use development time instead of story points, they should be able to predict the development time. The first research question investigates this:

RQ1. Estimating Development Time: Are the ML-based models built to estimate story points suitable for estimating development time?

To answer this research question, four-story point estimation models (namely, TF-IDF-SE, Deep-SE, LHC-SE[2], and GPT2-SE) have been used to estimate the development time for issues from agile projects. Implementation and configurations for all four algorithms are kept unchanged from the original setups. However, the target variable is changed from story points to the development time (in hours). The mean and median absolute errors produced by each of these models are compared against

[2]I use LHC$_{TC}$-SE from Chapter 5 as the best performing LHC-SE variant.

each other and that of the baseline estimators (i.e., Random Guessing, Mean and Median).

Although story points are shown to be biased and not a strong indicator of the development effort, they may serve as an auxiliary cost driver to it. In a further investigation, I analyse the effect (positive or negative) story points can have on development time estimation, should they be used as a cost-driver alongside the input features:

RQ2. SP as another Cost-Driver: Does using SP as another cost-driver alongside the other input features improve the estimation accuracy of automated agile effort estimation models?

To answer this research question, I edit the estimation model's implementation in order to input the story point of the target issue as another independent variable besides the issue text and evaluate its estimation performance in comparison with the original model.

7.3.2. Data

I sample data from the TAWOS dataset version 1.0 [27] (Chapter 3). To answer RQ1, I use the same set of projects and issues we used to answer RQ1 from Chapter 6, as this sample includes only resolved issues with recorded development time in the Timespent field of Jira, with a Timespent value of more than two minutes (Section 6.3.3 in Chapter 6). This dataset has 9,806 issues from 15 projects. Descriptive statistics of this set are provided in Table 6.1a.

To answer RQ2, I need issues that have reported both the SP and Timespent values. Therefore I use the subset of issues used in the second part of RQ2 from Chapter 6. This sample includes 697 issues from four projects (specifically, 128 issues from DM, 303 from TDQ, 104 from TMDM, and 162 from MDL). The correlation between Timespent and SP for this subset of issues is reported in Table 6.4.

7.3.3. Evaluation Measures

To evaluate the capability of the models under investigation, I compare their predictive performance against random guessing by reporting their Standard Accuracy (SA). The models are compared to the Mean and Median baselines and one another by their Mean and Median Absolute Errors (MAE and MdAE). See Section 4.3.4 for definitions and equations of SA, MAE and MdAE.

To distinguish between the methods that produce statistically significantly different results, I use the Wilcoxon Rank-Sum test (a.k.a. Mann–Whitney U test). Specifically, I test for the Null Hypothesis 4.3.1 (Chapter 4, page 69), and in case it was rejected, the Alternative Hypothesis 4.3.2 is accepted. I also use a standardised non-parametric effect size measure (i.e., the Vargha Delaney's $\hat{A}_{12}$ statistic) to assess the practical magnitude of the difference between the predictive performance of every two methods (Equation 4.4). The effect size is considered negligible for $0.6 < \hat{A}_{12}$ (indicated by an 'N' beside its value), small (S) for $0.6 \leq \hat{A}_{12} < 0.7$, medium (M) for $0.7 \leq \hat{A}_{12} < 0.8$, and large (L) for $\hat{A}_{12} \geq 0.8$ (the same boundaries used in the previous chapters).

7.4. RESULTS

7.4.1. RQ1. Estimating Development Time

The results of the four AI-based methods (including two Deep Learning methods) alongside two baseline estimators are shown in Table 7.1. The Standard Accuracy (SA) column shows the improvement of each of these methods against Random Guessing. The SA value for all methods is positive (outperforming Random Guessing) except for TF-IDF-SE on JSWCLOUD and TMDM. Nevertheless, TF-IDF-SE also achieves the lowest (best) MAE for five out of 15 projects (33%), the highest number among all the methods. This shows that TF-IDF-SE has the highest variance in results for different projects.

Considering the other methods, GPT2-SE and Deep-SE achieve the best MAE on three projects each (20%), LHC-SE achieves the best MAE in one project (7%), and the Median estimator in four projects (27%). The Mean estimator and Random Guessing never achieve the best MAE on any of the 15 projects under investigation. TF-IDF-SE and the Median estimator achieve the same best MAE on one project (i.e., BAM).

Considering that the unit for error is in hours, we can see that the best MAE achieved for 14 out of 15 projects is under 8.5 hours which is considered a typical working day for a software developer [167], [168].[3] This means that using these models to estimate the development time could be very useful in practice due to their low error.

However, to compare these models, we cannot rely solely on comparing their MAE values, as these results show that, in many cases, they achieve a very similar result. Therefore, I consider the statistical significance of the difference between the distributions of the absolute errors of each pair of methods as the determinative factor.

Table 7.2 shows the win-loss-tie statistics for each pair of estimation methods. In this table, each cell entry shows three values in order:

1. for how many cases (out of 15 projects) the method in the row produced a statistically significantly lower absolute error than the method in the column (i.e., the method in the row wins with a $p-value < 0.05$ for a one-way test)

2. for how many cases the method in the column produced a statistically significantly lower absolute error than the method in the row (i.e., the method in the row loses with a $p-value < 0.05$ for a one-way test in favour of the method in the column)

[3]The MAE for more than half the projects (8 out of 15) is less than or equal to four hours.

Table 7.1: RQ1: MAE, MdAE and SA values achieved by GPT2-SE, Deep-SE, LHC-SE, TF-IDF-SE, and the Mean and Median baselines. The best values per method and per project are printed in boldface. Error unit is in hours.

Project	Method	MAE	MdAE	SA	Project	Method	MAE	MdAE	SA	Project	Method	MAE	MdAE	SA
CWD	GPT2-SE	4.95	3.53	49.52	CLOV	GPT2-SE	4.60	5.01	62.42	TDQ	GPT2-SE	**11.45**	5.62	**26.96**
	Deep-SE	**4.86**	3.71	**50.39**		Deep-SE	3.12	2.52	74.47		Deep-SE	11.46	5.59	26.89
	LHC-SE	4.97	4.00	49.28		LHC-SE	6.54	7.37	46.53		LHC-SE	12.50	5.75	20.24
	TF-IDF-SE	5.69	**3.00**	41.99		TF-IDF-SE	**2.04**	**0.73**	**83.34**		TF-IDF-SE	14.89	7.00	5.01
	Median	4.88	3.42	50.20		Median	5.07	5.43	58.57		Median	11.88	**3.75**	24.24
	Mean	7.16	7.53	26.93		Mean	11.00	12.50	10.08		Mean	11.69	6.76	25.43
JSWCLOUD	GPT2-SE	**3.47**	2.22	**41.80**	FE	GPT2-SE	0.81	**0.57**	85.09	TDP	GPT2-SE	6.56	**3.49**	51.76
	Deep-SE	3.71	2.66	37.77		Deep-SE	1.18	1.15	78.42		Deep-SE	**6.21**	3.72	**54.35**
	LHC-SE	3.53	**2.00**	40.68		LHC-SE	1.63	1.75	70.23		LHC-SE	8.03	4.00	40.98
	TF-IDF-SE	6.49	7.00	-9.05		TF-IDF-SE	**0.73**	0.73	**86.59**		TF-IDF-SE	8.68	7.00	36.24
	Median	3.53	**2.00**	40.67		Median	2.47	2.70	54.79		Median	6.34	3.75	53.38
	Mean	4.62	4.28	22.35		Mean	5.24	5.55	3.94		Mean	8.41	9.45	38.22
JSWSERVER	GPT2-SE	6.41	**1.59**	27.08	TIDOC	GPT2-SE	3.20	1.40	50.14	TMDM	GPT2-SE	7.38	4.53	28.31
	Deep-SE	6.49	2.13	26.17		Deep-SE	3.43	1.46	46.60		Deep-SE	**7.24**	4.51	**29.64**
	LHC-SE	6.43	2.00	26.85		LHC-SE	**3.06**	1.63	**52.28**		LHC-SE	8.33	**4.00**	19.11
	TF-IDF-SE	6.67	2.00	24.17		TF-IDF-SE	3.61	**1.08**	43.84		TF-IDF-SE	10.60	7.00	-2.95
	Median	**6.39**	2.00	**27.39**		Median	3.19	1.75	50.33		Median	8.33	**4.00**	19.11
	Mean	7.15	4.36	18.70		Mean	5.11	4.63	20.38		Mean	7.26	4.86	29.42
JRASERVER	GPT2-SE	2.69	1.15	55.39	DM	GPT2-SE	7.75	6.34	52.89	TBD	GPT2-SE	8.16	3.42	34.66
	Deep-SE	2.67	1.58	55.64		Deep-SE	7.69	6.13	53.23		Deep-SE	8.06	4.09	35.45
	LHC-SE	2.73	1.50	54.75		LHC-SE	7.79	7.00	52.59		LHC-SE	8.14	5.00	34.84
	TF-IDF-SE	**2.64**	**1.00**	**56.22**		TF-IDF-SE	**7.62**	**6.00**	**53.68**		TF-IDF-SE	7.79	3.45	37.64
	Median	2.73	1.50	54.75		Median	7.79	7.00	52.59		Median	**7.69**	**3.00**	**38.44**
	Mean	4.62	4.96	23.35		Mean	10.79	9.84	34.39		Mean	9.43	8.10	24.50
BAM	GPT2-SE	4.04	2.26	45.44	NEXUS	GPT2-SE	2.31	1.76	36.01	TESB	GPT2-SE	**2.80**	**1.08**	**39.30**
	Deep-SE	4.12	**1.75**	44.35		Deep-SE	2.54	1.79	29.78		Deep-SE	2.83	1.33	38.58
	LHC-SE	4.01	2.00	45.86		LHC-SE	2.32	1.00	35.79		LHC-SE	2.80	1.50	39.21
	TF-IDF-SE	**4.00**	2.00	**45.99**		TF-IDF-SE	2.95	1.53	18.48		TF-IDF-SE	3.66	2.00	20.62
	Median	**4.00**	2.00	**45.99**		Median	**2.25**	**0.75**	**37.90**		Median	3.08	2.00	33.22
	Mean	5.22	4.63	29.44		Mean	2.56	1.94	29.18		Mean	3.55	3.00	22.94

3. for how many cases the difference between the distributions of the absolute errors produced by the two methods are not statistically significantly different (i.e., the two methods have a tie)

The “Summary” column sums up the number of times a method in the row wins, loses, or has a tie against all the other methods. This column shows that GPT2-SE holds first place with the highest number of wins (39) and the lowest number of losses (6). However, the difference is still very small for AI-based methods and the Median baseline. The Median baseline stands in second place with an equal number of wins with GPT2-SE (39) but only two more losses (8). Deep-SE is the third model, with 36 wins and eight losses.

These results show that all the AI-based methods investigated here are capable of estimating the development time within an absolute error of a few hours. Meanwhile, we should note that the Median baseline estimator is still among the best methods for agile effort estimation (in line with the previous results in Chapter 4 and 5).

Table 7.2: RQ1: Win-Loss-Tie summary of the Wilcoxon test results comparing AI-based effort estimation methods estimating development time against each other and the baseline methods. The best method is highlighted.

	Win | Loss | Tie								
Method	GPT2-SE	Deep-SE	LHC-SE	TFIDF-SE	Median	Mean	Random	Summary	Rank
GPT2-SE [21]		2 | 1 | 12	2 | 1 | 12	4 | 2 | 9	2 | 2 | 11	14 | 0 | 1	15 | 0 | 0	**39 | 6 | 45**	**1**
Deep-SE [26]	1 | 2 | 12		2 | 1 | 12	3 | 3 | 9	2 | 2 | 11	13 | 0 | 2	15 | 0 | 0	36 | 8 | 46	3
LHC-SE [29]	1 | 2 | 12	1 | 2 | 12		4 | 4 | 7	1 | 2 | 12	13 | 0 | 2	15 | 0 | 0	35 | 10 | 45	4
TFIDF-SE [23]	2 | 4 | 9	3 | 3 | 9	4 | 4 | 7		3 | 4 | 8	9 | 3 | 3	13 | 1 | 1	34 | 19 | 37	5
Median	2 | 2 | 11	2 | 2 | 11	2 | 1 | 12	4 | 3 | 8		14 | 0 | 1	15 | 0 | 0	39 | 8 | 43	2
Mean	0 | 14 | 1	0 | 13 | 2	0 | 13 | 2	3 | 9 | 3	0 | 14 | 1		13 | 0 | 2	16 | 63 | 11	6
Random	0 | 15 | 0	0 | 15 | 0	0 | 15 | 0	1 | 13 | 1	0 | 15 | 0	0 | 13 | 2		1 | 86 | 3	7

7.4.2. RQ2. Story Point as another Effort-Driver

To investigate this research question, I use GPT2-SE, as it is ranked the best-performing method in RQ1. The GPT2-SE’s implementation is adapted to accept the story point of the target issue as another input and concatenate it to the vector output of the GPT-2 transformer before it is fed to the MLP component (stage 2c in Figure 7.1). The estimation performance of the model then is compared to that of the original implementation, which uses only issue text as input. I call the new variant **GPT2-SE+**.

Results in Table 7.3 show that the MAE and MdAE produced by GPT2-SE+ are lower than that of GPT2-SE for all four open-source projects, which means that the addition of story points to the feature vector has a positive effect on the estimation accuracy of the GPT2-SE model. Nevertheless, the results of the Wilcoxon Rank-Sum test on the distribution of the absolute errors produced by each of these models in Table 7.4 shows that the improvement in estimation accuracy is not statistically significant for

Table 7.3: RQ2. Estimating development time (hours) by GPT2-SE, which solely uses issue text as input, and GPT2-SE+, which also exploits human-estimated story points as a cost-driver. The best values per metric are printed in bold for each project.

Project	Method	MAE	MdAE	SA	Project	Method	MAE	MdAE	SA
DM	GPT2-SE+	6.70	4.22	34.63	TDQ	GPT2-SE+	**16.92**	5.62	**25.97**
	GPT2-SE	6.91	5.98	32.64		GPT2-SE	17.22	8.48	24.67
	Median	**6.57**	5.00	**35.89**		Median	18.72	**5.00**	18.11
	Mean	6.99	4.63	31.78		Mean	17.92	9.96	21.62
MDL	GPT2-SE+	11.50	**4.23**	48.49	TMDM	GPT2-SE+	9.55	5.14	18.66
	GPT2-SE	11.69	6.16	47.66		GPT2-SE	9.70	5.27	17.31
	Median	**11.39**	4.98	**48.97**		Median	9.78	6.00	16.65
	Mean	18.09	15.58	18.97		Mean	**8.68**	**4.27**	**26.03**

Table 7.4: RQ3. Results of Wilcoxon Rank-Sum test (Vargha-Delaney effect size in parentheses) for GPT2-SE+ against GPT2-SE, and the Median and Mean baselines.

Project	Method	GPT2-SE+	GPT2-SE	Median	Mean
DM	GPT2-SE+ vs	-	0.453 (0.51) _	0.469 (0.51) _	0.154 (0.58) _
	GPT2-SE vs	0.554 (0.49) _	-	0.476 (0.51) _	0.434 (0.51) _
MDL	GPT2-SE+ vs	-	0.115 (0.59) _	0.379 (0.52) _	<0.001 (0.84) L
	GPT2-SE vs	0.888 (0.41) _	-	0.808 (0.44) _	<0.001 (0.81) L
TDQ	GPT2-SE+ vs	-	0.263 (0.53) _	0.468 (0.50) _	0.068 (0.58) _
	GPT2-SE vs	0.739 (0.47) _	-	0.705 (0.47) _	0.206 (0.54) _
TMDM	GPT2-SE+ vs	-	0.402 (0.52) _	0.380 (0.53) _	0.655 (0.47) _
	GPT2-SE vs	0.608 (0.48) _	-	0.463 (0.51) _	0.730 (0.45) _

any of the cases, and its improvement significance did not change against any of the baselines. Indeed, out of four projects, the Median baseline achieves the best MAE for two projects, the Mean baseline in one project and GPT2-SE+ for the one remaining project, non with a statistically significant difference.

In conclusion, these results suggest that although story points improve the estimation accuracy of the GPT2-SE model, there is no evidence to support a statistically significant difference, due to the Wilcoxon test results.

7.5. THREATS TO VALIDITY

This chapter uses Timespent value recorded for issues as the ground truth. The Timspent value is usually recorded by the developer who has developed and resolved the issue. Therefore, it is the closest value to the development time of the issue that could be collected from the dataset. However, this value might still not be accurate or honest. As stated in Section 5.6, and showed in this chapter, these values can be viewed as a placeholder used to test the model's ability to estimate the target variable (wherever their origin might be). Therefore, when available, the model can be trained on a more accurate target value.

I carefully selected unbiased accuracy performance measures to minimise threats to conclusion validity and applied statistical tests to rule out small differences.

The dataset used in this study represents a wide range of real-world projects. However, I cannot claim that this dataset is representative of all software projects. Especially the dataset used to investigate RQ2 is a relatively small dataset, as few issues in the TAWOS dataset have reported both story points and Timespent values. Furthermore, all projects investigated are collected from open-source repositories, which can differ from industrial projects in many aspects. A key difference that may affect the reliability of the Timespent values (i.e., our target variable), and therefore the estimation of development time, is the behaviour of contributors, developers, and project stakeholders. It is also expected that issue reports may be written in a more disciplined environment in a commercial project setting, thus providing more useful information and containing less noise. Therefore, further investigation of commercial projects from industrial software companies is needed to validate the conclusions made in this study.

7.6. CONCLUSION

In this chapter, I leveraged the state-of-the-art methods in story point estimation to train with and estimate the development time. Specifically, I adapted four AI-based estimation methods (including two deep-learning models) to estimate Timespent value for issues, the total time developers spent on developing issues.

The results showed that all these methods could estimate the development time with an average absolute error of a few hours. Although the AI-based models showed similar performance, the Median baseline estimator has achieved the second-best performance among all the methods, outperformed by only GPT2-SE, the recently introduced transformer-based deep-learning method.

Furthermore, supplementing the text input to GPT2-SE with the human-estimated story point of the issue introduced a marginal improvement to the estimation performance of the model. However, the improvement was not statistically significant.

Further analysis with more inclusive data from both open-source and industrial projects is needed to strengthen or rectify the conclusions achieved in this study. Evidently, leveraging the advancements in natural language processing, intelligent agile effort estimation models are becoming more effective in building accurate and useful estimation models. However, they still are not able to outperform a naïve baseline model like the Median estimator statistically significantly in all cases. Therefore, further investigation with different approaches, including utilising more effective cost drivers, might lead future work to a better solution.

Chapter 8

Conclusions and Future Work

This thesis aims to shed light on an important issue which can affect the success and usability of intelligent effort estimation models proposed for agile software development.

Estimating the effort required to develop software has always been a challenging task. In particular, agile software development methodologies face additional challenges in this matter due to several factors, including high tolerance to changes in requirements during the development that these methodologies promise, very fine-grained tasks which make it difficult to measure the size of the software unit to be produced, unstructured and noisy description of the requirements, and lack of documentation for some of the effort drivers. Nevertheless, several studies proposed different approaches for estimating story points as an indicator of the effort needed to develop software tasks.

Via large empirical studies, this thesis finds that the story points are not easily predictable by intelligent estimation models proposed so far, nor they are much effective in measuring the development effort of software tasks in agile. The human-estimated story points carry the bias of their estimators, which makes them improper as a target variable for the intelligent estimation models. In the following, I first outline the main findings of this thesis and then discuss the possible future directions of this research.

8.1. FINDINGS

Chapter 4 replicates prominent intelligent story point estimation approaches, including the state-of-the-art deep-learning models. The findings show that none of the available approaches is able to outperform a naïve baseline technique like the Median estimator. The novel clustering-based approach introduced in Chapter 5, which works on the premise that clustering similar data points together helps reduce variance and increases the accuracy of any model built upon them, achieved a similar result. All these approaches, which in essence try to find semantic similarities between user stories to estimate effort, very often fail to provide better estimations than the baselines. Therefore, their additional complexity does not seem justified. This suggests that semantic similarity between user stories might not be sufficient, or even effective, for agile effort estimation.

Chapter 6 studied the relationship between human-expert estimated story points and the time required by the developers to realise a task. This relationship is found to be low or medium for most of the projects (93% with proxies and 75% with actual development time). This study also found that the majority of the investigated projects (78%) lack consistent human-expert estimations for story points. These results provide empirical evidence that human-expert estimated story points might be biased and, thus, are not a good indicator for the issue development effort of agile open-source projects. Therefore, any intelligent effort estimation model, which learns from human-expert estimated story points, should consider the impact of such bias in their output. A solution can be avoiding the utilisation of story points altogether and instead training the models on the actual development time and estimating the development time.

Chapter 7 explored the usability of the current approaches proposed for story point estimation in the estimation of the actual development time. The findings indicate that all these methods could estimate the actual development time with an average absolute error of a few hours. However, the Median baseline estimator still achieved a comparable result (second-best place among all the methods investigated). Even supplementing the text input to GPT2-SE (the transformer-based estimation method) with the human-estimated story point of the issue introduced only a marginal non-significant improvement to the estimation performance of the model.

Moreover, the experiments conducted in Chapter 4 of this thesis further proved the importance of replication studies and making replication packages publicly available in order to support reproduction, replication and extension of previous work. A few small and honest bugs in the implementation of the previous methods inflated their estimation performance above their actual capability, fostering the research in a misguided direction. Acknowledging that even advanced deep-learning models, relying on the textual information of issue description, fail to outperform a baseline method that does not use any information may infer that such information does not provide adequate leverage for effort estimation. Therefore, using a more powerful technique to identify similar issues from the past solely based on their description would not be as helpful as looking into other sources of information to identify and use more effective effort drivers.

Another outcome of this thesis is the collection and curation of a large and rich dataset of issues from open-source projects (Chapter 3) that can be instrumental in enabling the research community to pursue further research in this line and to repeat, replicate, and extend the previous work that used this dataset. All the scripts and codes used in this thesis are also available online to expedite future work.

8.2. FUTURE WORK

Each chapter of this thesis laid directions for future work. Some of the most important ones are as follows.

Future work might devise and experiment with additional effort drivers extracted from issue reports in order to achieve more accurate effort estimates. These effort drivers can be project-, people-, or organisation-related features. For example, previous work showed that the developers' reputation, experience, and workload are among the effective effort drivers [25], [169].

In this work (Chapters 6 and 7), I considered the development time recorded by the developers (i.e., *Timsepent*) as a proxy for effort. I made sure that the issues used for our empirical evaluation had only one developer working on them. As such, *Timsepent* represents their effort (in person-hours). In projects where more than one developer may work on the same issue simultaneously, this time may present merely the elapsed time and might not accurately capture the overall effort spent by the developers working on a given issue. Future work should take this into consideration when quantifying and using development time in their research.

More advanced data analysis and cleaning prior to the model building may become effective when using textual input. Issue descriptions can be very noisy and include unnecessary information that can mislead the training model, especially when the current models use only a limited number of tokens from the beginning of the issue description. One promising avenue may be to identify and extract the most important points of the description to be used in the estimation process in combination with other effort drivers.

Moreover, all the research conducted in this thesis used open-source projects that may not be representative of standard agile practices. Therefore, further analysis with more inclusive data from both open-source and industrial projects is needed to strengthen or rectify the conclusions achieved in this thesis.

Furthermore, the explainability of the model's decision can be crucial in the acceptance of intelligent agile effort estimation models in practice. Future work may invest more in this important aspect of the proposed models [21], [63].

I hope that the results reported in this thesis will encourage further work aiming at improving methods for agile software effort estimation.

Bibliography

[1] A. Trendowicz and R. Jeffery, “Software project effort estimation,” *Foundations and Best Practice Guidelines for Success, Constructive Cost Model–COCOMO pags*, pp. 277–293, 2014.

[2] T. Rajala and H. Aaltonen, “Reasons for the failure of information technology projects in the public sector,” *The Palgrave handbook of the public servant*, pp. 1075–1093, 2021.

[3] K. Dejaeger, W. Verbeke, D. Martens, and B. Baesens, “Data mining techniques for software effort estimation: A comparative study,” *IEEE Transactions on Software Engineering*, vol. 38, no. 2, pp. 375–397, 2011.

[4] J. Wen, S. Li, Z. Lin, Y. Hu, and C. Huang, “Systematic literature review of machine learning based software development effort estimation models,” *Information and Software Technology*, vol. 54, no. 1, pp. 41–59, 2012.

[5] V. Tawosi, F. Sarro, A. Petrozziello, and M. Harman, “Multi-objective software effort estimation: A replication study,” *IEEE Transactions on Software Engineering (TSE)*, vol. 48, no. 8, pp. 3185–3205, 2021.

[6] R. F. Korte, “Biases in decision making and implications for human resource development,” *Advances in Developing Human Resources*, vol. 5, no. 4, pp. 440–457, 2003.

[7] B. W. Boehm, “Software engineering economics,” *IEEE Transactions on Software Engineering*, no. 1, pp. 4–21, 1984.

[8] I. Sommerville, *Software Engineering GE.* Pearson Australia Pty Limited, 2016.

[9] R. S. Pressman, *Software engineering: a practitioner’s approach*. Palgrave MacMillan, 2005.

[10] A. J. Albrecht and J. E. Gaffney, “Software function, source lines of code, and development effort prediction: A software science validation,” *IEEE Transactions on Software Engineering*, no. 6, pp. 639–648, 1983.

[11] C. Symons, “The COSMIC method for measuring the work-output component of productivity,” in *Rethinking Productivity in Software Engineering*, Springer, 2019, pp. 191–204.

[12] S. Di Martino, F. Ferrucci, C. Gravino, and F. Sarro, “Assessing the effectiveness of approximate functional sizing approaches for effort estimation,” *Information and Software Technology*, vol. 123, p. 106 308, 2020.

[13] F. Ferrucci, C. Gravino, P. Salza, and F. Sarro, “Investigating functional and code size measures for mobile applications,” in *Proceedings of Euromicro Conference on Software Engineering and Advanced Applications*, 2015, pp. 365–368.

[14] S. Abrahão, L. D. Marco, F. Ferrucci, J. Gómez, C. Gravino, and F. Sarro, “Definition and evaluation of a COSMIC measurement procedure for sizing web applications in a model-driven development environment,” *Information and Software Technology*, vol. 104, pp. 144–161, 2018.

[15] S. Di Martino, F. Ferrucci, C. Gravino, and F. Sarro, “Web effort estimation: Function point analysis vs. COSMIC,” *Information and Software Technology*, vol. 72, pp. 90–109, 2016.

[16] C. Santana, F. Leoneo, A. Vasconcelos, and C. Gusmão, “Using function points in agile projects,” in *International Conference on Agile Software Development*, Springer, 2011, pp. 176–191.

[17] K. Beck and M. Fowler, *Planning extreme programming*. Addison-Wesley Professional, 2001.

[18] M. Usman, E. Mendes, F. Weidt, and R. Britto, “Effort estimation in agile software development: A systematic literature review,” in *PROMISE*, 2014, pp. 82–91.

[19] M. Fernández-Diego, E. R. Méndez, F. González-Ladrón-De-Guevara, S. Abrahão, and E. Insfran, “An update on effort estimation in agile software development: A systematic literature review,” *IEEE Access*, vol. 8, pp. 166 768–166 800, 2020.

[20] B. Alsaadi and K. Saeedi, “Data-driven effort estimation techniques of agile user stories: A systematic literature review,” *Artificial Intelligence Review*, pp. 1–32, 2022.

[21] M. Fu and C. Tantithamthavorn, "GPT2SP: A transformer-based agile story point estimation approach," *IEEE Transactions on Software Engineering*, 2022.

[22] P. Abrahamsson, I. Fronza, R. Moser, J. Vlasenko, and W. Pedrycz, "Predicting development effort from user stories," in *2011 International Symposium on Empirical Software Engineering and Measurement*, IEEE, 2011, pp. 400–403.

[23] S. Porru, A. Murgia, S. Demeyer, M. Marchesi, and R. Tonelli, "Estimating story points from issue reports," in *PROMISE*, 2016, pp. 1–10.

[24] R. G. Soares, "Effort estimation via text classification and autoencoders," in *2018 International Joint Conference on Neural Networks (IJCNN)*, IEEE, 2018, pp. 01–08.

[25] E. Scott and D. Pfahl, "Using developers' features to estimate story points," in *Proceedings of the 2018 International Conference on Software and System Process*, 2018, pp. 106–110.

[26] M. Choetkiertikul, H. K. Dam, T. Tran, T. Pham, A. Ghose, and T. Menzies, "A deep learning model for estimating story points," *IEEE Transactions on Software Engineering*, vol. 45, no. 7, pp. 637–656, 2019. doi: 10.1109/TSE.2018.2792473.

[27] V. Tawosi, A. Al-Subaihin, R. Moussa, and F. Sarro, "A versatile dataset of agile open source software projects," in *Proceedings of 19th International Conference on Mining Software Repositories (MSR)*, ACM, 2022.

[28] V. Tawosi, R. Moussa, and F. Sarro, "Agile effort estimation: Have we solved the problem yet? insights from a replication study," *IEEE Transactions on Software Engineering (TSE)*, pp. 1–19, 2022.

[29] V. Tawosi, A. Al-Subaihin, and F. Sarro, "Investigating the effectiveness of clustering for story point estimation," in *Proceedings of the 29th IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER)*, IEEE, 2022, pp. 816–827.

[30] V. Tawosi, R. Moussa, and F. Sarro, "On the relationship between story point and development effort in agile open-source software," in *16th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*, ACM/IEEE, vol. 16, 2022.

[31] L. Angelis, I. Stamelos, and M. Morisio, “Building a software cost estimation model based on categorical data,” in *Proceedings Seventh International Software Metrics Symposium*, IEEE, 2001, pp. 4–15.

[32] M. Shepperd and C. Schofield, “Estimating software project effort using analogies,” *IEEE Transactions on Software Engineering*, vol. 23, no. 11, pp. 736–743, 1997.

[33] E. Kocaguneli, T. Menzies, J. Keung, D. Cok, and R. Madachy, “Active learning and effort estimation: Finding the essential content of software effort estimation data,” *IEEE Transactions on Software Engineering*, vol. 39, no. 8, pp. 1040–1053, 2012.

[34] G. Wittig and G. Finnie, “Estimating software development effort with connectionist models,” *Information and Software Technology*, vol. 39, no. 7, pp. 469–476, 1997.

[35] A. L. Oliveira, “Estimation of software project effort with support vector regression,” *Neurocomputing*, vol. 69, no. 13-15, pp. 1749–1753, 2006.

[36] S. Chulani, B. Boehm, and B. Steece, “Bayesian analysis of empirical software engineering cost models,” *IEEE Transactions on Software Engineering*, vol. 25, no. 4, pp. 573–583, 1999.

[37] F. Ferrucci, M. Harman, and F. Sarro, “Search-based software project management,” in *Software Project Management in a Changing World*, Springer, 2014, pp. 373–399.

[38] F. Sarro, A. Petrozziello, and M. Harman, “Multi-objective software effort estimation,” in *2016 IEEE/ACM 38th International Conference on Software Engineering (ICSE)*, IEEE, 2016, pp. 619–630.

[39] E. Kocaguneli, A. Tosun, and A. Bener, “AI-based models for software effort estimation,” in *2010 36th EUROMICRO Conference on Software Engineering and Advanced Applications*, IEEE, 2010, pp. 323–326.

[40] E. Kocaguneli, T. Menzies, and J. W. Keung, “On the value of ensemble effort estimation,” *IEEE Transactions on Software Engineering*, vol. 38, no. 6, pp. 1403–1416, 2011.

[41] A. Corazza, S. Di Martino, F. Ferrucci, C. Gravino, F. Sarro, and E. Mendes, “Using tabu search to configure support vector regression for effort estimation,” *Empirical Software Engineering*, vol. 18, pp. 506–546, 2013.

[42] A. Corazza, S. Di 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*, 2010, pp. 1–10.

[43] M. Harman, S. A. Mansouri, and Y. Zhang, “Search-based software engineering: Trends, techniques and applications,” *ACM Computing Surveys (CSUR)*, vol. 45, no. 1, pp. 1–61, 2012.

[44] F. Sarro, “Search-based approaches for software development effort estimation,” in *Proceedings of the 12th International Conference on Product Focused Software Development and Process Improvement*, 2011, pp. 38–43.

[45] C. J. Burgess and M. Lefley, “Can genetic programming improve software effort estimation? a comparative evaluation,” *Information and software technology*, vol. 43, no. 14, pp. 863–873, 2001.

[46] F. Ferrucci, C. Gravino, R. Oliveto, and F. Sarro, “Genetic programming for effort estimation: An analysis of the impact of different fitness functions,” in *2nd International Symposium on Search Based Software Engineering*, IEEE, 2010, pp. 89–98.

[47] F. Ferrucci, C. Gravino, and F. Sarro, “How multi-objective genetic programming is effective for software development effort estimation?” In *International Symposium on Search Based Software Engineering*, Springer, 2011, pp. 274–275.

[48] M. Harman, P. McMinn, J. T. De Souza, and S. Yoo, “Search based software engineering: Techniques, taxonomy, tutorial,” in *Empirical software engineering and verification*, Springer, 2010, pp. 1–59.

[49] F. Sarro and A. Petrozziello, “Linear programming as a baseline for software effort estimation,” *ACM transactions on software engineering and methodology (TOSEM)*, vol. 27, no. 3, pp. 1–28, 2018.

[50] A. J. Nebro, J. J. Durillo, and M. Vergne, “Redesigning the jmetal multi-objective optimization framework,” in *Proceedings of the companion publication of the 2015 annual conference on genetic and evolutionary computation*, 2015, pp. 1093–1100.

[51] M. Fowler, J. Highsmith, *et al.*, “The agile manifesto,” *Software Development*, vol. 9, no. 8, pp. 28–35, 2001.

[52] K. S. Rubin, *Essential Scrum: A practical guide to the most popular Agile process*. Addison-Wesley, 2012.

[53] M. Usman, J. Börstler, and K. Petersen, "An effort estimation taxonomy for agile software development," *International Journal of Software Engineering and Knowledge Engineering*, vol. 27, no. 04, pp. 641–674, 2017.

[54] O. Oni and E. Letier, "Analyzing uncertainty in release planning: A method and experiment for fixed-date release cycles," *ACM Transactions on Software Engineering and Methodology (TOSEM)*, vol. 31, no. 2, pp. 1–39, 2021.

[55] K. Schwaber and J. Sutherland, "The scrum guide. 2020," *Scrum Alliance*, 2020.

[56] A. J. Albrecht, "Measuring application development productivity," in *Proc. Joint Share, Guide, and IBM Application Development Symposium, 1979*, 1979.

[57] C. Symons, A. Abran, C. Ebert, and F. Vogelezang, "Measurement of software size: Advances made by the cosmic community," in *2016 Joint Conference of the International Workshop on Software Measurement and the International Conference on Software Process and Product Measurement (IWSM-MENSURA)*, IEEE, 2016, pp. 75–86.

[58] S. Di Martino, F. Ferrucci, C. Gravino, and F. Sarro, "Web effort estimation: Function point analysis vs. COSMIC," *Information and Software Technology*, vol. 72, pp. 90–109, 2016.

[59] M. Cohn, *Agile Estimating and Planning*. Pearson Education, 2005.

[60] H. Kniberg and M. Skarin, *Kanban and Scrum-making the most of both*. Lulu. com, 2010.

[61] J. Bowes, *Agile concepts: Estimating and planning poker*, May 2017. [Online]. Available: https://manifesto.co.uk/agile-concepts-estimating-planning-poker/.

[62] B. Marapelli, A. Carie, and S. M. Islam, "RNN-CNN MODEL: A bi-directional long short-term memory deep learning network for story point estimation," in *2020 5th International Conference on Innovative Technologies in Intelligent Systems and Industrial Applications (CITISIA)*, IEEE, 2020, pp. 1–7.

[63] M. Abadeer and M. Sabetzadeh, “Machine learning-based estimation of story points in agile development: Industrial experience and lessons learned,” in *2021 IEEE 29th International Requirements Engineering Conference Workshops (REW)*, IEEE, 2021, pp. 106–115.

[64] V. Tawosi, R. Moussa, and F. Sarro, *Agile effort estimation: Have we solved the problem yet? insights from a second replication study (GPT2SP replication report)*, 2022. doi: 10.48550/ARXIV.2209.00437. [Online]. Available: https://arxiv.org/abs/2209.00437.

[65] N. Bettenburg, M. Nagappan, and A. E. Hassan, “Think locally, act globally: Improving defect and effort prediction models,” in *2012 9th IEEE Working Conference on Mining Software Repositories (MSR)*, IEEE, 2012, pp. 60–69.

[66] G. Nagpal, M. Uddin, and A. Kaur, “Analyzing software effort estimation using k-means clustered regression approach,” *ACM SIGSOFT Software Engineering Notes*, vol. 38, no. 1, pp. 1–9, 2013.

[67] V. K. Bardsiri, D. N. A. Jawawi, S. Z. M. Hashim, and E. Khatibi, “Increasing the accuracy of software development effort estimation using projects clustering,” *IET software*, vol. 6, no. 6, pp. 461–473, 2012.

[68] L. L. Minku and S. Hou, “Clustering Dycom: An online cross-company software effort estimation study,” in *Proceedings of the 13th International Conference on Predictive Models and Data Analytics in Software Engineering*, 2017, pp. 12–21.

[69] T. Menzies, Y. Yang, G. Mathew, B. Boehm, and J. Hihn, “Negative results for software effort estimation,” *Empirical Software Engineering*, vol. 22, no. 5, pp. 2658–2683, 2017.

[70] J. J. C. Gallego, D. Rodrıéguez, M. Á. Sicilia, M. G. Rubio, and A. G. Crespo, “Software project effort estimation based on multiple parametric models generated through data clustering,” *Journal of Computer Science and Technology*, vol. 22, no. 3, pp. 371–378, 2007.

[71] T. Menzies, A. Butcher, D. Cok, *et al.*, “Local versus global lessons for defect prediction and effort estimation,” *IEEE Transactions on software engineering*, vol. 39, no. 6, pp. 822–834, 2012.

[72] S.-J. Huang, N.-H. Chiu, and Y.-J. Liu, “A comparative evaluation on the accuracies of software effort estimates from clustered data,” *Information and Software Technology*, vol. 50, no. 9-10, pp. 879–888, 2008.

[73] 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.

[74] R. Popli and N. Chauhan, “Cost and effort estimation in agile software development,” in *2014 international conference on reliability optimization and information technology (ICROIT)*, IEEE, 2014, pp. 57–61.

[75] E. Ungan, N. Çizmeli, and O. Demirörs, “Comparison of functional size based estimation and story points, based on effort estimation effectiveness in scrum projects,” in *2014 40th EUROMICRO Conference on Software Engineering and Advanced Applications*, IEEE, 2014, pp. 77–80.

[76] A. T. Raslan, N. R. Darwish, and H. A. Hefny, “Towards a fuzzy based framework for effort estimation in agile software development,” *International Journal of Computer Science and Information Security*, vol. 13, no. 1, p. 37, 2015.

[77] S. M. Satapathy and S. K. Rath, “Empirical assessment of machine learning models for agile software development effort estimation using story points,” *Innovations in Systems and Software Engineering*, vol. 13, no. 2, pp. 191–200, 2017.

[78] H. Huijgens and R. v. Solingen, “A replicated study on correlating agile team velocity measured in function and story points,” in *Proceedings of the 5th International Workshop on Emerging Trends in Software Metrics*, 2014, pp. 30–36.

[79] M. Salmanoglu, T. Hacaloglu, and O. Demirors, “Effort estimation for agile software development: Comparative case studies using COSMIC functional size measurement and story points,” in *Proceedings of the 27th International Workshop on Software Measurement and 12th International Conference on Software Process and Product Measurement*, 2017, pp. 41–49.

[80] T. Dybå and T. Dingsøyr, “Empirical studies of agile software development: A systematic review,” *Information and Software Technology*, vol. 50, no. 9, pp. 833–859, 2008, issn: 0950-5849. doi: https://doi.org/10.1016/j.infsof.2008.01.006.

[81] M. A. Langley, "Success Rates Rise - 2017 9th Global Project Management Survey," Tech. Rep., 2017. [Online]. Available: https://www.pmi.org/-/media/pmi/documents/public/pdf/learning/thought-leadership/pulse/pulse-of-the-profession-2017.pdf.

[82] *Jira Issue & Project Tracking Software | Atlassian*. [Online]. Available: https://www.atlassian.com/software/jira (visited on 01/21/2022).

[83] Q. Umer, H. Liu, and I. Illahi, "CNN-based automatic prioritization of bug reports," *IEEE Transactions on Reliability*, vol. 69, no. 4, pp. 1341–1354, 2019.

[84] C. Gavidia-Calderon, F. Sarro, M. Harman, and E. T. Barr, "The assessor's dilemma: Improving bug repair via empirical game theory," *IEEE Transactions on Software Engineering*, vol. 47, no. 10, pp. 2143–2161, 2021. doi: 10.1109/TSE.2019.2944608.

[85] Y. Huang, J. Wang, S. Wang, Z. Liu, D. Wang, and Q. Wang, "Characterizing and predicting good first issues," in *Proceedings of the 15th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*, 2021, pp. 1–12.

[86] S. Mani, A. Sankaran, and R. Aralikatte, "Deeptriage: Exploring the effectiveness of deep learning for bug triaging," in *Proceedings of the ACM India Joint International Conference on Data Science and Management of Data*, 2019, pp. 171–179.

[87] O. Chaparro, J. Lu, F. Zampetti, *et al.*, "Detecting missing information in bug descriptions," in *Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering*, 2017, pp. 396–407.

[88] M. Choetkiertikul, H. K. Dam, T. Tran, A. Ghose, and J. Grundy, "Predicting delivery capability in iterative software development," *IEEE Transactions on Software Engineering*, vol. 44, no. 6, pp. 551–573, 2017.

[89] M. Ortu, G. Destefanis, B. Adams, A. Murgia, M. Marchesi, and R. Tonelli, "The jira repository dataset: Understanding social aspects of software development," ser. PROMISE '15, Beijing, China: Association for Computing Machinery, 2015, isbn: 9781450337151. doi: 10.1145/2810146.2810147.

[90] M. Ortu, A. Murgia, G. Destefanis, *et al.*, “The emotional side of software developers in jira,” in *2016 IEEE/ACM 13th Working Conference on Mining Software Repositories (MSR)*, IEEE, 2016, pp. 480–483.

[91] A. Valdez, H. Oktaba, H. Gómez, and A. Vizcaıéno, “Sentiment analysis in jira software repositories,” in *2020 8th International Conference in Software Engineering Research and Innovation (CONISOFT)*, IEEE, 2020, pp. 254–259.

[92] T. Zhang, B. Xu, F. Thung, S. A. Haryono, D. Lo, and L. Jiang, “Sentiment analysis for software engineering: How far can pre-trained transformer models go?” In *2020 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, IEEE, 2020, pp. 70–80.

[93] V. Tawosi, A. Alsubaihin, M. Rebecca, and F. Sarro, *The TAWOS dataset*. [Online]. Available: https://github.com/SOLAR-group/TAWOS.git (visited on 01/27/2022).

[94] *The Jira REST Java Client, Version 5.2.0*. [Online]. Available: https://mvnrepository.com/artifact/com.atlassian.jira/jira-rest-java-client-app/5.2.0 (visited on 01/24/2022).

[95] R. Sepahvand, R. Akbari, and S. Hashemi, “Predicting the bug fixing time using word embedding and deep long short term memories,” *IET Software*, vol. 14, no. 3, pp. 203–212, 2020.

[96] Y. Lee, S. Lee, C.-G. Lee, I. Yeom, and H. Woo, “Continual prediction of bug-fix time using deep learning-based activity stream embedding,” *IEEE Access*, vol. 8, pp. 10 503–10 515, 2020.

[97] M. Habayeb, S. S. Murtaza, A. Miranskyy, and A. B. Bener, “On the use of hidden markov model to predict the time to fix bugs,” *IEEE Transactions on Software Engineering*, vol. 44, no. 12, pp. 1224–1244, 2017.

[98] M. Jimenez, R. Rwemalika, M. Papadakis, F. Sarro, Y. Le Traon, and M. Harman, “The importance of accounting for real-world labelling when predicting software vulnerabilities,” in *Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 2019, pp. 695–705.

[99] A. A. Bangash, H. Sahar, A. Hindle, and K. Ali, “On the time-based conclusion stability of cross-project defect prediction models,” *Empirical Software Engineering*, vol. 25, no. 6, pp. 5047–5083, 2020.

[100] F. Sarro, R. Moussa, A. Petrozziello, and M. Harman, “Learning from mistakes: Machine learning enhanced human expert effort estimates,” *IEEE Transactions on Software Engineering*, 2020.

[101] *Source Code and data for "A Deep Learning Model for Estimating Story Points" · GitHub*. [Online]. Available: `https://github.com/morakotch/datasets/tree/master/storypoint/IEEE%20TSE2018` (visited on 12/03/2020).

[102] M. Izadi, K. Akbari, and A. Heydarnoori, “Predicting the objective and priority of issue reports in software repositories,” *Empirical Software Engineering*, vol. 27, no. 2, p. 50, 2022.

[103] R. H. Al-Ta’ani and R. Razali, “A framework for requirements prioritisation process in an agile software development environment: Empirical study,” *International Journal on Advanced Science, Engineering and Information Technology*, vol. 6, no. 6, pp. 846–856, 2016.

[104] C. Fitzgerald, E. Letier, and A. Finkelstein, “Early failure prediction in feature request management systems: An extended study,” *Requirements Engineering*, vol. 17, pp. 117–132, 2012.

[105] L. Li, M. Harman, E. Letier, and Y. Zhang, “Robust next release problem: Handling uncertainty during optimization,” in *Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation*, 2014, pp. 1247–1254.

[106] J. J. Durillo, Y. Zhang, E. Alba, M. Harman, and A. J. Nebro, “A study of the bi-objective next release problem,” *Empirical Software Engineering*, vol. 16, no. 1, pp. 29–60, 2011.

[107] W. H. A. Al-Zubaidi, P. Thongtanunam, H. K. Dam, C. Tantithamthavorn, and A. Ghose, “Workload-aware reviewer recommendation using a multi-objective search-based approach,” in *Proceedings of the 16th ACM International Conference on Predictive Models and Data Analytics in Software Engineering*, 2020, pp. 21–30.

[108] K. Beck and M. Fowler, *Planning Extreme Programming*, 1st. Boston, MA, USA: Addison-Wesley Longman Publishing, 2000, isbn: 0201710919.

[109] M. Jørgensen, “A review of studies on expert estimation of software development effort,” *JSS*, vol. 70, no. 1-2, pp. 37–60, 2004.

[110] M. D. Uncles and S. Kwok, “Designing research with in-built differentiated replication,” *Journal of Business Research*, vol. 66, no. 9, pp. 1398–1405, 2013.

[111] N. Juristo and O. S. Gómez, “Replication of software engineering experiments,” in *Empirical software engineering and verification*, Springer, 2010, pp. 60–88.

[112] A. Santos, S. Vegas, M. Oivo, and N. Juristo, “Comparing the results of replications in software engineering,” *Empirical Software Engineering*, vol. 26, pp. 1–41, 2021.

[113] S. Suthaharan, “Support vector machine,” in *Machine Learning Models and Algorithms for Big Data Classification: Thinking with Examples for Effective Learning*. Boston, MA: Springer US, 2016, pp. 207–235, isbn: 978-1-4899-7641-3. doi: `10.1007/978-1-4899-7641-3_9`.

[114] D. F. Specht, “Probabilistic neural networks,” *Neural Networks*, vol. 3, no. 1, pp. 109–118, 1990.

[115] I. Rish *et al.*, “An empirical study of the naive bayes classifier,” in *IJCAI 2001 workshop on empirical methods in artificial intelligence*, vol. 3, 2001, pp. 41–46.

[116] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[117] F. A. Gers, J. Schmidhuber, and F. Cummins, “Learning to forget: Continual prediction with LSTM,” *Neural Computation*, 1999.

[118] T. Pham, T. Tran, D. Phung, and S. Venkatesh, “Faster training of very deep networks via p-norm gates,” in *2016 23rd International Conference on Pattern Recognition (ICPR)*, IEEE, 2016, pp. 3542–3547.

[119] M. Shepperd and S. MacDonell, “Evaluating prediction systems in software project estimation,” *Information and Software Technology*, vol. 54, no. 8, pp. 820–827, 2012.

[120] W. B. Langdon, J. Dolado, F. Sarro, and M. Harman, “Exact mean absolute error of baseline predictor, MARP0,” *Information and Software Technology*, vol. 73, pp. 16–18, 2016.

[121] B. A. Kitchenham, L. M. Pickard, S. G. MacDonell, and M. J. Shepperd, “What accuracy statistics really measure,” *IEEE Software*, vol. 148, no. 3, pp. 81–85, 2001.

[122] 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.

[123] *Deep-SE, fixed Python source code, TF/IDF-SE Python source code, and the datasets used in Chapter 4*. [Online]. Available: `https://figshare.com/s/709c7e18c52e4264b70e` (visited on 11/25/2021).

[124] E. Mendes, M. Kalinowski, D. Martins, F. Ferrucci, and F. Sarro, “Cross-vs. within-company cost estimation studies revisited: An extended systematic review,” in *Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering*, 2014, pp. 1–10.

[125] L. Minku, F. Sarro, E. Mendes, and F. Ferrucci, “How to make best use of cross-company data for web effort estimation?” In *2015 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*, IEEE, 2015, pp. 1–10.

[126] F. Ferrucci, E. Mendes, and F. Sarro, “Web effort estimation: The value of cross-company data set compared to single-company data set,” in *PROMISE*, ACM, 2012, pp. 29–38.

[127] M. Jimenez, R. Rwemalika, M. Papadakis, F. Sarro, Y. Le Traon, and M. Harman, “The importance of accounting for real-world labelling when predicting software vulnerabilities,” in *Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 2019, pp. 695–705.

[128] D. Falessi, J. Huang, L. Narayana, J. F. Thai, and B. Turhan, “On the need of preserving order of data when validating within-project defect classifiers,” *Empirical Software Engineering*, vol. 25, pp. 4805–4830, 2020.

[129] L. Taylor and G. Nitschke, “Improving deep learning with generic data augmentation,” pp. 1542–1547, 2018.

[130] A. Mikołajczyk and M. Grochowski, “Data augmentation for improving deep learning in image classification problem,” in *2018 international interdisciplinary PhD workshop (IIPhDW)*, IEEE, 2018, pp. 117–122.

[131] N. Mittas, I. Mamalikidis, and L. Angelis, “A framework for comparing multiple cost estimation methods using an automated visualization toolkit,” *Information and Software Technology*, vol. 57, pp. 310–328, 2015.

[132] P. A. Whigham, C. A. Owen, and S. G. Macdonell, “A baseline model for software effort estimation,” *ACM Transactions on Software Engineering and Methodology (TOSEM)*, vol. 24, no. 3, pp. 1–11, 2015.

[133] A. Arcuri and L. Briand, “A hitchhiker’s guide to statistical tests for assessing randomized algorithms in software engineering,” *Software Testing, Verification and Reliability*, vol. 24, no. 3, pp. 219–250, 2014.

[134] G. Neumann, M. Harman, and S. Poulding, “Transformed vargha-delaney effect size,” in *International Symposium on Search Based Software Engineering*, Springer, 2015, pp. 318–324.

[135] *R: The R Project for Statistical Computing, V. 4.0.1*, 2020. [Online]. Available: https://www.r-project.org/ (visited on 02/02/2021).

[136] F. Shull, V. Basili, J. Carver, *et al.*, “Replicating software engineering experiments: Addressing the tacit knowledge problem,” in *Proceedings international symposium on empirical software engineering*, IEEE, 2002, pp. 7–16.

[137] J. C. Carver, “Towards reporting guidelines for experimental replications: A proposal,” in *1st international workshop on replication in empirical software engineering*, vol. 1, 2010, pp. 1–4.

[138] V. Tawosi, A. Al-Subaihin, and F. Sarro, *Investigating the Effectiveness of Clustering for Story Point Estimation – Replication Package*. [Online]. Available: https://github.com/SOLAR-group/LHC-SE.git (visited on 06/03/2022).

[139] D. M. Blei, A. Y. Ng, and M. I. Jordan, “Latent dirichlet allocation,” *Journal of Machine Learning Research*, vol. 3, no. Jan, pp. 993–1022, 2003.

[140] A. Al-Subaihin, F. Sarro, S. Black, and L. Capra, “Empirical comparison of text-based mobile apps similarity measurement techniques,” *Empirical Software Engineering*, vol. 24, no. 6, pp. 3290–3315, 2019.

[141] A. A. Al-Subaihin, F. Sarro, S. Black, *et al.*, “Clustering mobile apps based on mined textual features,” in *Proceedings of the 10th ACM/IEEE international symposium on empirical software engineering and measurement*, 2016, pp. 1–10.

[142] F. Ebrahimi, M. Tushev, and A. Mahmoud, “Classifying mobile applications using word embeddings,” *ACM Transactions on Software Engineering and Methodology (TOSEM)*, vol. 31, no. 2, pp. 1–30, 2021.

[143] T. L. Griffiths and M. Steyvers, “Finding scientific topics,” *Proceedings of the National Academy of Sciences*, vol. 101, no. suppl_1, pp. 5228–5235, 2004.

[144] F. Murtagh and P. Legendre, “Ward’s hierarchical agglomerative clustering method: Which algorithms implement ward’s criterion?” *Journal of Classification*, vol. 31, no. 3, pp. 274–295, 2014.

[145] A. Starczewski and A. Krzyżak, “Performance evaluation of the silhouette index,” in *International conference on artificial intelligence and soft computing*, Springer, 2015, pp. 49–58.

[146] J. Cohen, *Statistical power analysis for the behavioral sciences*. Routledge, 2013.

[147] F. Sarro, F. Ferrucci, M. Harman, A. Manna, and J. Ren, “Adaptive multi-objective evolutionary algorithms for overtime planning in software projects,” *IEEE Transactions on Software Engineering*, vol. 43, no. 10, pp. 898–917, 2017.

[148] F. Sarro, M. Harman, Y. Jia, and Y. Zhang, “Customer rating reactions can be predicted purely using app features,” in *2018 IEEE 26th International Requirements Engineering Conference (RE)*, IEEE, 2018, pp. 76–87.

[149] M. Gammage, “Why your IT project may be riskier than you think,” *Harvard Business Review*, vol. 89, no. 11, pp. 22–22, 2011.

[150] 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*, 2015, pp. 1–10.

[151] A. E. D. Hamouda, “Using agile story points as an estimation technique in CMMI organizations,” in *2014 Agile Conference*, IEEE, 2014, pp. 16–23.

[152] F. Ferrucci, C. Gravino, P. Salza, and F. Sarro, “Investigating functional and code size measures for mobile applications: A replicated study,” in *Product-Focused Software Process Improvement*, P. Abrahamsson, L. Corral, M. Oivo, and B. Russo, Eds., Cham: Springer International Publishing, 2015, pp. 271–287, isbn: 978-3-319-26844-6.

[153] 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*, 2015, pp. 1–4.

[154] P. E. McKnight and J. Najab, "Mann-Whitney U Test," *The Corsini Encyclopedia of Psychology*, pp. 1–1, 2010.

[155] K. Pearson, *Notes on regression and inheritance in the case of two parents proceedings of the royal society of london, 58, 240-242*, 1895.

[156] C. Spearman, "The proof and measurement of association between two things.," pp. 72–101, 1961.

[157] M. G. Kendall, "A new measure of rank correlation," *Biometrika*, vol. 30, no. 1/2, pp. 81–93, 1938.

[158] S. Boslaugh, *Statistics in a nutshell: A desktop quick reference*. " O'Reilly Media, Inc.", 2012.

[159] C. Croux and C. Dehon, "Influence functions of the spearman and kendall correlation measures," *Statistical methods & applications*, vol. 19, no. 4, pp. 497–515, 2010.

[160] A. R. Gilpin, "Table for conversion of kendall's tau to spearman's rho within the context of measures of magnitude of effect for meta-analysis," *Educational and psychological measurement*, vol. 53, no. 1, pp. 87–92, 1993.

[161] J. F. Hemphill, "Interpreting the magnitudes of correlation coefficients.," *American Psychologist*, vol. 58, pp. 78–79, 2003.

[162] V. Tawosi, R. Moussa, and F. Sarro, *Online Appendix containing Data and R Scripts for "On the Relationship Between Story Point and Development Effort in Agile Open-Source Software"*, 2022. [Online]. Available: https://github.com/SOLAR-group/SPvsDevelopmentEffort.git (visited on 04/29/2022).

[163] J. M. Zhang, F. Li, D. Hao, *et al.*, "A study of bug resolution characteristics in popular programming languages," *IEEE Transactions on Software Engineering*, vol. 47, no. 12, pp. 2684–2697, 2019.

[164] J. P. Royston, "An extension of Shapiro and Wilk's W test for normality to large samples," *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, vol. 31, no. 2, pp. 115–124, 1982.

[165] A. Radford, K. Narasimhan, T. Salimans, I. Sutskever, *et al.*, “Improving language understanding by generative pre-training,” 2018.

[166] A. Vaswani, N. Shazeer, N. Parmar, *et al.*, “Attention is all you need,” *Advances in neural information processing systems*, vol. 30, 2017.

[167] A. N. Meyer, E. T. Barr, C. Bird, and T. Zimmermann, “Today was a good day: The daily life of software developers,” *IEEE Transactions on Software Engineering*, vol. 47, no. 5, pp. 863–880, 2019.

[168] A. N. Meyer, L. E. Barton, G. C. Murphy, T. Zimmermann, and T. Fritz, “The work life of developers: Activities, switches and perceived productivity,” *IEEE Transactions on Software Engineering*, vol. 43, no. 12, pp. 1178–1193, 2017.

[169] S. Alamir, P. Z. R. Silva, A. Pozanco, *et al.*, “A planning approach to agile project management. The Jira Planner,” *FinPlan 2021*, p. 1, 2021.