Utrecht University

MSc Business Informatics

Master Thesis

Leveraging Open-Source Data for Software Cost Estimation: A Predictive Modeling Approach

Gal Mrvar

First Supervisor: Dr. Siamak Farshidi

Second Supervisor: Dr. Slinger Jansen

2023

Utrecht University
Faculty of Science

Abstract

Software cost estimation is a long-standing research area in software engineering. The diverse array of cost-affecting factors, coupled with the dynamic nature of software development, necessitates constant caution in this domain. Over the years, several strategies and models were formulated to tackle this issue, each presenting different degrees of success and usability. This study introduces a software cost estimation tool that significantly improves the current scenario by automating the data extraction process from an online software project repository. This tool, devised through extensive research and expert insights, collects, aggregates, and stores project data in a dataset, creating a comprehensive knowledge base.

The automatic extraction and aggregation of project data overcome the manual and time-consuming data collection processes prevalent in the current scenario, thereby enhancing efficiency and precision. It provides an easily accessible and ready-to-use repository for researchers, enabling them to experiment and identify critical factors affecting software costs without being burdened by the data collection process.

Furthermore, our data repository allows for software effort estimation using various machine-learning techniques. Within the scope of this study, we implemented and evaluated four specific methods, offering researchers a launchpad for comparative analysis and refinement of existing models.

The implementation and application of the developed tool showcase its potential to improve the field. By offering a novel perspective and methodology for software cost estimation, it contributes significantly to this research area. Furthermore, it lays the groundwork for future researchers to further explore this domain with ease and precision, indicating a promising direction for the evolution of software cost estimation methodologies.

Acknowledgements

I would like to thank everyone who supported me throughout this research. Special thanks to Siamak for offering strong guidance and supervision, especially with weekly meetings where my progress was challenged and met with critical thinking, without which this work would not be possible.
These past ten months were extremely challenging, combining multiple research methods that required a lot of dedication, patience, and technical skills. In this work, I discovered previously unaware capabilities within myself. The work described in this thesis is a testament to the growth and development I have experienced over these months and during my studies at Utrecht University.
Finally, I would like to extend my sincere gratitude to my second supervisor Slinger and all of the participants in this study. Their experience and knowledge have been precious to this study.

To everyone who played a role, no matter how small, in the successful completion of this thesis - thank you.

Contents

Acknowledgements **ii**

1 Introduction **1**
1.1 Problem Statement . . . 2
1.2 Related Work . . . 3
1.3 Research Questions . . . 3
1.4 Study Contributions . . . 5
1.5 Conceptual Model . . . 5

2 Research Approach **7**
2.1 Research Methods . . . 7
2.2 Literature Study . . . 7
2.3 Design Science . . . 9
2.4 Case Study . . . 10
2.5 Expert Interviews . . . 11

3 Literature Review **12**
3.1 Data Sources and Search Strategy . . . 12
3.2 Search Process . . . 13
3.3 Inclusion and Exclusion Criteria . . . 14
3.4 Quality Assessment . . . 14
3.5 Data Extraction . . . 15
3.5.1 Influential Variables and Key Performance Indicators . . . 16
3.5.2 Models and Methods . . . 18
3.5.3 Evaluation Methods . . . 18
3.5.4 Datasets . . . 20
3.6 Data Analysis and Synthesis . . . 21

4 Expert interviews **23**
4.1 Research Method . . . 23
4.2 Sampling Strategy and Selection of Experts . . . 24
4.2.1 Participants' Demographics and Work Roles . . . 24
4.3 Interview Protocol . . . 25
4.4 Conducting the Interviews . . . 25
4.4.1 Data Analysis . . . 26
4.5 Results and Findings . . . 26
4.5.1 Insights Supporting Key Points . . . 26
4.5.2 Results from the Interviews . . . 27
4.6 Discussion . . . 28
4.7 Potential Researcher Bias and its Impact . . . 29

5 Data Preparation and Extraction 30
5.1 Factor Classification . . . 30
5.1.1 Manual Factors . . . 31
5.1.2 Automatic Factors . . . 31
5.2 Data Extraction . . . 33
5.2.1 Utilizing the GitHub GraphQL API . . . 33
5.2.2 Mapping Data with Automatic Factors . . . 34
5.3 Dataset Creation and Preprocessing . . . 36
5.3.1 Final Dataset Description . . . 37

6 Software Cost Estimation Modeling 38
6.1 Preliminary Considerations . . . 38
6.1.1 Consideration of Manual Factors . . . 38
6.1.2 Integration with the COCOMO II Model . . . 39
6.2 Machine Learning Approaches . . . 40
6.2.1 K-Nearest Neighbors (KNN) . . . 41
6.2.2 Hierarchical Clustering . . . 41
6.2.3 Autoencoder . . . 41
6.2.4 SVM . . . 42
6.3 Construction of the Estimation Model . . . 42

7 Empirical Evidence 44
7.1 Case study . . . 44
7.1.1 Project Requirements . . . 45
7.1.2 Results . . . 47
7.1.3 Analysis . . . 47
7.2 Experiment . . . 48
7.2.1 Data Collection . . . 48
7.2.2 Results and Evaluation . . . 49
7.2.3 Analysis . . . 51

8 Discussion 52
8.1 Final Estimation Model . . . 52
8.2 Threats to Validity . . . 53
8.2.1 Construct Validity . . . 53
8.2.2 Internal Validity . . . 54
8.2.3 External Validity . . . 54
8.2.4 Conclusion Validity . . . 54
8.3 Ethical Considerations . . . 55

9 Conclusion 56
9.1 Cost Modeling and Estimation (MRQ) . . . 56
9.2 Literature on Cost Estimation (RQ1) . . . 56
9.3 Parameters in Estimation Approaches (RQ2) . . . 57
9.4 Data Acquisition Techniques (RQ3) . . . 57
9.5 Parameters for Cost Estimation Model (RQ4) . . . 57
9.6 Extracting Project Requirements (RQ5) . . . 58
9.7 Estimating Cost Based on Project Similarity (RQ6) . . . 58
9.8 Evaluating the Cost Estimation Model (RQ7) . . . 58
9.9 Future Research Directions . . . 59

A Interview Protocol 60

References 61

Chapter 1

Introduction

Software effort estimation or software cost estimation has been proven to be especially important regarding software development (Jones, 2008). For this reason, the research area in this field began in the 1960s, and since that time, a significant number of approaches concerning software project cost estimation have been published (Boehm, Abts, & Chulani, 2000). These techniques and models mainly support problems such as budgeting, trade-off, risk analysis, project planning, and software improvement investment analysis (Boehm et al., 2000).

There were multiple attempts and approaches created in the past to solve these problems and accurately predict the effort of software development. They vary from expert judgment to machine learning approaches and algorithms. However, existing tools and techniques have proven inefficient in estimating the cost, resources, and schedule (Tadayon, 2005). This is why expert judgment remains the most popular method in software cost estimation (Shepperd & Cartwright, 2001). The benefits of using expert review instead of algorithmic approaches for this task are mostly related to estimates being customized to the specific organizational culture. This directly contributes to the estimation's accuracy since the experts' final estimation is subjective and based on feelings and logic (Tadayon, 2005). At the same time, researchers claim that expert opinion's personal and unstructured nature makes the estimation method particularly vulnerable. One of the main dangers is unconscious prejudice made by researchers and writers (Hughes, 1996).

With the arrival of Agile software development, researchers addressed the problem of cost estimation by creating new techniques. One such approach is Planning Poker, introduced by (Grenning, 2002), which uses group consensus to estimate. In general, the method combines expert opinion, analogy, and disaggregation (Cohn, 2005). However, results showed that Planning Poker could reduce estimation performance when the team lacks experience with estimation or similar tasks.

In the past decades, several researchers addressed this issue and focused on designing a model to predict the effort of software development accurately. This resulted in a large number of prediction models (Aslam, Ijaz, Lali, & Mehmood, 2017). However, few achieved satisfactory results under all circumstances since their performance changes with different data sets, making them very unstable in practice (Shepperd & Kadoda, 2001).

For this reason, this research aims to create a machine learning-based approach to tackling the problem of software effort estimation. The main goal is to analyze existing developers' communities, mainly GitHub[1], and then try to extract relevant data from this platform and later use it for software effort estimation using state-of-the-art machine learning algorithms. The motivation for using the machine learning approach is that it allows complete automation

[1] https://github.com/

of the data collection process, which usually can not be done manually and frequently.

Firstly, an overview of existing methods and approaches in software effort estimation will be provided using a semi-systematic literature review. This will be followed by comparing gathered data and extracting relevant features, KPIs, algorithms, and evaluation methods. Moreover, several interviews with experts in this domain will be conducted to understand better the research domain and our findings in the literature study. Based on the findings mentioned in the above measures, an estimation model for software cost estimation was developed using the following steps:

1. Data collection: The tool will automatically collect data from software project repositories based on metadata discovered in the literature study and expert interviews.

2. Knowledge base: The collected data will be stored in a dataset defined as the tool's knowledge base.

3. Cost estimation model: This model will be used to estimate the effort for software development based on the data stored in the knowledge base and future requirements for the software in development.

This thesis elaborates on the complete process of the creation of the tool and the rationale of decisions that were made, as well as the findings that were recognized during this study.

1.1 Problem Statement

Software effort estimation (SEE) is a significant and challenging field in software development. Over and underestimation of software effort and cost led to the failure of many past projects (de A. Cabral, Oliveira, & da Silva, 2023). The authors (Ramasubbu & Balan, 2012) conducted field studies in three companies using industry-standard, metrics-driven, regression-based cost estimation techniques. They discovered that early estimates for newly starting projects were inaccurate and contained up to 300% variance in mean estimation errors. This resulted in extremely inaccurate estimation, which led to significant losses as the companies were using fixed-price contracting, and therefore projects could not be re-priced after project initiation.

Underestimation of projects can lead to clients giving up on sponsoring the project. On the other hand, overestimation can lead to increased resources that would otherwise be unnecessary. This usually happens because of unrealistic goals, weak resource estimations, and a poor understanding of the project's complexity (de A. Cabral et al., 2023).

In the past, a wide variety of studies have been conducted concerning software development effort estimation. At the same time, those studies produced limited success, which led to some researchers criticizing SEE, claiming that it advocates more than it evaluates. Additionally, the role of SEE is to rely on past data to produce reliable results. However, that data is often incomplete, uncertain, and noisy (Sehra, Brar, Kaur, & Sehra, 2017).

For this reason, **one of the main issues at hand lies in the way that current software effort estimation (SEE) techniques operate, which is largely dependent on specific, often outdated, datasets. This dependence creates significant inaccuracies due to the lack of a comprehensive, end-to-end approach that encompasses the whole process from data collection to data processing, and finally, to estimation modeling. As a result, the datasets in use quickly become obsolete in the rapidly evolving field of software development, making it challenging to obtain timely and accurate project inputs. This limitation of SEE techniques has been a contributing factor to its lack of popularity and the high error rate seen in its predictions.**

Therefore, our objective is to enhance the field of software effort estimation research by designing an automated system. This comprehensive tool aims to streamline the entire workflow, encompassing data extraction, manipulation, and ultimately, estimation modeling. By integrating these steps, we aspire to increase the efficiency and accuracy of the estimation process, thereby addressing the prevalent issues and shortcomings of current SEE techniques.

1.2 Related Work

A variety of methods have been employed historically to address the challenges inherent in the software effort estimation (SEE) research domain. Predominantly, these methods can be broadly divided into algorithmic and non-algorithmic categories.

Algorithmic models, for instance, include COCOMO (Boehm, 1984), SLIM (Putnam, 1978), and function points (A. Albrecht & Gaffney, 1983). These models necessitate the provision of initial inputs that are often difficult to acquire at the inception of a project. These inputs typically encompass metrics such as the lines of code and the complexity of the project.

On the contrary, non-algorithmic approaches leverage soft computing techniques like neural networks, regression, and fuzzy logic, demonstrating considerable potential in resolving more intricate issues. These complexities encompass non-linear problems, optimization challenges, and situations requiring intelligent control and decision-making (Suresh Kumar & Behera, 2020). These methods' adaptability and learning capability make them a potent tool in SEE, primarily when dealing with complex and uncertain software projects.

Numerous scholars have endeavored to enhance the field of software effort estimation (SEE) by implementing soft computing techniques. However, as the results of our literature review illustrate, most of these approaches have been applied to existing datasets and did not aim to develop a comprehensive tool encompassing data collection and estimation concurrently.

As far as data collection is concerned, various projects and institutions have accumulated data for this particular purpose. For instance, the International Software Benchmarking Standards Group (ISBSG) (*International Software Benchmarking Standards Group*, 2022) annually collates data from a variety of private institutions, thereby creating a database filled with software metrics. This database enables researchers to conduct an array of different investigations. Nevertheless, the ISBSG data is not freely accessible.

In contrast, the PROMISE repository (Sayyad Shirabad & Menzies, 2005) offers a collection of publicly available datasets intended to aid researchers in constructing predictive software models, which also encompasses SEE. However, the PROMISE repository does not reflect contemporary software practices.

Concurrently, innovative strategies are being implemented to collect data for software effort estimation. A notable example is the study conducted by (E. I. Mustafa & Osman, 2020), where data from 120 software development projects spanning 42 organizations in Sudan were aggregated to create a dataset.

An approach comparable to the one proposed in this study was carried out by (Qi et al., 2017), who created and published a SEE dataset by extracting data from open-source projects on the GitHub platform. However, this approach was limited to projects developed in a single programming language: Java. Additionally, the comprehensive tool utilized for data extraction and subsequent software cost estimation was not publicly accessible during this study.

1.3 Research Questions

Relying on the problem statement, the following main research question was formulated:

- **MRQ**: *How can a robust and systematic approach be developed to model the cost estimation of software projects, considering the essential factors of stakeholder requirements and priorities?*

In support of this research question, the following sub-research questions were developed:

- **RQ1**: *What are the state-of-the-art methodologies and techniques for cost estimation in the literature?*

- **RQ2**: *What are the key parameters utilized by the cost estimation approaches documented in the literature?*

  To address the above questions, a semi-systematic literature review was conducted, the details of which will be expounded upon in Chapter 3. The main objective is to gain a more profound understanding of existing literature and identify significant elements for consideration during this study.

- **RQ3**: *What are the prevalent data acquisition techniques employed to automatically extract software project requirements from developer communities such as GitHub?*

  This question will assist us in understanding and comparing the techniques that will allow for the extraction of requirements and data from the GitHub platform. This data will subsequently be utilized in the development of the estimation model.

- **RQ4**: *What essential parameters should be considered when constructing a cost estimation model for software projects?*

  A semi-systematic literature review was conducted to support this question. Identifying the appropriate factors is a demanding yet crucial aspect of SEE. Discerning the right factors will give us insight into key considerations for creating the estimation model.

- **RQ5**: *How can automated techniques be employed to extract software project requirements from developer communities, facilitating the construction of a comprehensive knowledge base?*

  To answer this question, the techniques identified in RQ3 will be applied to gather data and establish a knowledge base for our estimation model.

- **RQ6**: *How can the cost estimation of a software project be conducted by leveraging its similarities and patterns identified through the analysis of existing projects in the knowledge base?*

  The aim is to provide insights on drawing comparisons between new projects and those pre-existing in the created knowledge base. This includes the modeling and creation of the final estimation model.

- **RQ7**: *What are the recommended evaluation methods to assess the performance and effectiveness of the proposed cost estimation model?*

  A semi-systematic literature review was utilized to gather information on the most popular evaluation approaches in machine learning relevant to SEE. This information was instrumental when we evaluated and compared our results with other methodologies.

1.4 Study Contributions

This research enhances our comprehension of software effort estimation (SEE) and its methodologies. A semi-systematic literature review was adopted to offer comprehensive insights into contemporary practices in SEE, along with their associated challenges.

Furthermore, the primary aim was assembling and creating an extensive dataset procured from the GitHub development platform. This assembled dataset is aimed to facilitate future researchers in this domain to experiment with various methodological approaches, leveraging the gathered data. This also includes extracting new potential beneficial factors from the accumulated data. Moreover, future scholars might utilize the developed tool to extract new valuable repositories, resulting in an up-to-date dataset. Finally, the tool eliminates the extensive effort to construct such a dataset.

In conclusion, this study seeks to construct a more reliable estimation model, which is designed to aid stakeholders such as project managers, developers, and product owners in planning and organizing their software development projects with improved efficiency.

1.5 Conceptual Model

The ultimate research goal, along with the development of the estimation model, is depicted using a conceptual model as shown in Figure 1.1. The creation of this conceptual model was influenced by the pipeline proposed by (Farshidi & Zhao, 2022).

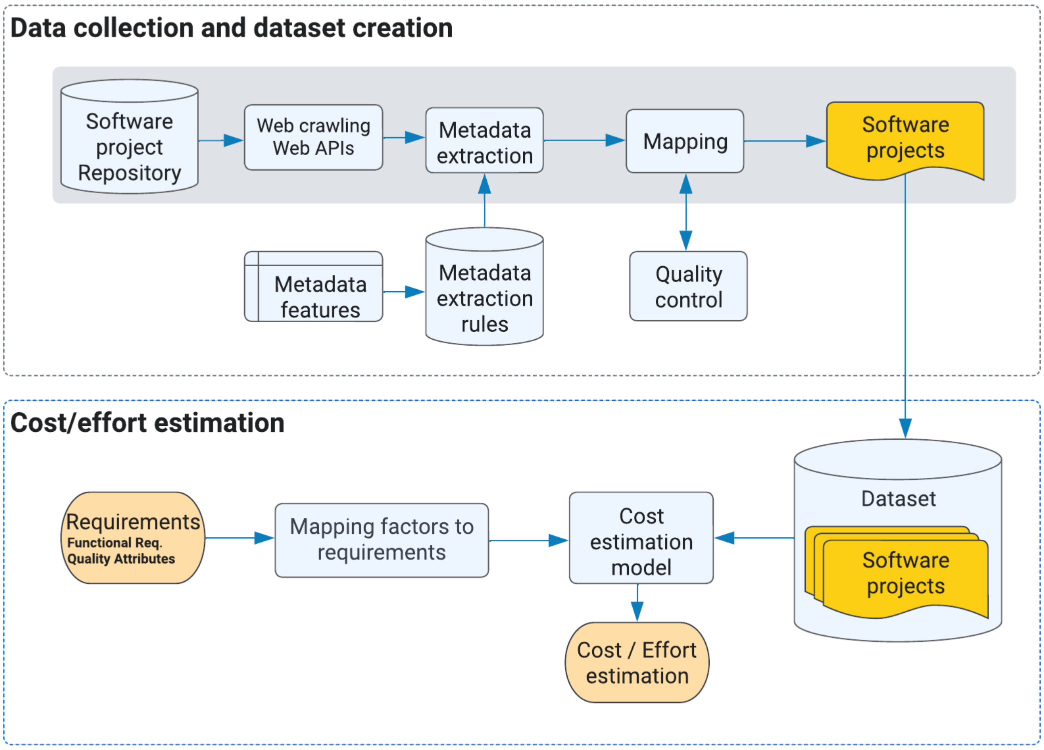


Figure 1.1: Conceptual Model of the Software Effort Estimation Process.

The **Software Project Repository** denotes the GitHub repository leveraged for data mining in this study. GitHub houses data from many software projects, with open-source projects publicly available for data extraction. Despite the existence of numerous repositories on the internet, GitHub was selected due to its prevalent usage and versatile API, which enables the collection of a vast spectrum of data.

Web Crawling and Web APIs refer to the processes adopted for data extraction from the repository. Web crawling involves a spider bot browsing repositories systematically via a list of predefined links. The crawler identifies all hyperlinks in the retrieved documents, subsequently adding them to the list for future visits. In contrast, Web APIs are interfaces designed by the repository website (in this case, GitHub), providing an endpoint for direct data collection without needing a crawler. The collected data is organized in a structured format, such as fundamental values or JSON, regardless of the method.

Metadata Extraction constitutes the process of extracting metadata from the gathered documents, tasked with managing influential features and properties, or textual content within a document relevant to this study.

Metadata Features denote essential features collated through the semi-systematic literature review detailed in Chapter 3. These features were further delineated and evaluated by domain experts via numerous interviews.

Metadata Extraction Rules are human-made guidelines, predicated on metadata features, aimed at enhancing the precision of metadata extraction. They refine potential extracted values that can be ascribed to the metadata features.

The **Language Model** analyzes documents to transmute qualitative data into quantitative data. It employs statistical and probabilistic methods to verify the likelihood of a specific sequence of words appearing in a particular document.

Mapping involves adding external information (e.g., domain keywords) to the extracted metadata features based on predefined rules. These rules were formulated using SLR, domain experts' insights, and language model predictions.

Quality Control signifies a stage that ensures the efficacy of the mapping process. It evaluates mapping based on the number of values correctly assigned to metadata features and the number of potential topics. If the quality level is insufficient, the mapping function's threshold and the rules should be revised to enhance mapping quality.

Software Projects refers to the repository data extracted in a valuable format, which the estimation model can later employ. It encompasses all factors and KPIs identified as crucial in the preceding steps.

The **Data Set of Software Projects** is a repository where the extracted repositories will be stored. The cost estimation model will utilize this data set to perform estimation based on previous projects.

Requirements allude to the list of requirements that the development system ought to possess. These requirements could include the system's functional specifications, quality attributes, and other critical factors that have been proven essential for the future software project.

Mapping Factors to Requirements is an essential process to align the identified factors with the project requirements necessary for model training. The alignment of the gathered data with the requirements is critical for model training.

The **Cost Estimation Model** represents the final iteration of the developed model. This model utilizes the factors of identified requirements for a prospective project as input to predict the final cost or effort of the software. To accomplish this, the model employs the data of past projects stored in the dataset. The model's output is the **Cost/Effort Estimation**, which is typically expressed in terms of the time needed for software development.

Chapter 2

Research Approach

This chapter delineates the methodology adopted for this study, outlines the initial research plan, and discusses the approaches to address the research questions. The chapter begins by providing an overview of the various research methods used, each of which will be clarified subsequently, with justifications and rationales for their inclusion in the study.

2.1 Research Methods

The research questions presented in Section 1.3 were approached using a diversified set of research methods. This multipronged approach ensured a holistic view of the research problems, enhancing the findings' robustness and reliability.

A total of four research methods were employed in this study: literature review, expert interviews, design science, and case study. The respective roles of these methods in answering each research question are presented in Table 2.1. An "X" in the intersection of a method and a research question indicates the use of that method to address the question.

2.2 Literature Study

A semi-systematic literature review (SLR) was conducted to collate pertinent research information pertaining to the study. This process helped identify trends in the field, focus the direction of the study to better serve the scientific community, and gather essential features and Key Point Indicators (KPIs) used in Software Effort Estimation (SEE) for training the model in this study. The review also facilitated the selection of appropriate machine learning models, algorithms, and evaluation methods.

The protocol proposed by (B. Kitchenham, 2004) was adhered to for conducting the SLR, comprising eleven phases: problem formulation, research questions, review protocol (search strategy), search process, searching, screening, inclusion/exclusion criteria, quality assessment, data extraction, data analysis and synthesis, and reporting. Each of these phases will be detailed subsequently.

The results, findings, and detailed procedure of the conducted SLR are documented in Chapter 3.

Problem formulation and research questions are crucial for performing an effective SLR, delivering optimal results and findings. The problem in the SEE scope and research questions used during this research were elaborated in Chapter 1.

The **review protocol (search strategy)** delineates the approach to be followed during the SLR. This study adopted a five-step SLR, leveraging the phases proposed by (B. Kitchenham, 2004). The first step encompassed the search process, searching, and screening; the second

Research questions		Research methods			
		Literature study	Design Science	Case Study	Expert interview
MRQ	How can a robust and systematic approach be developed to model the cost estimation of software projects, considering the essential factors of stakeholder requirements and priorities?	X	X	X	X
RQ1	What are the state-of-the-art methodologies and techniques for cost estimation in the literature?	X	X		
RQ2	What are the key parameters utilized by the cost estimation approaches documented in the literature?	X	X		
RQ3	What are the prevalent data acquisition techniques employed to automatically extract software project requirements from developer communities such as GitHub?	X	X		
RQ4	What essential parameters should be considered when constructing a cost estimation model for software projects?	X	X		X
RQ5	How can automated techniques be employed to extract software project requirements from developer communities, facilitating the construction of a comprehensive knowledge base?		X		
RQ6	How can the cost estimation of a software project be conducted by leveraging its similarities and patterns identified through the analysis of existing projects in the knowledge base?		X		
RQ7	What are the recommended evaluation methods to assess the performance and effectiveness of the proposed cost estimation model?		X	X	

Table 2.1: Research Methods Employed.

step defined and implemented inclusion/exclusion criteria; the third step conducted a quality assessment of selected papers; the fourth step extracted data from these papers, and the fifth step analyzed and synthesized the collected data. Each step was performed in a separate spreadsheet to maintain a clear record of progress.

The **search process** involved manually searching for pertinent papers in the selected domain based on the initial hypothesis and understanding of the field. For this study, primary focus was accorded to four online libraries: IEEEXplore[1], Springer[2], ACM DL[3], and ScienceDirect[4]. The study primarily focused on papers published post-2010, allowing for a contemporary understanding of the field. Each collected paper was documented in a spreadsheet containing relevant details (title, URL, authors, abstract, keywords, year of publication, citations, venue of publication, and venue ranking.). This information then facilitated the generation of relevant search terms, used in the subsequent searching phase.

The **searching** phase was initiated upon generating a search term from the previously collected papers. This term was then used in the online libraries mentioned earlier, enabling the collection of the most relevant papers in the research domain via their search engines. The papers were exported into a CSV file and included in the same spreadsheet.

The **screening** phase entailed a cursory review of the collected papers, including screening abstracts, keywords, and other potentially relevant information. Based on this information, papers were ranked to indicate their relevance to the research. The ranking employed ordinal values (None, Low, Medium, High).

Inclusion/Exclusion criteria were then defined to determine which papers should be considered in the subsequent steps. The criteria were defined based on the relevance determined in the screening phase and also on venue ranking, citations, year of publication, and research type.

[1] https://ieeexplore.ieee.org/

[2] https://www.springer.com/

[3] https://dl.acm.org/

[4] https://www.sciencedirect.com/

A **quality assessment** was then conducted to evaluate the quality of each selected paper following the inclusion/exclusion process. This included extracting pertinent information from the papers, such as research method and type, data collection method, and the evaluation method employed by the authors. Additionally, it was noted whether the paper had a clear problem statement, research questions, research challenges, a clear statement of findings, and real-world use cases. Based on these parameters, another criterion was defined to assess the quality of the papers. Papers that met these criteria were considered in the SLR and, consequently, this research study.

Data extraction was then conducted on the papers that met the quality assessment criteria. Relevant data were extracted for the research, such as the features used by authors in their software effort estimation, the models they used, and the evaluation methods they employed. This helped garner insights into what is currently considered state-of-the-art in this research field.

Analyzing and synthesizing data was conducted to comprehend the collected data, disregard unpopular models or features, and group similar features with different names but essentially the same meaning.

Finally, **reporting** provides the ultimate outcome of the conducted SLR, consisting of a comprehensive report of the findings. The report, a detailed discussion of the results, and a complete explanation of the executed SLR are elaborated in Chapter 3.

2.3 Design Science

This study adopted a design science methodology, as proposed by (Hevner & Chatterjee, 2010), specifically developed to support research projects in the field of information systems.

The principal aim of design science research is to introduce innovative artifacts and outline the processes for constructing such artifacts in order to enhance the prevailing environment. In the context of information systems, artifacts can encompass constructs, models, methods, instantiations, algorithms, and more. Design science research encompasses three integral cycles.

The first is referred to as the **Relevance Cycle**, which underscores the applicability of the research within the context of its intended use. This cycle not only specifies requirements for the research but also defines the acceptance criteria for evaluating the results. This cycle provokes critical questions: Does the design artifact foster improvements in the environment, and how can these enhancements be measured?

The second cycle, dubbed the **Rigor Cycle**, asserts that design science research builds upon a substantial knowledge base comprising scientific theories and engineering methods. This enables rigorous investigation within design science research. This knowledge base should encapsulate the state-of-the-art expertise and experiences within the research domain, in addition to existing artifacts and processes within the application domain.

The third, the **Design Cycle**, constitutes the nucleus of the research project. In this cycle, activities oscillate between creating and evaluating the artifact, utilizing direct feedback to refine the design further. Although this is where the core research is conducted, the researcher must balance the effort expended on artifact construction and evaluation.

Recognizing all three cycles completes the design of an artifact that not only has a sturdy place within the research environment but also withstands evaluations in the theoretical and practical realms of the research area. Figure 2.1 depicts the Design Science model and the life cycles it embodies.

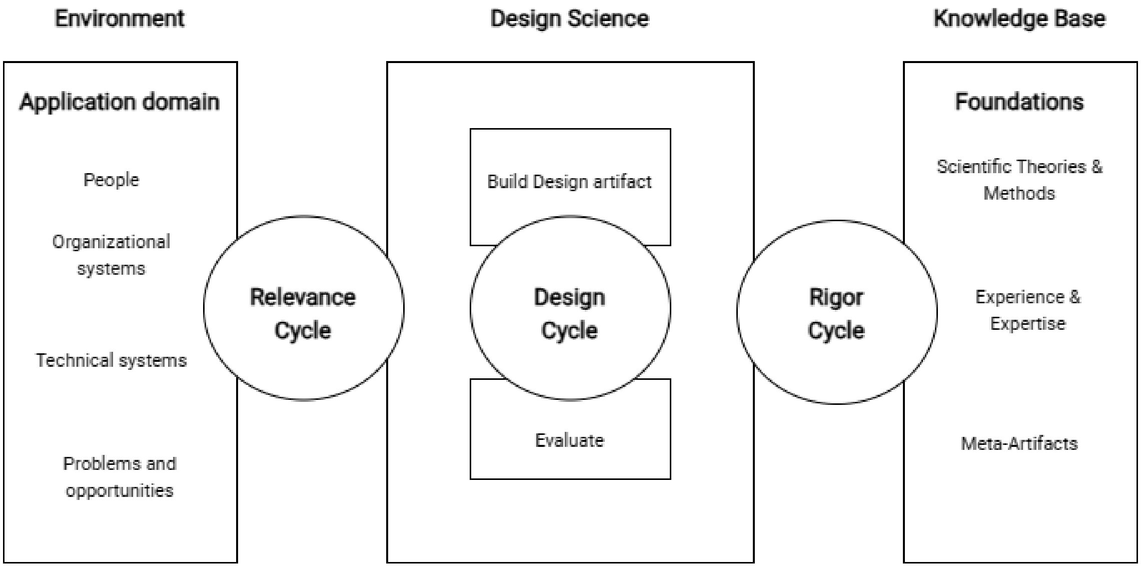


Figure 2.1: Design Science Model Based on (Hevner & Chatterjee, 2010).

2.4 Case Study

A case study is recognized as an empirical research method that investigates a phenomenon within its real-life context (Yin, 1981).

The fundamental objectives of conducting a case study encompass the description, explanation, and evaluation of a hypothesis. This approach can be utilized to gather data about a phenomenon, implement a specific tool, and evaluate its efficacy using interviews (Farshidi, 2020; Farshidi, Kwantes, & Jansen, 2023).

For this research, the guidelines proposed by Yin (Yin, 1981) were adhered to in the conduct and planning of the case studies.

The case study was conducted within an organization that is or was engaged in software production, enabling the testing of the created model using real-world data. Additionally, test data for the model was extracted from an online platform that provides freelance work worldwide, thereby enabling evaluation of the adopted approach.

The data collected offers insight into the model's efficacy and its potential utility for individuals seeking to estimate software development efforts. A detailed examination of the executed experiments and their corresponding results is delineated in Chapter 7.

2.5 Expert Interviews

In a quest to gain deeper insights into the field of research and identify meaningful areas of focus, several semi-structured interviews were conducted with experts in the field of SEE. This step was undertaken after the systematic literature review provided a clear overview of the research trends in this area.

The conducted interviews were semi-structured, meaning they included a predefined set of questions. However, these were not followed strictly, enabling open communication and fostering the generation of new ideas and suggestions for this research project. Furthermore, to ensure that all relevant information was included in the research, the ACM SIGSOFT standard was followed to guide the presentation and analysis of these interviews (Ralph et al., 2020).

The ultimate objective of the interviews extended beyond acquiring in-depth knowledge pertaining to software development and its associated cost estimation. It also aimed to appraise the factors discovered during the Systematic Literature Review (SLR), and potentially identify novel, significant elements that could contribute to this research. A comprehensive elaboration of the interviews conducted and their subsequent results are presented in Chapter 4.

Chapter 3

Literature Review

This chapter elucidates the results of the systematic literature review (SLR) delineated in Chapter 2. The SLR aimed to attain comprehensive knowledge of the software effort and cost estimation research field, including its prevailing trends. Additionally, the SLR aimed to address a number of research questions (**RQ1-4**) and collect relevant data that could inform this research and the formulation of the final estimation model. This incorporates factors and KPIs of significance in other research papers, models, and evaluation methods they have deployed. The latter enabled a comparative analysis with other methodologies, thereby facilitating the assessment of this study.

The guidelines proposed by (B. Kitchenham, 2004) were adhered to during the conduct of the SLR. Furthermore, the systematic literature review performed by (Farshidi, Jansen, & van der Werf, 2020) served as a reference when undertaking the SLR.

The ensuing sub-chapters elaborate on the process, rationale, and salient findings of the executed SLR. Additionally, the comprehensive collection of research papers and resulting findings from the SLR are documented in (Mrvar, 2023).

3.1 Data Sources and Search Strategy

The overall search strategy process was detailed in Chapter 2. It essentially comprised two search methods: initial hypothesis and automatic search. The initial hypothesis search facilitated the collation of the initial set of papers, which subsequently generated a search term from their common keywords. This search term was employed to automate data gathering. The exhaustive search process will be explored in the following sub-chapter.

The primary sources harnessed in this search encompass digital libraries, namely ACM Digital Library, Springer Publishing, IEEE Xplore Digital Library, and ScienceDirect. The focus was predominantly on these four libraries as they offer high-quality papers, lending significant value to the scientific community. It is noteworthy that Google Scholar[1] was not utilized during the automatic search process owing to its tendency to yield a high degree of irrelevant studies and grey literature. Furthermore, its potential for overlap with the other libraries employed in this SLR is substantial. However, Google Scholar was incorporated during the initial hypothesis search phase as it provided an overview of papers, thereby enabling the circumvention of grey literature.

[1] https://scholar.google.com/

3.2 Search Process

The procedural steps and the number of acquired papers throughout the search process are depicted in Figure 3.1. It is important to note that the figure represents the aggregate number of papers retrieved from the initial hypothesis and automatic search phases. During the initial hypothesis search phase, a collection of papers was acquired, relying on our pre-existing knowledge and theoretical underpinnings within this domain. To address the research questions, the following initial search queries were implemented to procure relevant papers:

- Software cost estimation
- Software effort estimation
- Information extraction for software cost estimation
- Analysis of software cost estimation accuracy
- Software cost estimation parameters
- Data extraction from Github
- Requirements extraction from Github

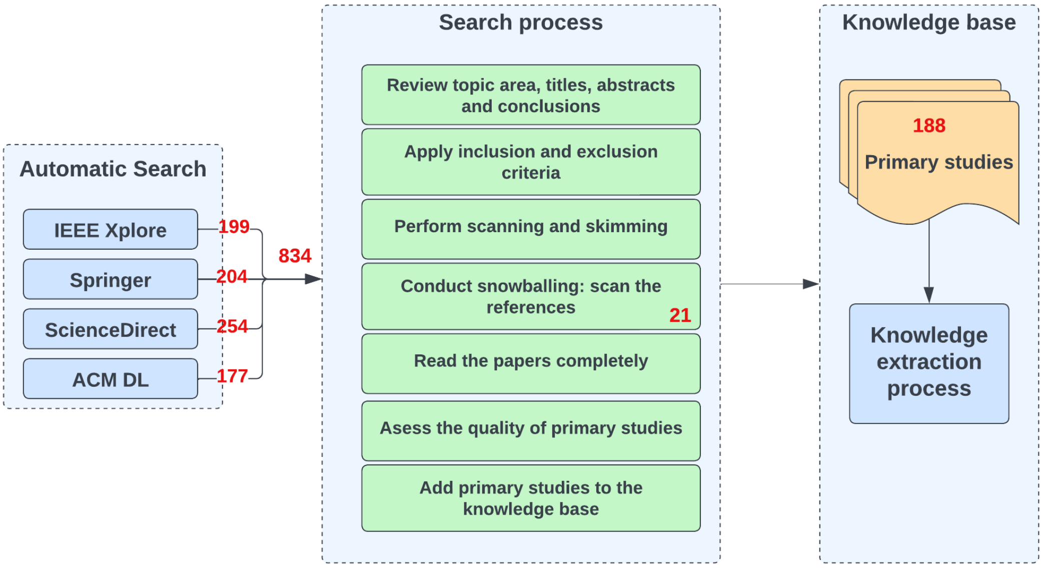

Automatic Search
IEEE Xplore
Springer
ScienceDirect
ACM DL
199
204
254
177
834
Search process
Review topic area, titles, abstracts and conclusions
Apply inclusion and exclusion criteria
Perform scanning and skimming
Conduct snowballing: scan the references
21
Read the papers completely
Asess the quality of primary studies
Add primary studies to the knowledge base
Knowledge base
188
Primary studies
Knowledge extraction process

Figure 3.1: SLR Search Process based on (Farshidi et al., 2020).

The initial hypothesis search phase yielded a repository of 223 papers, inclusive of 12 papers sourced from Google Scholar, not depicted in Figure 3.1. These papers were meticulously cataloged in a spreadsheet along with all relevant information as detailed in Chapter 2. Upon assembling this collection, their generic keywords were leveraged to formulate a search term subsequently used in the automatic search phase. The employed search term is provided below:

(”software effort estimation” OR ”software cost estimation” OR ”effort estimation” OR ”cost estimation” OR ”software estimation”) AND (”machine learning” OR ”neural network” OR ”genetic algorithm” OR ”artificial neural network”) AND (”software development” OR ”software engineering” OR ”agile software development” OR ”agile development”)

This search term encapsulated the latest trends in software effort and cost estimation, along with the most pertinent terms relative to this study, thereby automating the collection of papers from the digital libraries discussed in the preceding section. The results of the automatic search were exported in CSV or Bibtex format, facilitating an efficient method for collecting and cataloging those documents within the spreadsheet. Following this, duplicates and incomplete or corrupted entries were removed. The automatic search phase amassed an additional 623 papers, culminating in a total of 846 papers obtained throughout the manual and automatic search phases. It is noteworthy that the emphasis was predominantly on papers published post-2010. However, a few older papers were included, deemed meaningful during the manual search phase or the snowballing phase.

During the SLR, some papers were added through the application of various techniques. The snowballing technique, for example, was employed, tracing the references of the already assembled papers and incorporating them into the SLR, provided they were found to be contributory to the identified research questions. This technique supplemented an additional 21 papers. A separate strategy entailed the manual addition of relevant papers discovered at any phase of the SLR. An extra 11 papers were incorporated in this manner.

In total, 878 papers were considered in the SLR. Following the application of inclusion/exclusion criteria and quality assessment, the final number of papers subjected to analysis was 188. The procedural steps and criteria for paper selection will be detailed in the ensuing subsections.

3.3 Inclusion and Exclusion Criteria

Upon the completion of both manual and automated search stages, an evaluative phase was initiated to investigate the assembled literature (inclusive of papers collected through the snowballing methodology). During this phase, the abstract and keywords of each paper were carefully analyzed, facilitating a preliminary judgment regarding their relevance to the research study at hand. Each of these articles was classified on an ordinal scale featuring four categories: None, Low, Medium, and High. The categorization of 'None' signified complete irrelevance of the paper to the research, whereas 'High' implied a substantial degree of relevance.

Following the completion of the screening process, the inclusion and exclusion criteria were applied to filter the relevant from the non-relevant literature. Several factors were considered while determining the applicability of a publication to the Systematic Literature Review (SLR). These encompassed the language of the paper (English), its availability, its relevance (ranging from None to High), the year of publication, the number of citations, and the ranking of the conference or journal where the paper was presented.

Subsequent to the contemplation of the above-mentioned parameters, a score was assigned to each paper. For instance, the score decreased with each passing year since the paper's publication, while it escalated commensurate to the number of citations garnered by the paper. Upon calculation of scores, a predetermined threshold was applied to shortlist the literature for the SLR.

3.4 Quality Assessment

Post the application of the inclusion and exclusion criteria, the quality of the selected papers necessitates evaluation (B. Kitchenham, 2004). This evaluation serves as an extension of the inclusion and exclusion criteria, presenting a detailed, focused examination of each study. The primary emphasis of this research was on the quality assessment criteria outlined in Table 3.1.

Each paper was systematically reviewed and annotated by responding to quality assessment questions with distinct dichotomous outcomes, 'Yes' or 'No'. For instance, if a given paper included research questions, the corresponding field in the datasheet was marked as 'Yes' for that paper.

Finally, scores were calculated based on these responses, and a cut-off threshold was established to exclude papers that failed to meet the defined quality parameters. Consequently, a total of 188 primary studies qualified for consideration in this SLR.

Quality assessment	Definition
Clear problem statement	Does the study contain a clear problem statement?
Research questions	Does the study contain research questions?
Clear research challenges	Are the clear research challenges of the study explained?
A clear statement of findings	Are the research results and findings elaborated clearly and understandably?
Real-world use cases	Does the study contains a real-world use case?

Table 3.1: Quality Assessment Criteria Predicated on (B. Kitchenham, 2004).

3.5 Data Extraction

In accordance with the directives proposed by (B. Kitchenham, 2004), the process of data extraction and knowledge acquisition commenced subsequent to the implementation of the inclusion and exclusion criteria, and the quality assessment. The primary objective was to gather all relevant information from the shortlisted articles and catalog it systematically to enable a comprehensive understanding of the research.

Relevant data that was considered for this SLR encompassed the variables utilized by the studies to estimate software effort/cost, the employed models, and the evaluation methodologies. Additionally, it was ascertained that most studies harnessed existing datasets for the execution of Software Effort Estimation (SEE). Hence, these datasets, along with their distinctive attributes, were also extracted as they potentially harbor valuable information for model development, particularly in terms of the factors and attributes they encapsulate. Administrative details, which might prove to be instrumental in future investigations, were also gathered. These included: the name of the approach, the learning type, the approach's category and domain, the data acquisition method (manual or automatic), and the GitHub URL to the approach (or corresponding URLs for other platforms, if available).

The amassed information was intended to be employed subsequently in the design science phase, especially if deemed significant post-expert consultations. The identified factors were utilized to train our model and estimate the effort/cost associated with software development. The same applicability pertains to the models and evaluation methods. The prevalence of specific models could indicate their success and potential suitability for the approach under consideration.

3.5.1 Influential Variables and Key Performance Indicators

Within the confines of this study, we employ the term "factors" to denote influential variables or attributes that can markedly affect the cost estimation of software projects. These variables, determined through an exhaustive literature review, constitute an integral component of this study. Our objective in identifying these variables and Key Performance Indicators (KPIs) is to comprehend the aspects researchers have considered imperative while estimating the cost of software projects. Additionally, certain identified variables were utilized to extract data from online software repositories, thereby creating a dataset. This dataset subsequently served as a foundation for the development of our final estimation model.

Throughout the SLR, 254 factors and KPIs were identified, which have been employed in the analyzed approaches. Upon identification, these factors were systematically organized and grouped, culminating in a finalized list of 106 factors. The complete process of the procedure and the reasoning behind it will be elaborated on later.

Table 3.2 delineates the identified factors and KPIs, along with their corresponding frequency of occurrence in the reviewed literature. The values in the table are arranged in decreasing order of frequency, implying the most prevalent factors are positioned in the top left quadrant. It is imperative to note that factors mentioned only once were dismissed and hence do not feature in the table, as they were deemed infrequent and consequently, irrelevant.

While most factors are self-evident, a few warrant additional elucidation. For instance, a Use Case Point is a methodology for estimating the size and effort of software development based on the software's Use Case (Effendi, Setiawan, & Rasjid, 2019). The function point, as proposed by (A. J. Albrecht, 1979), serves as a "unit of measurement" to quantify the amount of business functionality provided to a user by an information system. Story points, typically used in agile development, are defined as a unit of measure employed to express an estimate of the overall effort necessary for the comprehensive implementation of a task (Scott & Pfahl, 2018).

The comprehensive list and definitions of the factors depicted in the table are accessible in (Mrvar, 2023).

Factor/KPI	#	Factor/KPI	#	Factor/KPI	#	Factor/KPI	#
Use Case Points	18	Special user training facilities	5	Collocation of the whole team	3	No. of days in one iteration	3
Team Experience	17	Concurrency	5	Facility with proper agile-style work environment	3	No. of working days per month	3
Effort time	16	Distributed Systems	5	Reward system appropriate for agile	3	No. of working hours per day	3
Function points	11	Provide direct access for third parties	5	Managers who have light-touch or adaptive management style	3	Customer Type	3
Story point	10	Object oriented experience	5	Following agile-oriented requirement management process	3	Used Methodology	2
Language	9	Stable requirements	5	Following agile-oriented project management process	3	Language experience	2
Security	9	Part-time workers	5	Following agile-oriented configuration management process	3	Language type	2
Project size (KLOC)	9	Actor weight	4	Strong communication focus with daily face-to-face meetings	3	Required reliability	2
Environmental factors	9	Application Type	4	Honoring regular working schedule – no overtime	3	Platform volatility	2
Motivation	9	Manager Experience	4	Strong customer commitment and presence	3	Developer platform	2
Requirement	9	Developer familiarity	4	Customer having full authority	3	Business function size	2
Complexity	8	Technical factors	4	Well-defined coding standards up front	3	Target technology	2
Application experience	8	Easy installation	4	Pursuing simple design	3	Constraints	2
Reuse	8	No. of user stories	4	Rigorous refactoring activities	3	Scope	2
Type/domain	7	Management	4	Right amount of documentation	3	Usability	2
Operational ease for users	7	Actor Type	3	Regular delivery of software	3	Response adjectives	2
Resource constraints	6	Development Type (enchanement/ new development)	3	Delivering most important features first	3	Familiar with Objectory	2
Analyst capability	6	Personnel continuity	3	Correct integration testing	3	Estimated points from Planning Poker	2
Team cohesion	6	System size	3	Appropriate technical training to team	3	Story card priority	2
Customer communication	6	DBMS (database M system)	3	Project nature being non-life-critical	3	BLI complexity	2
Complex internal processing	6	Personel capability	3	Project type being of variable scope with emergent requirement	3	Familiar with RUP	2
Task size / Storysize	6	Easy to change	3	Projects with dynamic, accelerated schedule	3	Commit (github)	2
Multisite development	5	Strong executive support	3	Projects with small team	3	Active day (github)	2
Transaction volume	5	Committed sponsor or manager	3	Projects with no multiple independent teams	3	Productivity factor	2
Team Size	5	Cooperative organizational culture instead of hierarchal	3	Projects with up-front cost evaluation done	3	Request Type	2
Portability	5	Oral culture placing high value on face-to-face communication	3	Projects with up-front risk analysis done	3		
End user efficiency	5	Organizations where agile methodology is universally accepted	3	No. of SP completed in an iteration	3		

Table 3.2: Identified factors and Key Performance Indicators (KPIs).

3.5.2 Models and Methods

Various techniques and models have been designed concerning software effort and cost estimation. These range from model-based approaches like SLIM and COCOMO to regression and learning-based ones such as neural networks and OLS (Boehm et al., 2000).
The SLR uncovered several learning approaches like regression, neural networks, and NLP (Natural Language Processing), among others. In total, 178 distinct models and methods were gathered and analyzed. Table 3.3 displays the most frequently utilized models and the frequency of each method's application in the literature. It's important to note that models and methods were only included if they were used at least twice.
Apart from learning-based approaches, a few feature selection algorithms were also collated, including the Genetic Algorithm (GA) and Particle Swarm Optimization (PSO) (Bilgaiyan, Mishra, & Das, 2016). The SLR also found instances of other algorithmic approaches and models utilized in software effort and cost estimation. However, this study mainly emphasized learning approaches that could be potentially useful for the estimation model intended to be developed.
The exhaustive list and the definition of each gathered model can be found in (Mrvar, 2023).

3.5.3 Evaluation Methods

Evaluation methodologies were collected and analyzed to determine how the authors of the selected papers assessed their approaches and the underlying rationale. Comprehending these methods enabled us to juxtapose our results with other approaches, thereby facilitating insights into the effectiveness of our estimation model with other models in this field.
In total, this SLR identified 86 evaluation methods, 43 of which were used more than once, and 32 were employed more than twice. Figure 3.2 illustrates the distribution of the 20 most frequently employed evaluation methods pinpointed during the SLR process. Other identified methods were used less than five times.
The preeminent evaluation method is MMRE (Mean Magnitude Relative Error), utilized in 78 papers. This is followed by PRED(p) (Prediction at Level p), used in 60 papers. Subsequent in line are MAE (Mean Absolute Error), MRE (Magnitude of Relative Error), SA (Standardized Accuracy), MdMRE (Median Magnitude Relative Error), and so forth.
The exhaustive list of the collected evaluation methods and their definitions can be accessed in (Mrvar, 2023).

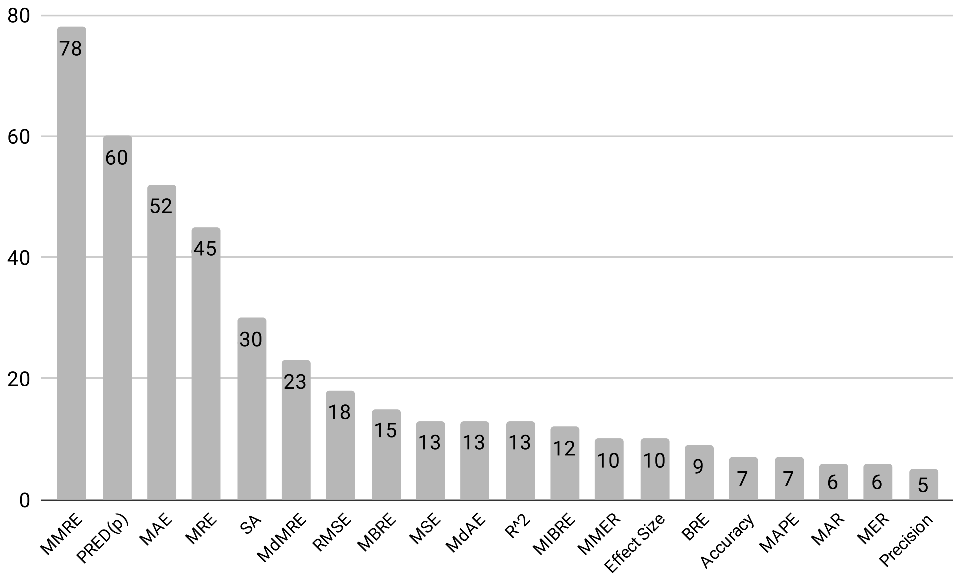
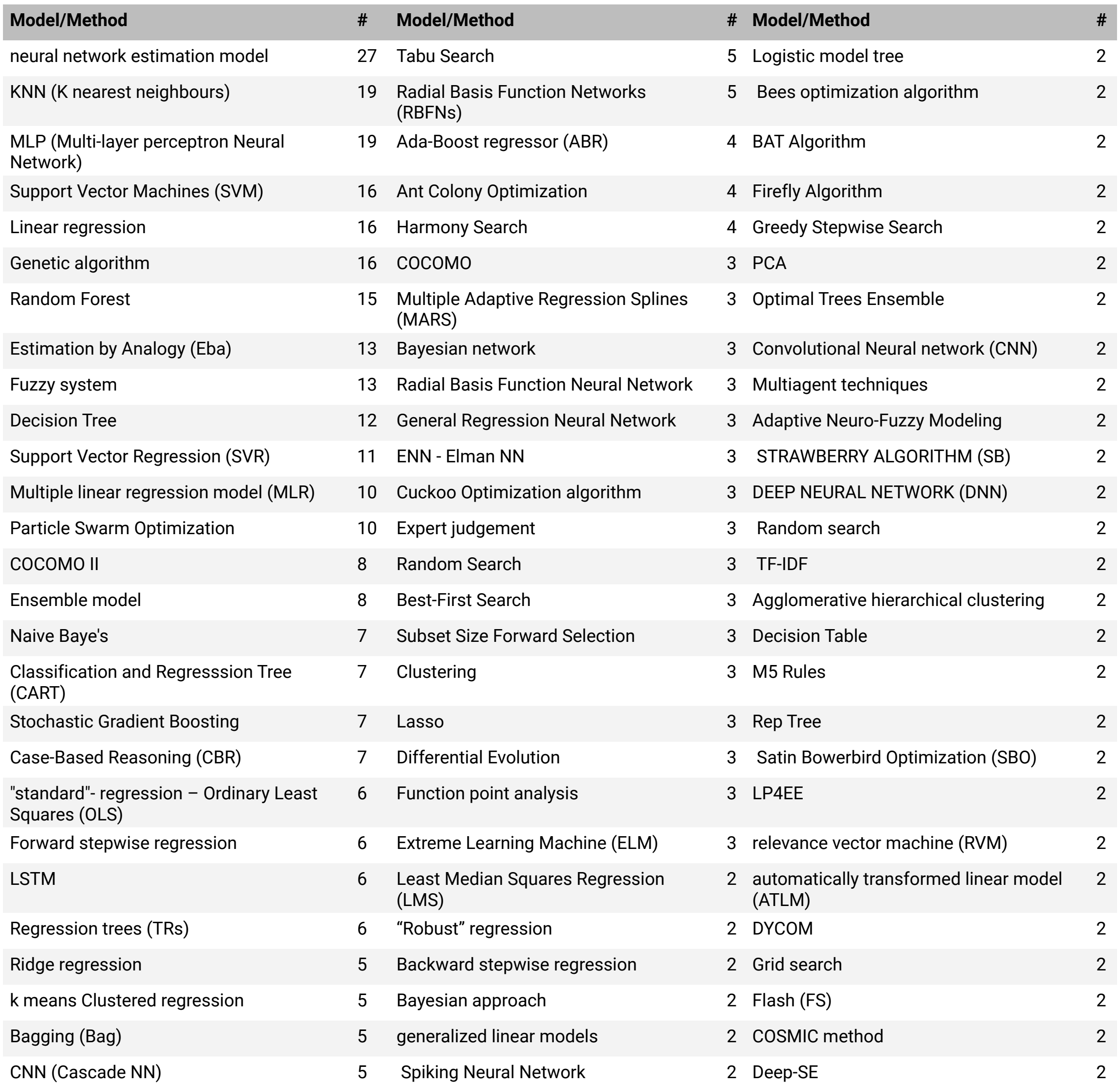


Figure 3.2: Distribution of Collected Evaluation Methods.



Model/Method	#	Model/Method	#	Model/Method	#
neural network estimation model	27	Tabu Search	5	Logistic model tree	2
KNN (K nearest neighbours)	19	Radial Basis Function Networks (RBFNs)	5	Bees optimization algorithm	2
MLP (Multi-layer perceptron Neural Network)	19	Ada-Boost regressor (ABR)	4	BAT Algorithm	2
Support Vector Machines (SVM)	16	Ant Colony Optimization	4	Firefly Algorithm	2
Linear regression	16	Harmony Search	4	Greedy Stepwise Search	2
Genetic algorithm	16	COCOMO	3	PCA	2
Random Forest	15	Multiple Adaptive Regression Splines (MARS)	3	Optimal Trees Ensemble	2
Estimation by Analogy (Eba)	13	Bayesian network	3	Convolutional Neural network (CNN)	2
Fuzzy system	13	Radial Basis Function Neural Network	3	Multiagent techniques	2
Decision Tree	12	General Regression Neural Network	3	Adaptive Neuro-Fuzzy Modeling	2
Support Vector Regression (SVR)	11	ENN - Elman NN	3	STRAWBERRY ALGORITHM (SB)	2
Multiple linear regression model (MLR)	10	Cuckoo Optimization algorithm	3	DEEP NEURAL NETWORK (DNN)	2
Particle Swarm Optimization	10	Expert judgement	3	Random search	2
COCOMO II	8	Random Search	3	TF-IDF	2
Ensemble model	8	Best-First Search	3	Agglomerative hierarchical clustering	2
Naive Baye's	7	Subset Size Forward Selection	3	Decision Table	2
Classification and Regresssion Tree (CART)	7	Clustering	3	M5 Rules	2
Stochastic Gradient Boosting	7	Lasso	3	Rep Tree	2
Case-Based Reasoning (CBR)	7	Differential Evolution	3	Satin Bowerbird Optimization (SBO)	2
"standard"- regression – Ordinary Least Squares (OLS)	6	Function point analysis	3	LP4EE	2
Forward stepwise regression	6	Extreme Learning Machine (ELM)	3	relevance vector machine (RVM)	2
LSTM	6	Least Median Squares Regression (LMS)	2	automatically transformed linear model (ATLM)	2
Regression trees (TRs)	6	“Robust” regression	2	DYCOM	2
Ridge regression	5	Backward stepwise regression	2	Grid search	2
k means Clustered regression	5	Bayesian approach	2	Flash (FS)	2
Bagging (Bag)	5	generalized linear models	2	COSMIC method	2
CNN (Cascade NN)	5	Spiking Neural Network	2	Deep-SE	2

Table 3.3: Identified Methods and Models.

3.5.4 Datasets

The SLR revealed that 123 out of the 188 analyzed articles employed existing datasets for the estimation of software effort/cost. Consequently, these datasets were assembled and analyzed. This rendered an option to subsequently test the developed estimation model using data from existing datasets, thereby facilitating a comparison of our results with other approaches utilizing identical data. Additionally, the attributes of these datasets were identified and extracted, proving to be an informative resource when extracting comparative data and creating the final prediction model.

In total, the conducted SLR identified 37 datasets, 25 of which were used at least twice. In this study, we selected to focus on datasets that were used minimally twice, which suggests that the dataset might be of value. It is noteworthy that datasets that couldn't be retrieved were excluded from further analysis and, therefore, are not featured in this study; this also applies to datasets with only a singular occurrence in the SLR.

Table 3.4 illustrates the chosen datasets along with their annual frequency of use, indicating the popularity of each dataset per annum. The data is sorted according to the total frequency shown in the last row. Based on the conducted SLR, it is evident that the ISBSG dataset emerged as the most popular, followed by the NASA2 dataset, etc.

Table 3.5 presents a description of the selected datasets (used at least twice). The table showcases the name of the dataset and the number of records and attributes each dataset comprises. Additionally, the source of each dataset, which also indicates the year of the latest publication of each dataset, was incorporated into the table. The information was arranged chronologically based on the frequency derived from the conducted SLR.

	ISBSG	NASA 2 (93)	Desharnais	COCOMO	Albrecht	Maxwell	China	Kemerer	Miyazaki	Kitchenham	Finnish	Telecom	NASA 60	TUKUTUKU	USP05-TF	SDR	Zia	Story point dataset	KotenGray	Dataset1	TAWOS
2022	6	4	5	1	5	5	6	4	2	3	2	2	0	1	0	1	0	1	0	0	2
2021	3	8	5	8	5	7	7	7	5	4	2	1	1	0	0	0	1	0	0	0	0
2020	7	5	5	4	4	2	1	1	1	0	1	0	2	1	0	0	1	1	1	1	0
2019	9	3	8	4	4	3	4	3	3	1	1	1	0	3	0	1	0	0	0	0	0
2018	9	10	7	7	6	5	5	4	4	3	1	2	2	0	3	0	1	1	0	0	0
2017	5	6	4	6	4	2	3	4	2	1	0	0	0	0	1	1	1	0	1	1	0
2016	2	1	2	2	3	3	3	2	3	1	1	0	0	0	0	0	0	0	0	0	0
2015	1	3	2	2	1	1	1	1	0	1	1	1	0	1	1	0	0	0	0	0	0
2014	2	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
2013	1	1	2	2	1	2	0	0	0	1	1	0	1	0	0	1	0	0	0	0	0
2012	1	1	0	1	1	0	0	1	1	0	0	1	0	0	0	0	0	0	0	0	0
2011	2	2	1	2	1	0	0	1	0	0	0	1	1	0	0	1	0	0	0	0	0
2010	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Total	49	45	41	39	35	30	30	28	21	16	10	9	7	6	5	5	4	3	2	2	2

Table 3.4: Yearly Distribution of Identified Datasets.

Data set	Records	Attributes	Source
ISBSG	11,128	264	(*International Software Benchmarking Standards Group*, 2022)
NASA 2 (93)	93	24	(Sayyad Shirabad & Menzies, 2005)
Desharnais	81	12	(Sayyad Shirabad & Menzies, 2005)
COCOMO81	63	91	(Sayyad Shirabad & Menzies, 2005)
Albrecht	24	8	(A. J. Albrecht & Gaffney, 1983)
Maxwell	62	27	(Maxwell, 2002)
China	499	19	(Sayyad Shirabad & Menzies, 2005)
Kemerer	15	8	(Kemerer, 1987)
Miyazaki	48	9	(Miyazaki, Terakado, Ozaki, & Nozaki, 1994)
Kitchenham	145	10	(B. Kitchenham, Pfleeger, McColl, & Eagan, 2002)
Finnish	38	9	(Shepperd & Schofield, 1997)
Telecom	18	4	(Shepperd & Schofield, 1997)
NASA60	60	17	(Sayyad Shirabad & Menzies, 2005)
TUKUTUKU	67	43	(Mendes, Mosley, & Counsell, 2003)
USP05-TF	76	15	(Sayyad Shirabad & Menzies, 2005)
SDR	12	25	(Sayyad Shirabad & Menzies, 2005)
Zia	21	9	(Ziauddin & Zia, 2012)
Story point dataset	16	3	(Choetkiertikul et al., 2019)
Dataset1	28	5	(Silhavy, Silhavy, & Prokopova, 2015)
TAWOS	44	10	(Tawosi, Al-Subaihin, Moussa, & Sarro, 2022)

Table 3.5: Description of Datasets and Their Sources.

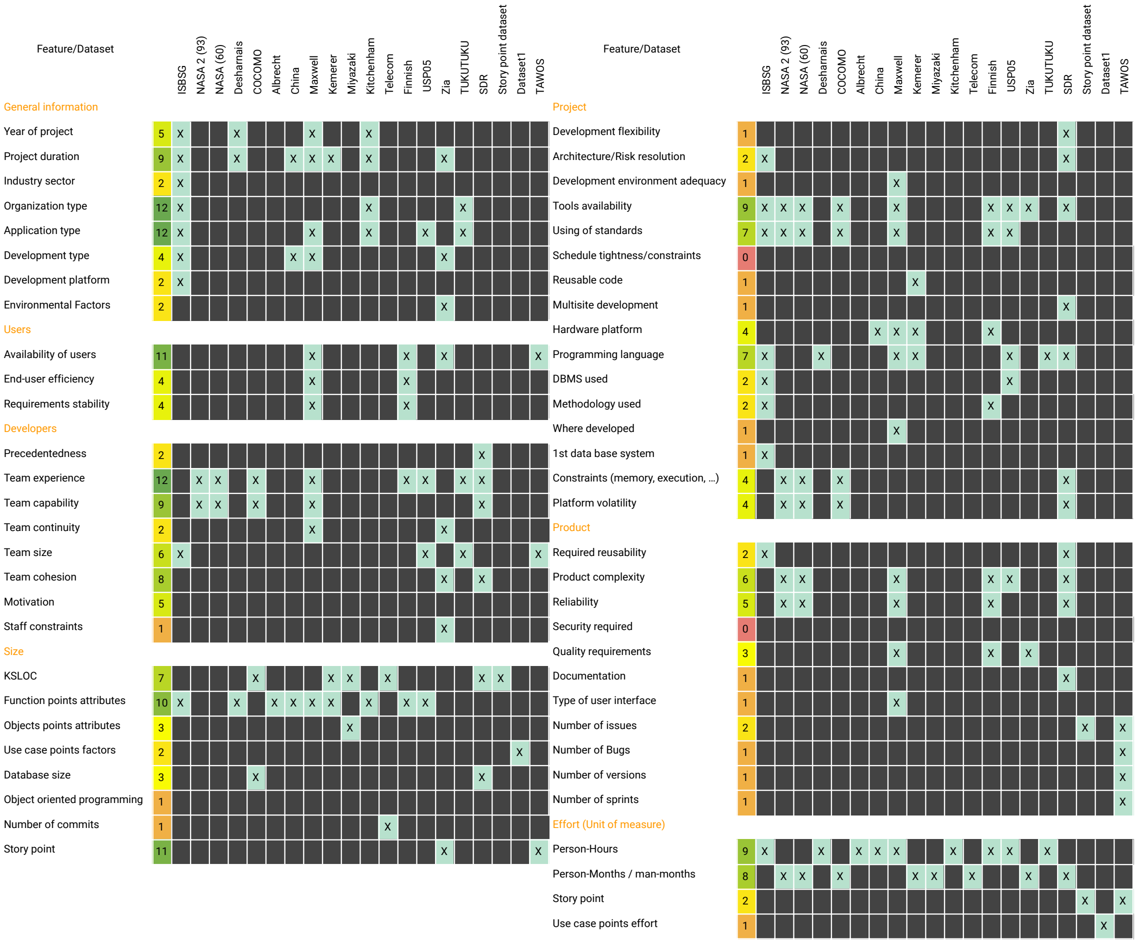
3.6 Data Analysis and Synthesis

The final step before presenting the findings involved analyzing and synthesizing the collected data. This included the features, models, datasets, and evaluation methods that were gathered. Additionally, the data was examined for accuracy and overall coherence.
The SLR yielded 254 features, making their organization a complex task, as different approaches employ different terminologies for what might essentially signify the same feature. Hence, the goal was to concentrate on a systematic approach and minimize bias as much as possible. First, features with only a single occurrence in the SLR were discarded since they were deemed insignificant. Second, features that could possibly imply the same concept were grouped. For instance, the following factors, mentioned across different approaches, were consolidated into one factor named **Team Experience**:

(”Team Experience”, ”Personnel Experience”, ”Developer Experience”, ”Personnel Capability”, ”Team’s Prior Experience”, ”Developer Reputation”, ”Skill Level Development”)

Every modification and decision was meticulously documented in a separate sheet. This approach trimmed down the number of features from 254 to 106.
In terms of analyzing and synthesizing data from models and evaluation methods, the primary focus was on documentation and recording definitions and categories of each method. This facilitated a clearer understanding of them and made their application more convenient when required.

Finally, different datasets were compared and their attributes were extracted. The goal was to derive insights from those attributes and consider them as an additional set of factors relevant to this research. (E. Mustafa & Osman, 2018) undertook an extensive comparison of 31 datasets they had gathered. Consequently, their study served as a starting point for comparing datasets discovered in this SLR. They notably extracted attributes from those datasets and grouped them by category and significance. This led to the final set of 48 collected attributes in 6 categories. Our research extended their results by incorporating additional datasets and attributes from the compiled datasets. Additionally, datasets that were inaccessible to us or those that were used less than twice were excluded. The results of (E. Mustafa & Osman, 2018) were expanded by including five additional features (Environmental Factors, Story point, Number of issues, Number of Bugs, Number of versions, and Number of sprints) and six datasets (Zia, TUKUTUKU, SDR, Story point dataset, Dataset1, and TAWOS).
A comprehensive comparison of the selected datasets can be found in Figure 3.6.



Feature/Dataset		ISBSG	NASA 2 (93)	NASA (60)	Desharnais	COCOMO	Albrecht	China	Maxwell	Kemerer	Miyazaki	Kitchenham	Telecom	Finnish	USP05	Zia	TUKUTUKU	SDR	Story point dataset	Dataset1	TAWOS
General information																					
Year of project	5	X			X				X			X									
Project duration	9	X			X			X	X	X		X				X					
Industry sector	2	X																			
Organization type	12	X										X					X				
Application type	12	X							X			X			X		X				
Development type	4	X						X	X							X					
Development platform	2	X																			
Environmental Factors	2															X					
Users																					
Availability of users	11								X					X		X					X
End-user efficiency	4								X					X							
Requirements stability	4								X					X							
Developers																					
Precedentedness	2																	X			
Team experience	12		X	X		X			X					X	X		X	X			
Team capability	9		X	X		X			X									X			
Team continuity	2								X							X					
Team size	6	X													X		X				X
Team cohesion	8															X		X			
Motivation	5																				
Staff constraints	1															X					
Size																					
KSLOC	7					X				X	X		X					X	X		
Function points attributes	10	X			X		X	X	X	X		X		X	X						
Objects points attributes	3										X										
Use case points factors	2																			X	
Database size	3					X												X			
Object oriented programming	1																				
Number of commits	1												X								
Story point	11															X					X

Feature/Dataset		ISBSG	NASA 2 (93)	NASA (60)	Desharnais	COCOMO	Albrecht	China	Maxwell	Kemerer	Miyazaki	Kitchenham	Telecom	Finnish	USP05	Zia	TUKUTUKU	SDR	Story point dataset	Dataset1	TAWOS
Project																					
Development flexibility	1																	X			
Architecture/Risk resolution	2	X																X			
Development environment adequacy	1								X												
Tools availability	9	X	X	X		X			X					X	X	X		X			
Using of standards	7	X	X	X		X			X					X	X						
Schedule tightness/constraints	0																				
Reusable code	1									X											
Multisite development	1																	X			
Hardware platform	4							X	X	X				X							
Programming language	7	X			X				X	X					X		X	X			
DBMS used	2	X													X						
Methodology used	2	X												X							
Where developed	1								X												
1st data base system	1	X																			
Constraints (memory, execution, …)	4		X	X		X												X			
Platform volatility	4		X	X		X												X			
Product																					
Required reusability	2	X																X			
Product complexity	6		X	X					X					X	X			X			
Reliability	5		X	X					X					X				X			
Security required	0																				
Quality requirements	3								X					X		X					
Documentation	1																	X			
Type of user interface	1								X												
Number of issues	2																		X		X
Number of Bugs	1																				X
Number of versions	1																				X
Number of sprints	1																				X
Effort (Unit of measure)																					
Person-Hours	9	X			X		X	X	X			X		X	X		X				
Person-Months / man-months	8		X	X		X				X	X		X			X		X			
Story point	2																		X		X
Use case points effort	1																			X	

Table 3.6: Comparison of Attributes in Selected Software Cost Estimation Datasets.

Chapter 4

Expert interviews

This chapter details the method employed for conducting and analyzing expert interviews, a vital component of the data collection process in this research. These interviews yielded significant insights into the realm of software cost estimation and aided in evaluating factors identified through the literature review in the preceding chapter. They also facilitated a better comprehension of the industry's landscape and the key challenges plaguing software cost estimation. The ACM SIGSOFT standard guided the presentation and analysis of these interviews, ensuring all critical information was incorporated into the research (Ralph et al., 2020).

The chapter begins by explaining the research method. This is followed by the process of identifying and selecting expert interviewees, the demographics of the chosen experts, and the execution of the interviews.

Subsequently, the Interview Protocol, outlined in appendix A, is discussed. This section expounds on the structure of the interviews and the categories of questions asked, thereby providing transparency in the data-gathering approach. It allows for a more comprehensive understanding of the methodology and the planning that went into the interviews. This is followed by a detailed account of how the interviews were conducted, which delves into the practical aspects of the interview process, the subsequent data analysis and collection, and the strategies for data recording and transcription. The communication methods utilized for the interviews, the typical duration, and the ethical considerations taken into account are also discussed. Furthermore, the methods for analyzing and interpreting the responses from the experts are outlined.

Finally, the chapter presents the findings from the interviews. These findings form a crucial part of the data employed in this study to develop a comprehensive estimation model for software cost/effort estimation. By evaluating factors gathered in SLR, a judgment was made on which factors should be included in the estimation model and which not.

4.1 Research Method

The research methodology employed for this study involved **semi-structured** interviews. This method offers a combination of systematic data collection with the flexibility to delve deeper into certain topics, guided by expert feedback and the natural progression of conversation (Fylan, 2005). Such an approach proves particularly beneficial when interviewing experts from varied domains or industries, where divergent opinions can potentially offer new insights throughout the interview. This is especially true for information not widely available or discussed in existing literature.

4.2 Sampling Strategy and Selection of Experts

LinkedIn[1], a professional networking platform, served as the primary medium for the selection of experts. The selection criteria included the individual's professional title or position, specific skills, academic qualifications, and years of experience in software development or related domains. Potential experts were evaluated and ranked based on their relevance to the research topic, primarily their direct involvement in software development and cost estimation procedures within their respective organizations.

Following the ranking process, potential participants were contacted directly via email. This correspondence introduced them to the study, inviting them to contribute their valuable insights through the interview. The email invitation emphasized the significance of their expertise to the study, assured them of confidentiality and discretion with their responses, and detailed the format and expected duration of the interview.

The final selection of experts encompassed diverse roles within the field, ranging from CEOs to product and project managers, as well as software developers. Such diversity ensured a comprehensive array of perspectives on software cost estimation. Ten experts consented to participate and were interviewed for this research. It is noteworthy that a total of 86 individuals were approached to partake in the interviews.

4.2.1 Participants' Demographics and Work Roles

A total of 10 participants were incorporated into the study, comprising individuals from different geographical locations, namely, the Netherlands (4 participants) and Slovenia (6 participants), to ensure a diverse representation.

An array of academic qualifications and professional roles were presented amongst the participants. From an academic perspective, 20% of the participants (2 individuals) possessed a Master's degree, while the majority, 80% (8 individuals), held a Bachelor's degree.

Analyzing their professional roles, the participants represented a broad spectrum of positions within the software industry. At the executive level, there were 2 CEOs who lead companies specializing in software production. Mid-management was represented by 4 participants who held roles as either project or product managers. Consultancy, another vital domain in the software landscape, was represented by 2 participants. The hands-on technical aspect was covered by one software developer and one CTO, ensuring that the views of those who actively create and oversee the technical architecture were also included.

The overarching intent in the selection of participants was to encompass a diverse set of background knowledge. This heterogeneity facilitated the collection of varied perspectives on the subject matter of software cost estimation, thereby enriching this study.

[1] https://www.linkedin.com/

4.3 Interview Protocol

The interview protocol was designed to gather in-depth insights into the experts' experiences and viewpoints on software cost estimation. As detailed in Appendix A, the interview process was divided into four main phases. The first phase involved a brief introduction to the project and the primary objective of the interview, followed by initial questions to understand the experts' professional backgrounds and the contextual aspects of their work.

In the third phase, the focus shifted to cost estimation. Questions in this phase aimed to elucidate the methods currently employed for cost estimation within the experts' organizations, the frequency of these estimates, the tools or platforms utilized in this process, the precision of these estimates, and the key factors or Key Performance Indicators (KPIs) considered crucial. During this phase, the experts were presented with a Google spreadsheet[2] containing a list of factors identified from the literature review detailed in chapter 3. They were requested to mark each factor as relevant (Y), potentially relevant (M), or not relevant. The final phase of the interview comprised closing questions, encompassing feedback on the research project, the experts' willingness to adopt a model for estimating software cost/effort, and preferences regarding the confidentiality of their responses and other personal information.

4.4 Conducting the Interviews

The interviews were facilitated online via the Microsoft Teams[3] and Google Meet[4] platforms, selected for their convenience for the interviewees and their functionality to record conversations. These recordings were subsequently transcribed using the SharePoint[5] transcription feature to ensure accuracy in documenting the experts' inputs. The interview protocol served as the guiding structure throughout the interviews, enabling the experts to articulate their expertise and views on the topic consistently. In addition to this, they were shown the Google spreadsheet containing the list of factors identified from the literature review and asked to provide their assessment on the relevance of each factor to software cost estimation, as mentioned previously. Furthermore, GitHub metadata[6] factors were incorporated into the list of factors. These factors, derived directly from GitHub repositories, could potentially be pertinent for software cost estimation and provide a comprehensive perspective for the extraction of data for our estimation model.

Experts were also given the opportunity to contribute additional factors they deemed relevant for software effort estimation. These factors were subsequently appended to the list under the label "Added after interviews". This interactive element of the interviews permitted a more thorough exploration of the experts' perspectives, surpassing the scope of pre-structured interview questions and the earlier conducted SLR.

[2] https://docs.google.com/

[3] https://www.microsoft.com/en-us/microsoft-teams/group-chat-software

[4] https://meet.google.com/

[5] https://www.microsoft.com/en-ww/microsoft-365/sharepoint/collaboration

[6] https://docs.github.com/en/rest/meta

4.4.1 Data Analysis

The comprehensive interview process included both audio recording and note-taking, ensuring no significant details were missed. The primary aim of these interviews was to assess and validate the factors identified during the Systematic Literature Review (SLR) and to potentially uncover new factors deemed critical by the experts.

In order to streamline the vast amount of data gathered from the interviews, a systematic approach was adopted. Participants were presented with a spreadsheet where each factor could be classified as either irrelevant (ignored), potentially significant (marked as "Maybe"), or undoubtedly significant (marked as "Yes").

Subsequent to the completion of the interviews, the data was subjected to an analysis process involving a simple arithmetic operation. Scores were assigned based on the frequency of the "Maybe" and "Yes" responses for each factor. This frequency-based scoring system was adopted to identify the most commonly acknowledged factors across the participant group. Following an exhaustive analysis, only factors that were identified as relevant ('Yes' or 'Maybe') at least three times were considered significant for the purpose of this study. Factors that did not meet this threshold were classified as less relevant to our research.

This approach allowed us to isolate the most impactful factors and assured that our software cost estimation model was grounded in the factors that industry experts considered paramount. The comprehensive list of factors and the resulting data from the interviews are presented in the subsequent section.

4.5 Results and Findings

The conducted interviews have afforded valuable insights into the landscape of software development and associated cost estimation. Notably, the diversity in the participants' backgrounds underscored that priorities in this area could be significantly influenced by whether or not an individual has a technical background.

The following section outlines the results from these interviews, capturing the most compelling comments and quotes made by the professionals. Moreover, it presents the factors relevant to software cost estimation, as identified through the SLR and later evaluated through these discussions.

4.5.1 Insights Supporting Key Points

The expert interviews conducted during this study yielded valuable insights into the field of software cost estimation. Broadly, the consensus underscored the significance of the field in relation to software development, although some experts expressed concerns about the feasibility of applying machine learning approaches for cost estimation, considering the substantial impact of social factors on the final cost of software development. The range of opinions on the efficacy of current estimation methods was diverse, with some reporting satisfactory accuracy in their organizations, typically when factoring in risk elements into their calculations, while others disclosed frequent discrepancies between estimates and actuals, which can lead to significant operational challenges and client dissatisfaction. The following section presents some of the noteworthy perspectives gathered during the interviews, each illuminating a unique facet of software cost estimation. To maintain confidentiality, the experts are anonymized and identified by numbers.

Expert 1, an individual working in a consultancy, shared their organization's approach to cost estimation: *"We usually make a work breakdown structure and try to estimate that, which is not always ideal but it is a good estimation"*. This expert indicated that they estimate the time needed for each smaller piece of work, subsequently factoring in a risk component. They further noted, *"A lot of estimation is not only the actual coding, such as design, configuration, looking at how things actually work, and idle time"*, expressing skepticism about the accuracy of machine learning techniques that solely rely on repository data for estimation.

Expert 7, a CTO in a private organization, offered an alternative perspective on software cost estimation: *"You can get pretty accurate with cost estimation, but for me, cost estimation is only half of what you're looking for. Because you are looking at your return on investment which is more difficult and in my opinion far more important."* The expert emphasized the idea that a company typically has more ideas than resources to implement them, hence turning the challenge into a prioritization task—deciding what merits the time and effort of development.

Lastly, Participant 4, a project manager in a consultancy company, expressed skepticism about estimation, stating, *"My opinion is quite dismissive of the meaningfulness of estimation."* They suggested that cost estimation might not be as pivotal in their line of work as conventionally perceived and could even be more trouble than it's worth due to inherent unpredictability. They also raised the question, *"Why is this software needed?"*, advocating for the consideration of the business value of the software being built as a more pertinent principle to follow.

4.5.2 Results from the Interviews

The expert interviews yielded significant insights into the relevance and importance of various factors in software cost estimation. A summary of the experts' evaluations on the identified factors can be seen in table 4.1. This table enumerates the factors extracted from the literature review and documents each expert's assessment of these factors. Experts are anonymized and represented in the header row for privacy reasons.

The numeral placed between the factor and its corresponding evaluations signifies the count of 'Yes' or 'Maybe' indications, encapsulating the relative importance and credibility of each factor based on the aggregated responses. The array of responses captured in the table is notably diverse, mirroring the varied experiences and viewpoints of the experts consulted.

Certain factors, such as 'Project Duration', 'Development type', and 'Team experience' were commonly indicated as significant by the experts, receiving either a 'Yes' or a 'Maybe' from the majority. This highlights their relative prominence in the domain of software cost estimation.

Conversely, certain factors like 'Environmental Factors' were often overlooked by the experts, suggesting their potential lower impact on determining software development efforts.

The detailed evaluations from each expert, as presented in figure 4.1, establish a robust foundation for our ensuing analysis and model formulation. It is worth noting that factors incorporated after the interviews were treated identically as the prior ones. Owing to the iterative nature of the data collection, there is a possibility that some interviewees may have overlooked factors mentioned in subsequent interviews. Nevertheless, a majority of participants expressed satisfaction with the quantity and diversity of the factors and attributes already in place.

Factor/Interview		Expert 1	Expert 2	Expert 3	Expert 4	Expert 5	Expert 6	Expert 7	Expert 8	Expert 9	Expert 10
General information	37										
Year of project	5			M			M	Y	Y		Y
Project duration	8			Y	Y	Y	Y	Y	Y	Y	Y
Industry sector	5					Y	Y	Y	Y	Y	
Application type (mobile, web...)	4				M		M		Y	Y	
Development type	9	Y		Y	Y	Y	Y	Y	Y	M	Y
Development platform (IDE, OS)	4				Y	Y		Y	Y		
Environmental Factors	2			Y			M				
Users	28										
Availability of users	7		Y	Y	Y		Y	Y	Y	M	
End-user efficiency	5		Y		Y		Y		M	Y	
Requirements stability	8		Y	Y	Y	Y	Y	Y	Y	Y	
Customer communication	8		Y	Y	Y	Y	Y	Y	Y	Y	
Developers	92										
Precedentedness	8	M			Y	Y	Y	Y	Y	M	Y
Team experience	10	M	Y	Y	Y	Y	Y	Y	Y	M	Y
Manager Experience	4			Y			Y	Y	Y		
Analyst Capability	6			Y	Y	Y	Y	Y	Y		
Team capability	9	M	Y	Y	Y	Y	Y	Y	Y	M	
Team continuity / context switching	8	M	Y	Y	Y		Y	Y	Y		Y
Team size	6	M	M	Y	M			Y	Y		
Team cohesion	9	M		Y	Y	M	Y	Y	Y	M	Y
Motivation	8		Y	Y	M	M	Y	Y	Y		Y
Staff constraints	6			Y	Y	Y	Y	Y	Y		
Application experience	6			Y		Y	Y	Y	Y		Y
Part-time workers	2			Y					M		
Management (general)	6			Y	M	M	Y		Y		Y
Active day (github)	4				Y	M	Y		Y		
Size	46										
Project size (KSLOC)	10	M	Y	Y	Y	Y	Y	Y	Y	Y	Y
Function points attributes	4					Y	Y	Y	Y		
Objects points attributes	4					Y	Y	Y	Y		
Use case points factors	5				M	Y	M	Y	Y		
Database size	4				M		M		Y		Y
Object oriented programming	2						Y				Y
Number of commits	5				M	M		m	Y		Y
Story point	5				Y	Y	M	Y	Y		
Task size / Story size	7			Y	Y	M	Y	Y	Y		Y
Project	97										
Development flexibility	7		Y	Y	Y	Y	Y	Y		Y	
Architecture/Risk resolution	9	Y	Y	Y	Y	Y	Y	Y	Y	M	
Development environment adequacy	4			M	Y	Y			M		
Tools availability	7			Y	Y	Y	Y	Y	Y		Y
Using of standards	5			Y	M		Y		Y	Y	
Schedule tightness/constraints	6			Y	Y	Y		Y	Y		Y
Reusable code	9		M	Y	M	M	Y	Y	Y	Y	Y
Multisite development	4			Y		Y	Y		Y		
Hardware platform	5			Y		M	Y		M	Y	
Programming language	6	Y				M		Y	Y	Y	Y
DBMS used	4	Y						Y	Y	Y	
Methodology used	7	Y		Y	M		Y	Y	Y		Y
1st data base system	3						Y		Y	Y	
Constraints (memory, execution, ...)	8	Y		Y	Y	Y		Y	Y	Y	Y
Platform volatility	3					Y		Y	Y		
Task/BLI Complexity	10	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y

		Expert 1	Expert 2	Expert 3	Expert 4	Expert 5	Expert 6	Expert 7	Expert 8	Expert 9	Expert 10
Added after interviews	25										
Project sponsor (CEO, customer)	3			Y	Y				Y		
Project manager	3			Y					Y		Y
Architect	4			Y	Y				Y		Y
Technical lead	3			Y	Y				Y		
Person week	2							Y	Y		
Uncertainty (example: high, medium, low)	5	Y						Y	Y	Y	Y
Number of interfaces	2	Y							Y		
Number of connections to the outside world	2	Y							Y		
Customer feedback	1								Y		
Database structure	2									Y	Y
Code complexity	2									Y	Y
Github Metada	138										
Repository name	2						M		Y		
Description	5						Y	M	Y	M	Y
Number of stars	6				Y	M	Y		Y	Y	Y
Number of forks	6				Y	M		M	Y	M	Y
Number of open issues	7				Y	Y	Y	Y	Y	Y	Y
Number of closed issues	7				Y	Y	Y	Y	Y	Y	Y
Number of contributors	8	M			Y	Y	M	Y	Y	Y	Y
Programming language	8	Y	Y	Y		Y	M		Y	Y	Y
License	3				Y		Y		Y		
Last updated date	4						Y		Y	Y	Y
README file	6	Y					M	M	Y	M	Y
Code of conduct	3						Y		Y	M	
Contributing guidelines	3						M		Y	M	
Security policy	5				Y		Y	Y	Y	Y	
Labels	4							Y	Y	Y	Y
Projects	5					Y	M	M		Y	Y
Milestones	6				Y	Y	Y	M		Y	Y
Issues	8		Y		Y	Y	Y	Y	Y	Y	Y
Pull requests	5				Y			Y	Y	M	Y
Wikis	7	Y			Y		Y	M	Y	Y	Y
Releases	6				Y	Y	M		Y	Y	Y
Commits	7				Y	M	M	Y	Y	Y	Y
Lines of code	6				Y		M	Y	Y	Y	Y
All of the packages	4						M	Y		Y	Y
Total hours used on repository	7		Y		Y	Y	Y	Y		Y	Y
Product	92										
Required reusability	9		M	Y	Y	Y	Y	Y	Y	Y	Y
Product complexity	10	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Required Reliability	8			Y	Y	Y	Y	Y	Y	Y	Y
Security required	10	Y	Y	M	Y	Y	Y	Y	Y	Y	Y
Quality requirements	8			Y	Y	Y	Y	Y	Y	Y	Y
Documentation	5		Y	Y				Y	Y	Y	
Type of user interface	5	Y					Y	Y	M	Y	
Number of issues	4					Y		Y	Y		Y
Number of Bugs	5				Y	Y		Y	Y		Y
Number of versions	4					Y			Y	Y	Y
Number of sprints	2							M	Y		
Transaction volume	5	Y						Y	Y	Y	Y
Portability	5						Y	Y	Y	Y	Y
Easy installation	5					Y	Y	M	Y		Y
Easy to change	7			Y		Y	Y	y	Y	Y	Y
Effort (Unit of measure)	15										
Effort time (general)	7	Y	Y	Y	Y	Y	Y	Y			
Person-Hours	4				Y				Y	Y	Y
Person-Months / man-months	2				Y				Y		
Story point	1				Y						
Use case points effort	1				Y						

Table 4.1: Interview Results.

4.6 Discussion

The primary aim of the interviews was to gather and evaluate a collection of factors, rather than to encompass the actual practices of cost estimation and their efficacy. It is important to note that during the Systematic Literature Review (SLR) phase, there were very few studies identified that endeavored to evaluate the factors that could influence the cost of a software project. However, several notable studies that were considered during the SLR shared similarities with the final goals of the interviews conducted in this study.

Usman et al. (Usman, Mendes, & Börstler, 2015) conducted a survey focused on the impacts of agile practices on software cost, where the emphasis was primarily placed on techniques, and to a lesser extent, effort predictors such as size and cost metrics.

Another study bearing resemblance to our interviews was carried out by (Chow & Cao, 2008), where the researchers aimed at identifying critical success factors in agile software development projects. These success factors can potentially be related to the cost metrics or factors considered in our study. Intriguingly, (Stankovic, Nikolic, Djordjevic, & Cao, 2013) built upon the same factors to create a survey study in agile software projects in IT companies in the former Yugoslavia. It is important to highlight that all of these mentioned studies were reviewed in the SLR as detailed in Chapter 3.

The absence of studies evaluating cost-impacting factors underscores the potential value of the approach taken in this research. It is our hope that future researchers may find this method useful in their evaluation of different factors or in conducting a greater number of interviews to generate more reliable results. Additionally, the insights llearned from the interviews conducted for this study could potentially enhance the understanding of the factors influencing the cost of software development.

4.7 Potential Researcher Bias and its Impact

Researcher bias can manifest in various forms and may inadvertently influence the findings of a study. In this particular context, potential sources of bias may stem from our selection process and the structure of the interview process.

Our selection of participants, predominantly approached through LinkedIn, could potentially introduce a selection bias. As we reached out directly to potential interviewees, it is conceivable that personal connections or networks might have played a role in our selection process, inadvertently leading to a non-representative sample of experts in the field of software cost estimation.

Another potential source of bias could stem from the structuring of our interviews. We utilized a predefined set of questions for the interviews, which, while ensuring consistency across interviews, may inadvertently introduce bias. It is possible that the formation of these questions was influenced by our own preconceptions or hypotheses, introducing a form of confirmation bias.

Additionally, the participants were presented with a predefined list of factors to evaluate. The detailed explanation and analysis of these factors required substantial time from the participants, which may have unintentionally constrained their capacity to express their opinion adequately or consider other factors they previously deemed significant. This approach might introduce bias in our findings as the opinions might not fully represent the experts' viewpoints or may have inadvertently omitted other potentially relevant factors.

Recognizing these potential sources of bias is important to accurately interpret the findings of our study. Further research could aim to address these potential biases, for instance, by employing a more diverse selection process or adopting a more open-ended interview structure that allows participants greater flexibility to express their perspectives.

Chapter 5

Data Preparation and Extraction

This chapter delineates the initial stages of our estimation model development, commencing with the crucial steps of data preparation and extraction. A thorough and comprehensive data preparation phase is essential as it forms the groundwork for the ensuing software cost estimation model. This phase categorizes factors identified from SLR and expert interviews into two primary categories: manual and automatic. This categorization is contingent on whether the factor can be directly or indirectly procured from the GitHub GraphQL API[1]. Automatic factors can be either directly extracted or inferred from the API, whereas manual factors necessitate manual calculation due to their qualitative nature and relative difficulty in quantification. They typically encapsulate more socio-cultural elements, such as team experience, which, despite being critical for software cost estimation, are inherently complex to derive from Github repositories.

Subsequent sections delve into the data extraction process from GitHub via the GraphQL API. This extraction process forms the cornerstone of our approach as it furnishes the raw data, which, after suitable mapping with automatic factors, constitutes our dataset. The intention is to transparently articulate the procedural steps involved in querying the requisite data via the API, in addition to outlining the subsequent mapping of this data to the corresponding automatic factors.

The concluding part of this chapter is dedicated to the creation of the dataset that will be employed for model training. It primarily examines data preprocessing, an imperative step that ensures the data is in a usable and appropriate format. A comprehensive description of the resultant dataset is subsequently presented.

This chapter bridges the theoretical understanding derived from expert interviews and the practical application of those insights in model development. The goal is to establish a transparent and replicable process that will support the validity of the findings in the later stages of this research.

5.1 Factor Classification

Identified factors from previous steps were broadly classified into Manual and Automatic categories. Manual factors cannot be directly extracted or derived using GitHub API and often depend on individual skills, experience, and human judgment. On the other hand, Automatic factors can be directly measured and quantified or at least derived using GitHub API.

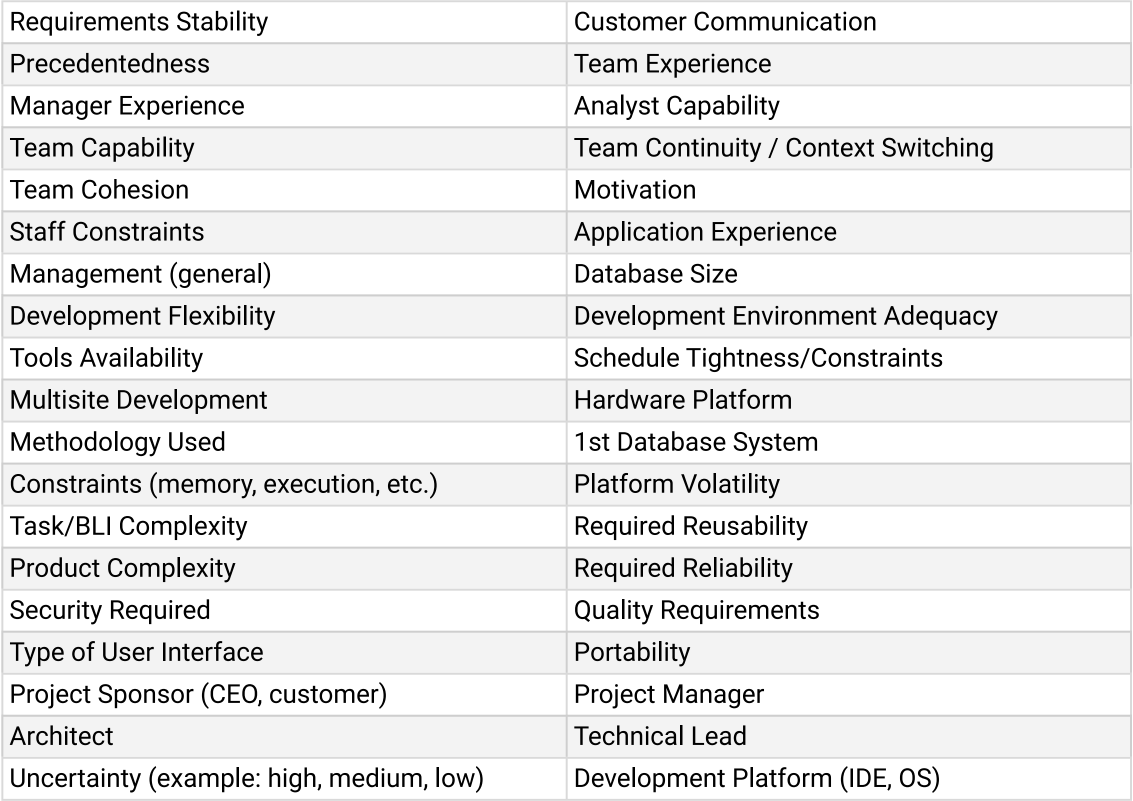
As elaborated, the factors identified in this study are based on the analysis of an extended semi-systematic literature review and expert interviews. Additionally, careful consideration of

[1] https://docs.github.com/en/graphql

what information can be realistically obtained was considered. It is important to note that this classification does not imply any hierarchy or priority among the factors.

5.1.1 Manual Factors

As discussed, manual factors are mostly qualitative and depend on human judgment and experience. These include factors related to project management, team dynamics, experience, and the working environment. Table 5.1 lists identified manual factors.
Given the nature of these factors, they were not directly included in the machine-learning model. However, they were considered during the cost estimation phase using the Constructive Cost Model (COCOMO) II (Boehm et al., 1995), which is a widely accepted model for software cost estimation that takes into account a variety of factors, including those that are harder to quantify. However, in this study, only the basic model of COCOMO was used since the priority was turned towards model development using data from GitHub repositories.



Requirements Stability	Customer Communication
Precedentedness	Team Experience
Manager Experience	Analyst Capability
Team Capability	Team Continuity / Context Switching
Team Cohesion	Motivation
Staff Constraints	Application Experience
Management (general)	Database Size
Development Flexibility	Development Environment Adequacy
Tools Availability	Schedule Tightness/Constraints
Multisite Development	Hardware Platform
Methodology Used	1st Database System
Constraints (memory, execution, etc.)	Platform Volatility
Task/BLI Complexity	Required Reusability
Product Complexity	Required Reliability
Security Required	Quality Requirements
Type of User Interface	Portability
Project Sponsor (CEO, customer)	Project Manager
Architect	Technical Lead
Uncertainty (example: high, medium, low)	Development Platform (IDE, OS)

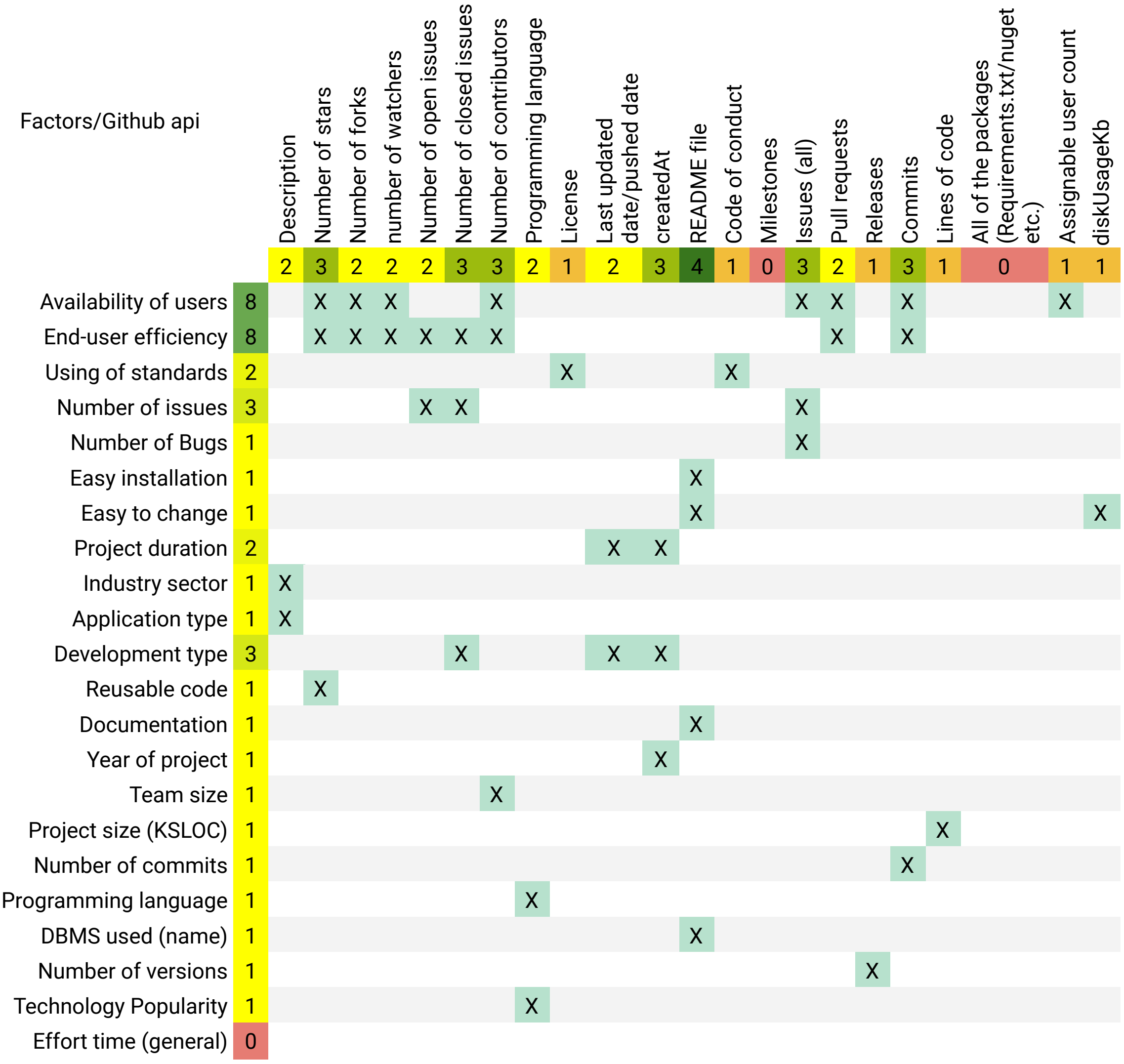
Table 5.1: Manual Factors.

5.1.2 Automatic Factors

Automatic factors are more quantitative and can be directly measured using tools like APIs. If direct measurement is not possible, NLP or other techniques can be used to derive such factors from the data that was gathered. For example, using a readme file or description of a repository to extract valuable information that could indicate for which industry the software is being developed. As elaborated, the GitHub GraphQL API was used to extract such relevant data. The automatic factors are mostly related to the actual coding activity, such as the number of commits, lines of code, issues, and contributors. Figure 5.2 provides a visualization of the automatic factors considered in this study.

Not all automatic factors were used as input for the estimation model. The chosen factors were selected based on their availability through the GitHub API and their potential impact on the software development effort. Additionally, the choice of factors that served as input for the estimation model was based on the opinion of the experts and the availability of the corresponding data at the beginning of the software development itself. During this study, the focus was turned towards the following automatic factors: Availability of users, End-user efficiency, Industry sector, Application type, Development type (enhancement/ new development), Reusable code, Documentation, Team size, Project size (KSLOC), Programming language, and DBMS used (name).

All automatic factors, along with their corresponding GitHub API feature and how they were derived, are listed in table 5.2. For example, the factor "Development type" was calculated from the following Github Api features: "Number of closed issues", "last updated/pushed date", and "created at".



Factors/Github api		Description	Number of stars	Number of forks	number of watchers	Number of open issues	Number of closed issues	Number of contributors	Programming language	License	Last updated date/pushed date	createdAt	README file	Code of conduct	Milestones	Issues (all)	Pull requests	Releases	Commits	Lines of code	All of the packages (Requirements.txt/nuget etc.)	Assignable user count	diskUsageKb
		2	3	2	2	2	3	3	2	1	2	3	4	1	0	3	2	1	3	1	0	1	1
Availability of users	8		X	X	X			X								X	X		X			X	
End-user efficiency	8		X	X	X	X	X	X									X		X				
Using of standards	2									X				X									
Number of issues	3					X	X									X							
Number of Bugs	1															X							
Easy installation	1												X										
Easy to change	1												X										X
Project duration	2										X	X											
Industry sector	1	X																					
Application type	1	X																					
Development type	3						X				X	X											
Reusable code	1		X																				
Documentation	1												X										
Year of project	1											X											
Team size	1							X															
Project size (KSLOC)	1																			X			
Number of commits	1																		X				
Programming language	1								X														
DBMS used (name)	1												X										
Number of versions	1																	X					
Technology Popularity	1								X														
Effort time (general)	0																						

Table 5.2: Automatic factors.

5.2 Data Extraction

Data extraction is a critical process in this project as it is the foundation for the rest of the analysis. This process involves retrieving relevant data from the chosen data source, in this case, GitHub. GitHub is an online platform that hosts software development and version control using Git. It is a rich and main data source for our project, offering a variety of metrics that can be used as automatic factors in our cost estimation model.

The data extraction process was performed using the Python programming language, known for its extensive libraries and tools that facilitate data extraction and manipulation. A vital component of this data extraction process was using the GitHub GraphQL API. This API provides a flexible and efficient approach to data extraction, allowing for precise queries that return only the data fields required for the analysis. This helps reduce the volume of data transferred. An important thing to note is that a (Elmers, 2023) Github repository and its corresponding code served as the baseline from where this project started with the data extraction process.

For the data extraction, it was necessary to have a GitHub key. This key is required to authenticate our data requests with the GitHub GraphQL API. The data obtained from the API is in JSON format (Peng, Cao, & Xu, 2011). JSON is a format commonly used in computer science for data sharing or storing, and it is easy to read and write. The data extraction process was focused on the most recent data within the last year, from January 1, 2022, to the present day (10th of April, 2023). The decision was made with experts' opinions since they argued that software development had changed significantly in the last 10 years. Therefore, older repositories could indicate different processes and thus render unexpected results. Additionally, 5000 repositories were marked as sufficient for effort estimation. However, this number can be increased if further research in this area is necessary. The choice of 5000 repositories was based on the desire to create a robust and representative dataset that would produce statistically significant results. A larger dataset ensures a wider variety of projects, enhancing the study's generalizability. While a larger dataset might provide additional insights, including 5000 repositories also considered the feasibility and computational resources required for data scraping. Therefore, the choice was made to balance data variety, statistical significance, and feasibility.

5.2.1 Utilizing the GitHub GraphQL API

The GitHub GraphQL API provides a robust and flexible platform for data extraction from GitHub. The use of GraphQL offers several advantages over traditional RESTful APIs. It allows for more precise and complex queries, leading to a more efficient data extraction process, as you only request the specific data fields you need.

However, the GitHub GraphQL API comes with certain limitations, such as a restriction on the number of repositories that can be retrieved with a single query. To overcome this, a bisection method was used to divide the range of stars and dates into smaller regions, each containing less than or equal to 1000 repositories. This ensured that the data extraction process adhered to the API's restrictions while still retrieving the maximum amount of data possible. Moreover, only 100 repositories can be extracted in one request. Therefore pagination option was used to extract all repositories that fell into the range of one query. In this project's scope, we have encountered frequent errors with requests that included a larger set of repositories. Therefore, the page size limit was set to 24 repositories at once.

The process of extracting data was saved using Pickle[2], a Python module used for serializing

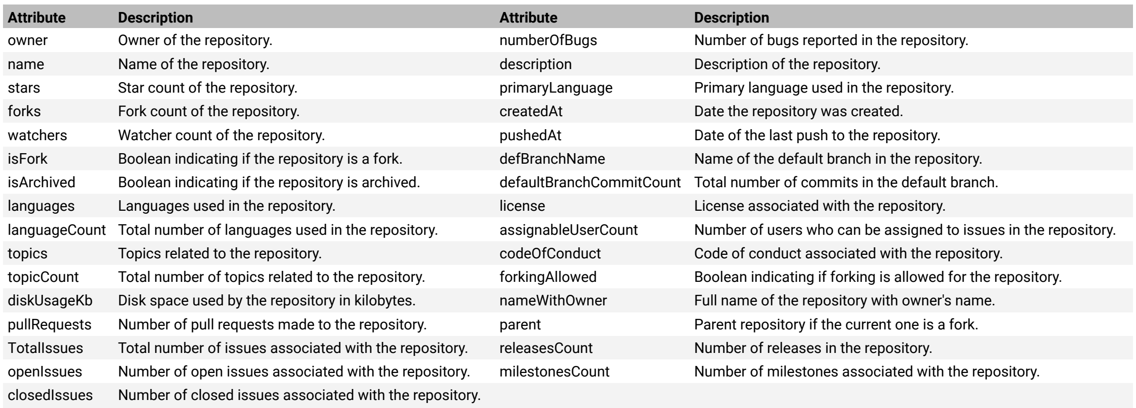
[2] https://docs.python.org/3/library/pickle.html

and de-serializing Python object structures. This allows for easy extraction in case something goes wrong during the process, which is crucial given the large volume of data involved.

In addition to the GitHub GraphQL API, data were extracted using other methods. The CodeTabs API[3] was used to retrieve the number of lines of code for each repository, while the raw.githubusercontent.com[4] URL was used to download the README file for each repository in text format. The traditional GitHub API was also used to retrieve each repository's total number of contributors. The PyGithub[5] library was used to extract this information, which simplifies the process of interacting with the GitHub API.

In total, 5699 repositories were extracted from the API, which slightly surpasses the initially planned threshold of 5000 repositories. The reason for this number being above 5000, a threshold lies in the bisection of the range and stars. This means that the last API request began when the final count was close to our threshold of 5000 repositories and returned a set of repositories that pushed the total count to 5699 repositories. Given that a larger dataset generally provides a broader basis for analysis and there is no inherent disadvantage in having more data, the decision was made to include all retrieved repositories in the final dataset. The complete list of extracted attributes is depicted in table 5.3.

Overall, the data extraction process was a complex but essential stage in developing the cost estimation model. It required a careful selection of data sources and extraction methods. Additionally, it presented a time-consuming operation in this research since extracting the data takes a long time, especially with complications that occur during testing where ensuring everything works as it should demands a lot of patience and planning. However, the result was a comprehensive dataset rich in the metrics necessary for the subsequent phases of the project.



Attribute	Description	Attribute	Description
owner	Owner of the repository.	numberOfBugs	Number of bugs reported in the repository.
name	Name of the repository.	description	Description of the repository.
stars	Star count of the repository.	primaryLanguage	Primary language used in the repository.
forks	Fork count of the repository.	createdAt	Date the repository was created.
watchers	Watcher count of the repository.	pushedAt	Date of the last push to the repository.
isFork	Boolean indicating if the repository is a fork.	defBranchName	Name of the default branch in the repository.
isArchived	Boolean indicating if the repository is archived.	defaultBranchCommitCount	Total number of commits in the default branch.
languages	Languages used in the repository.	license	License associated with the repository.
languageCount	Total number of languages used in the repository.	assignableUserCount	Number of users who can be assigned to issues in the repository.
topics	Topics related to the repository.	codeOfConduct	Code of conduct associated with the repository.
topicCount	Total number of topics related to the repository.	forkingAllowed	Boolean indicating if forking is allowed for the repository.
diskUsageKb	Disk space used by the repository in kilobytes.	nameWithOwner	Full name of the repository with owner's name.
pullRequests	Number of pull requests made to the repository.	parent	Parent repository if the current one is a fork.
TotalIssues	Total number of issues associated with the repository.	releasesCount	Number of releases in the repository.
openIssues	Number of open issues associated with the repository.	milestonesCount	Number of milestones associated with the repository.
closedIssues	Number of closed issues associated with the repository.		

Table 5.3: GitHub GraphQL API extracted attributes.

5.2.2 Mapping Data with Automatic Factors

Factors classified as retrievable from GitHub GraphQL API were then mapped with data extracted from the API. Which Github metadata feature was essential for which factor can be visible in the table 5.2. Some factors required no mapping, for example, the number of bugs, issues, and year of the project; the rest were derived from various Github metadata

[3] https://api.codetabs.com/v1/loc/?github

[4] https://raw.githubusercontent.com/

[5] https://pypi.org/project/PyGithub/

features. This subsection will mainly cover the factors chosen for the final estimation model since they can be determined before the start of the project, or they represent a higher impact on software cost/effort estimation. Moreover, the next section will describe factors calculated using statistical operations.

The "Industry sector" and "Application type" of a repository were determined using NLP or, more precisely, Spacys' classy classification[6] which provides a pre-trained model for classification. The initial input for the model is a specific dictionary with labels and their corresponding examples. After that, the model can classify any given text with these labels and also provide a likelihood score for such classification. Classification of the mentioned factors was conducted using a description of every repository. The initial plan was to include a Readme file; however, classification accuracy decreased since Readme files usually include plenty of technical information.

The Global Industry Classification Standard GICS list of industries was used to configure the model to classify each repository into the "Industry sector" factor. The GICS list contains eleven industries: Energy, Materials, Industrials, Consumer Discretionary, Consumer Staples, Health Care, Financials, Information Technology, Communication Services, Utilities, and Real Estate. The main goal was to determine which industry sector a given repository belongs to. The industry's name represents a label served to the model. Additionally, a description of an industry and its main keywords were provided for every label to improve classification accuracy. Additionally, when comparing the dataset descriptions, the Information Technology (IT) industry had a higher frequency. Therefore, if a result of classification was tied to IT or if the score of the highest classification was not above 0.3, then IT was chosen as the industry for that repository.

Each repository was classified into Application type or Software type using the same Spacy's classy classification. The taxonomy of software types proposed by (Forward & Lethbridge, 2008) was used as the list of labels for the model. Importantly, only the main types were used in this research: Data-dominant software, Systems software, Control-dominant software, and Computation-dominant software.

Development type (enhancement or new development) was determined as "new development" if the project duration of a repository was younger than 100 days. Otherwise, it was marked as "enhancement". The documentation size was calculated using a simple arithmetic approach by counting the number of chapters in the Readme File. Database management system (DBMS) was extracted using the (*DB-Engines Ranking*, 2023) ranking list of the most popular DBMS systems where the top 100 most popular systems were considered. The process included searching for a given DBMS name in a Readme file and considering the most frequent DBMS mentioned. Additionally, technology popularity was calculated using the same DB-Engines list plus (*TIOBE Index*, 2023) of the most popular programming languages. The calculation of the popularity score was created using the average between the ranking of the main programming language of a repository and the DBMS used. Finally, before the actual creation of the dataset, the "Using of standards" factor was calculated based on the occurrence of licenses or codes of conduct in the specific repository. In case any of them were present, using standards was set to true, otherwise false. It is important to note that the decision was made to ignore this factor for the final effort estimation because it is subject to bias since open-source projects that were used in our case usually have an open-source license; therefore, it is difficult to find any correlation with the effort of specific projects. For example, MIT License is used in 2025 repositories in our dataset, Apache license 2.0 in 858 repositories, and GNU General Public License in 514 v3.0.

[6] https://github.com/davidberenstein1957/classy-classification

5.3 Dataset Creation and Preprocessing

After the initial mapping, the dataset was created and ready to be expanded and preprocessed. Pandas data and analysis tool [7] was used to preprocess our data. It is a free and open-source tool built on the Python programming language. Additionally, all data operations were conducted in Jupyter Notebook[8] an open-source tool facilitating interactive data science computation, which enables easy and clear manipulation of data.

Firstly, the format was changed from JSON to CSV since it had a faster processing time with Pandas data frame. This was followed by removing the Readme file from the dataset since it occupied unnecessary space in the dataset and made it difficult to read and understand. Columns CreatedAt and PushedAt, which indicate the created date of the repository and the final push date (last code submission) were also converted to date format for easier data manipulation.

After the initial preprocessing of the data, the rest of the factors that required statistics were calculated. This included end-user efficiency, user availability, reusable code, and ease of use. All of the factors will be elaborated on in the following subsection. An important thing to note is that the decision to calculate specific factors was made in cooperation with experts in software development. Additionally, the process can later be adapted if future research finds additional insights on this topic.

End-user efficiency and user availability were calculated from almost the same factors, where each factor had a corresponding weight that can be found in table 5.4. The process included calculating the score for these two factors, where the value of each attribute in the dataset was multiplied by a corresponding weight and then summed together. Finally, the scores of all repositories were divided into three quantiles, and based on that score, each repository was later classified into low, medium, and high categories.

The calculation of reusable code followed the same principle where the number of stars was considered as the indicator of code reusability since it implies the popularity of each repository and the likelihood of it being downloaded and reused. The number of stars was therefore split into three quantiles; based on that; each repository was classified into low, medium, and high categories.

Factors of easy installation and easy to change were grouped into one factor called Ease of Use, which indicates how simple it is to use a specific repository. This was calculated using technology popularity and documentation size, where the emphasis was made on the documentation size. The score was again calculated where higher ease of use score indicates that the repository has a larger documentation size and lower technology popularity (lower means more popular), which suggests that it is easier to use since it has a lot of documentation available. This score was later split into three quantiles and classified into low, medium, and high for each repository.

Finally, the essential COCOMO II (Boehm et al., 1995) model was used to calculate the effort of each repository. The calculation was made based on each repository's lines of code (LOC) and included Person Months (PM) as the final effort estimation for each repository. An in-depth explanation of the basic COCOMO II model will be elaborated in the next chapter.

[7] https://pandas.pydata.org/

[8] https://jupyter.org/

Dataset attribute	End User Efficiency	User Availability
Stars	0,1	0,1
Forks	0,15	0,1
Watchers	0,05	0,05
Contributors	0,3	0,3
Closed Issues	0,05	--
Open Issues	0,05	--
Issues	--	0,15
Pull Requests	0,1	0,05
Commits	0,2	0,15
Assignable User Count	--	0,1

Table 5.4: Weights for End User Efficiency and User Availability factors.

5.3.1 Final Dataset Description

As already elaborated, the final dataset contains 5699 open-source software projects published between January 1, 2022, and the 10th of April, 2023. Additionally, only repositories that had more than 5 stars were considered. The table 5.5 indicates additional attributes mapped and derived after the data was extracted from the API. The final dataset also includes the attributes extracted from the API, which can benefit future research in this area. The complete list of extracted attributes was elaborated in subsection 5.2.1. Furthermore, the final version of the created dataset is available in (Mrvar, 2023), which includes the initial data collected from repositories and derived factors that were calculated additionally.

Dataset Attribute	Description
ProjectDurationDays	Duration of the project in days
IndustrySector	Industry sector of the project
SoftwareType	Type of software being developed
DBMS	Database Management System used in the project
DocumentationSize	Size of the documentation of the project
IsNewDevelopment	Indicator of whether the project is a new development or enhancement
UsingOfStandars	Indicator of whether the project uses standards
TechnologyPopularity	Popularity score of the technology used in the project
end_user_efficiency_score	Score indicating the efficiency of the end user
end_user_efficiency_category	Category based on the end user efficiency score
availability_of_users_score	Score indicating the availability of end users
availability_of_users_category	Category based on the availability of end users score
reusable_code	Indicator of whether the code is reusable or not
ease_of_use_score	Score indicating the ease of use of the project
ease_of_use_category	Category based on the ease of use score
Effort	Effort required to develop the project in person months
Tdev	Estimated time needed to develop the project in months

Table 5.5: Descriptions of Dataset Attributes.

Chapter 6

Software Cost Estimation Modeling

6.1 Preliminary Considerations

Before delving into the details of the final software cost estimation model, several preliminary considerations need to be discussed. These considerations pertain primarily to manual factors influencing software cost estimation and their integration within the COCOMO II model. Subsequently, we present a justification for the handling of these factors in this study. Furthermore, we delve into the applicability of COCOMO II and how the effort was calculated in this study.

6.1.1 Consideration of Manual Factors

Manual factors, as discussed in subsection 5.1.1, constitute a set of variables that lean heavily towards social elements, require expert judgment for determination, or are not directly quantifiable. Despite the challenges they pose, their critical role in software cost/effort estimation necessitates careful consideration, especially when deploying machine learning techniques.

Within the confines of this research, we categorized factors such as team capability, the methodology employed, team experience, manager experience, etc., as manual factors. These factors, derived from both the literature study and expert interviews carried out in this study (elaborated upon in the preceding chapters), exert significant influence on the software development process and, by extension, the cost of software development. For instance, a more experienced team can accelerate implementation and software development's final testing, leading to overall cost reduction.

Another salient manual factor is the methodology adopted during software development. Methodologies can range from Agile to traditional waterfall development approaches. Agile software development, for example, can foster improvements through more stable requirements, earlier fault detection, and enhanced communication, among others (Kumar & Bhatia, 2012), which can ultimately reflect positively on the final cost of software development.

Despite the focus of this study being primarily on automatic factors, owing to their feasible extraction from the GitHub API, we did not entirely discount manual factors. For the final estimation, we employed COCOMO II in conjunction with various machine-learning algorithms to provide an estimate of the final effort required for software development. The conceptual model of this study, and hence the developed estimation model, was elaborated upon in section 1.5.

6.1.2 Integration with the COCOMO II Model

The Constructive Cost Model 2 (COCOMO II), a pervasive algorithmic software cost estimation model established by Boehm et al. (Boehm et al., 1995), formulates effort as a function of program size, expressed in thousands of lines of code (KLOC).

In this research, the COCOMO II model was integrated into our cost estimation approach due to its robust methodology for estimating effort based on a repository's size, which was determined from the number of lines of code present in the repositories in our dataset. The effort adjustment factor was also acknowledged to accommodate the aforementioned manual factors influencing the development effort. It should be noted that not all manual factors discovered during the literature study and expert interviews were included, as the COCOMO II model accommodates only a select group of these factors, referred to as "Effort Adjustment Factors" (EAF). The COCOMO II cost model is represented by the following equations:

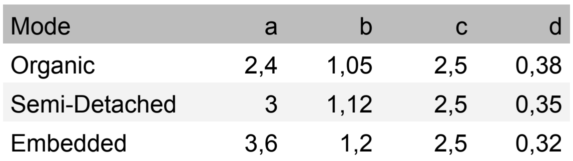
$$\text{Effort} = a \times (Size)^{b} \times EAF \tag{6.1}$$

$$\text{Tdev} = c \times (Effort)^{d} \tag{6.2}$$

Where:

- **Effort:** The cumulative effort required for software development, articulated in person-months.
- **a, b, c, and d:** Constants for calibration based on defined modes as discussed below. These constants are deduced from historical project data and vary depending on the specific software being developed.
- **Size:** Denotes the software project's size. In this research, the size was depicted in lines of code (LOC).
- **EAF:** The Effort Adjustment Factor signifies the combined effect of cost drivers on the overall project effort.
- **Tdev:** Represents the time frame for software development measured in months.

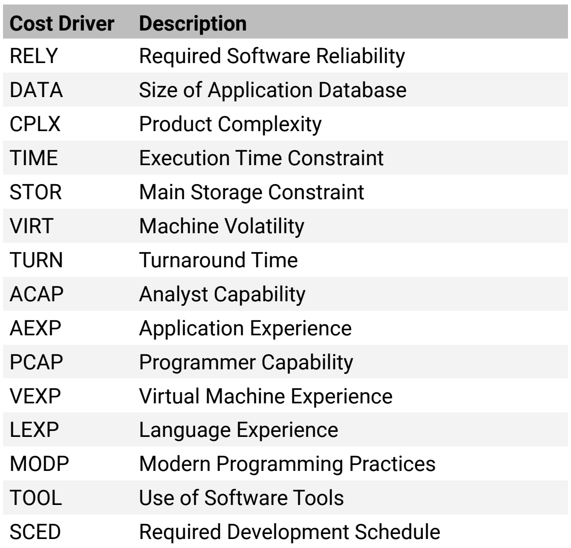
The COCOMO II model incorporates specific modes outlined in Table 6.1. The selection of modes in this study was predicated on the project size. If the project size is between 2 and 50 KLOC, the Organic mode is selected, representing a simple and small project with a relatively minor problem domain. For project sizes between 50 and 300 KLOC, the Semi-detached mode, representing medium-sized projects, is selected. Lastly, for projects exceeding 300KLOC, the Embedded mode is chosen, signifying large-scale and complex projects. Thus, the A, B, C, and D constants were determined based on these modes and utilized in the equations provided above.



Mode	a	b	c	d
Organic	2,4	1,05	2,5	0,38
Semi-Detached	3	1,12	2,5	0,35
Embedded	3,6	1,2	2,5	0,32

Table 6.1: COCOMO II parameters for each mode.

Table 6.2 displays the EAF factors incorporated into the equation. These factors embody effort adjustment factors and can be included in software cost estimation. However, during the creation of the dataset for this study, these factors were not encompassed in the final analysis, primarily due to their challenging extraction from GitHub repositories. Consequently, the EAF was uniformly set to 1



Cost Driver	Description
RELY	Required Software Reliability
DATA	Size of Application Database
CPLX	Product Complexity
TIME	Execution Time Constraint
STOR	Main Storage Constraint
VIRT	Machine Volatility
TURN	Turnaround Time
ACAP	Analyst Capability
AEXP	Application Experience
PCAP	Programmer Capability
VEXP	Virtual Machine Experience
LEXP	Language Experience
MODP	Modern Programming Practices
TOOL	Use of Software Tools
SCED	Required Development Schedule

Table 6.2: COCOMO II Effort Adjustment Factors (EAF).

6.2 Machine Learning Approaches

The central objective of this study was to employ Machine Learning (ML) techniques to locate the most akin projects from the compiled dataset, given specific input. The ultimate aim was to provide end-users with a snapshot of the time frame involved in the development of similar projects based on their requirements.

For this study, four distinct ML techniques were applied to the same dataset, including K-Nearest Neighbors (KNN), Hierarchical Clustering, Autoencoders, and Support Vector Machines (SVM). The decision on the selection of the model was informed by previous literature studies where similar strategies were employed, coupled with their popularity and demonstrated success in tasks such as clustering or identification of similarities. An important thing to note is that the objective of this study was not to pinpoint the optimal ML technique, but rather to validate the feasibility of the comprehensive process required for software cost estimation.

Generally, the input for each ML technique remained consistent. This input encompassed predetermined factors elaborated upon in the preceding chapter, which were further segregated into numerical and categorical variables. The numerical input factors included **DocumentationSize**, **contributors_count**, and **LOC**. These factors represent quantifiable facets of a software project, where Documentation size is gauged by the number of chapters in the Readme file, contributors count denotes the number of contributors engaged in a project, and LOC symbolizes lines of code. Conversely, the categorical input factors incorporated were **availability_of_users_category**, **end_user_efficiency_category**, **ease_of_use_category**,

IndustrySector, **SoftwareType**, **primaryLanguage**, **DBMS**, and **reusable_code**. All these factors served as input to an ML model, aiming to return the ten most similar projects in our dataset, along with the effort/time required to develop these projects. It's vital to note that LOC was excluded during ML clustering as it's challenging to ascertain this value realistically prior to the project's commencement.

With the application of these four machine learning techniques, this study was able to gain valuable insights from the dataset to assist in software cost estimation. Each approach provided a different perspective on project similarity and development effort. Moreover, each technique yielded different repositories, thus facilitating easier comparison and offering a range of choices for the approach to employ in software cost estimation. The results are anticipated to bridge the gap between project requirements and execution, yielding a more efficacious and efficient estimation model.

The subsequent subchapters briefly delve into these four ML techniques utilized, focusing on their specifics and underlying scientific principles.

6.2.1 K-Nearest Neighbors (KNN)

K-Nearest Neighbors (KNN) is a non-parametric, supervised learning method known for its simplicity in classification. KNN operates on the concept that patterns nearest to a target pattern x, which requires labeling, deliver beneficial label information (Kramer & Kramer, 2013). The KNN method is generally used for classification or regression problems, but within the scope of this study, it was utilized as an instance-based learning or similarity search, instead of classification or regression. This means that the algorithm identifies the nearest neighbors in the feature space without predicting specific classes or numerical values.

In this project, KNN was utilized to locate the most similar projects in our dataset. The variable number of neighbors used for majority voting in identifying the closest instances was set to 10.

6.2.2 Hierarchical Clustering

Hierarchical Clustering (HCA) is an extensively used unsupervised machine learning algorithm that begins by treating each object as a singleton cluster and subsequently merges pairs of clusters until one large cluster comprising all objects from the dataset is formed. This result is depicted in a tree-like representation known as a dendrogram. This process is referred to as an agglomerative or bottom-up approach, as opposed to a partitional or top-down approach, which can also be used to obtain hierarchical clustering using repeated bisections (Zhao, Karypis, & Fayyad, 2005). Hierarchical clustering is a widely used approach in software effort estimation, with some studies demonstrating that analysis can significantly improve when using agglomerative hierarchical clustering (Hai, Nhung, & Jasek, 2022).

In this study, Ward's hierarchical agglomerative clustering method was used for clustering, which aims to minimize the variance between clusters. The Euclidian metric was employed to calculate the distance between samples.

6.2.3 Autoencoder

Autoencoders are a type of neural network designed to encode input into a compressed, meaningful representation and then decode it to reconstruct information that closely resembles the original (Bank, Koenigstein, & Giryes, 2020). They belong to the unsupervised learning domain and primarily use two functions: encoding and decoding. Autoencoders have proven effective in classification tasks and are used to solve various problems, including text and image classification (P. Zhou, Han, Cheng, & Zhang, 2019).

6.2.4 SVM

Support Vector Machines (SVM) were developed by (Cortes & Vapnik, 1995) for binary classification. SVM represents a supervised machine learning algorithm that, similarly to KNN, can be used for classification or regression problems. However, it is primarily used for classification challenges.

The algorithm seeks the optimal separating hyperplane between two classes by maximizing the margin between the closest points of these classes. The points on the boundaries are known as support vectors, and the midpoint of the margin is called the optimal separating hyperplane. Furthermore, data points on the wrong side of the margin are downweighted to reduce their influence, a step known as "soft margin". Subsequently, kernel techniques are used to project data points, typically into a higher-dimensional space, if a linear separator cannot be found. This task can be formulated as a quadratic optimization problem that known techniques can solve. The problem capable of performing all these tasks is called a Support Vector Machine (Meyer & Wien, 2015).

Multiple extensions of SVM were developed, including v-classification, One-class classification, and multi-class classification. This study uses One-class classification with a linear kernel, enabling outlier or novelty detection. It establishes a boundary representing standard data, then compares how far each data point is from this boundary.

6.3 Construction of the Estimation Model

The model for software effort estimation developed in this research is roughly illustrated in Figure 6.1. Although it echoes the conceptual model outlined in Chapter 1, this graphical depiction explains the actual estimation process.

The model's initial part encompasses a data collection process, thoroughly detailed in Chapter 5. It involves data extraction from GitHub, data mapping to metadata attributes and KPIs, and the application of COCOMO II for estimating each repository's effort. Furthermore, all the necessary automatic factors need to be calculated from the obtained data. Using this information, the final dataset was generated with all the required details to commence the effort estimation process.

Once the final dataset is compiled, the estimation can begin, utilizing software project specifications that can be supplied as manual or automatic factors discussed in Chapter 5. It's important to note that the estimation model doesn't require human input/manual factors since they aren't incorporated into the dataset. Hence, similarities can't be computed between the provided information and the actual repositories in the dataset. Consequently, the estimation model only considers automatic factors representing the functional requirements of the intended software product. Nevertheless, the manual factors are still factored in using the COCOMO II and Lines of Code (LOC) estimation offered by the user conducting the analysis.

Upon supplying the automatic factors, the cost estimation model can search for similar projects in the dataset based on these factors. As previously stated, this process employed four different ML techniques: KNN, Hierarchical clustering, Autoencoder, and SVM. The effectiveness of each model will be demonstrated later in Chapter 7.

The cost estimation model's final output is the ten most similar projects identified by the ML technique. These ten projects are then evaluated for outliers using the Interquartile Range

(IQR) method for outlier detection (Vinutha, Poornima, & Sagar, 2018), based on the effort reported in person months by each project. This strategy was deemed effective as some returned repositories during the testing phase were deemed anomalies. Hence, the decision to normalize the results produced by the ML techniques was made. IQR is a technique that aids in identifying outliers in continuously distributed data, calculated using the following equation:

$$\text{IQR} = Q3 - Q1 \tag{6.3}$$

Where: Q1 represents the first quartile, and Q3 signifies the third quartile, which is recursively calculated using the median value. After determining the IQR, Q1, and Q3, the boundaries for outliers were established as:

$$\text{Lower boundary} = Q1 - (1.5 \times IQR) \tag{6.4}$$
$$\text{Upper boundary} = Q3 + (1.5 \times IQR) \tag{6.5}$$

Subsequently, all values beyond these boundaries were discarded based on the returned effort in person months.

Once the outliers were excluded, the final estimation was determined by averaging all remaining repositories in the list of data returned by the ML model. The final output variables are **Effort**, indicating the effort in person-months, and **Tdev**, representing the time needed for software development. As previously stated, both values reflect the average of all repositories identified as most similar to the data supplied to the model.

The final estimation model and the complete code utilized in this process can be found in (Mrvar, 2023).

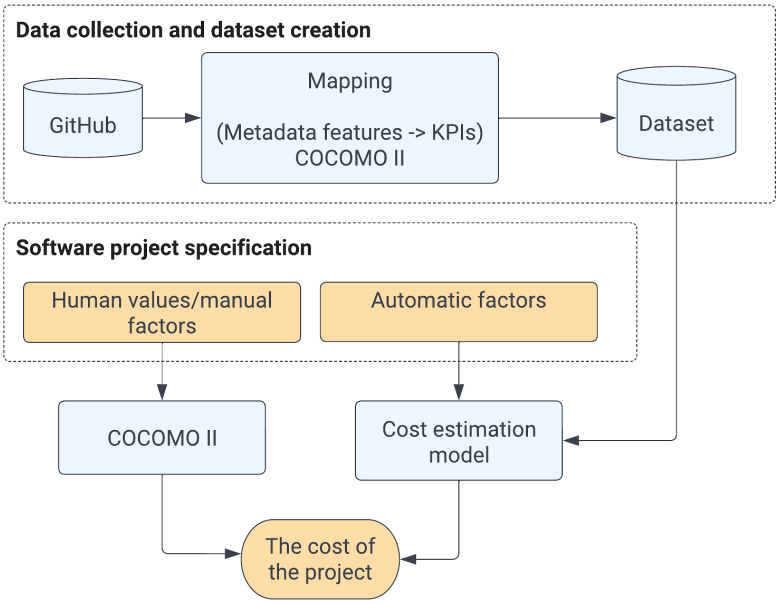

Data collection and dataset creation
GitHub
Mapping
(Metadata features -> KPIs)
COCOMO II
Dataset
Software project specification
Human values/manual factors
Automatic factors
COCOMO II
Cost estimation model
The cost of the project

Figure 6.1: The cost estimation process.

Chapter 7

Empirical Evidence

To substantiate the efficiency and applicability of the proposed cost estimation model, two evaluation methods were deployed. The primary approach employed a case study conducted in tandem with a small-scale IT firm based in Slovenia. As a secondary approach, an experimental design was utilized, testing the estimation model against a set of randomly selected projects listed on Freelancer.com[1].

The comprehensive aim of these evaluations was to assess the proposed cost estimation model in practical, real-world contexts. This methodology is intended to measure the success rate, applicability, and generalizability of the model across a diverse range of projects.

This chapter outlines the conduct of the experiments, presents the resultant data and provides an in-depth discussion of the findings.

7.1 Case study

As elaborated in Chapter 2, a case study is acknowledged as an empirical research method designed to investigate a phenomenon within its real-life context (Yin, 1981). The guidelines proposed by Yin (Yin, 1981) were strictly adhered to in both the planning and conduct of the case studies for this research.

Objective: The central goal of this research, and consequently the case study, was the construction and validation of the estimation model for software project cost estimation.

The case: The case analysis centered on a small IT company based in Slovenia.

Methods: Data collection encompassed multiple interviews and sessions to understand the specifics of past software projects, their requirements, and their ultimate effort.

Selection strategy: A multiple case study approach (Yin, 1981) was utilized in this research to facilitate data analysis. Examining three software projects provided a comprehensive understanding of individual cases and facilitated extensive exploration of the research questions.

Theory: The formulated estimation model can serve as a valid reference model to aid stakeholders in cost estimation for software development processes.

Protocol: To appraise the estimation model and execute the case study, the following protocol was followed

1. **Project Data Elicitation:** Participants provided details about their past projects, including requirements and development efforts.

2. **Implementation of ML Models:** The identified requirements were analyzed, and the proposed ML models were applied to identify similar projects in our knowledge base and their corresponding effort.

[1] https://www.freelancer.com/

3. **Results:** The results were gathered and averaged as detailed in chapter 6.

4. **Discussion and Analysis:** The final results were examined and presented to stakeholders for discussion.

In addition, the standards set by the ACM SIGSOFT for empirical research (Ralph et al., 2020) were followed to ensure the rigor and relevance of the case study.

The case study undertaken in this research deployed real-life industry data from a small tech enterprise based in Slovenia. This company's core expertise lies in digitalizing business processes for its clientele, offering various services, including custom web application development, contemporary web page design, and mobile and desktop application development. Furthermore, the company provides consulting services to its clients.

At the time of this research, the company had 6 employees, primarily specializing in software development utilizing .NET technologies. The company's primary focus was on digitalizing industrial processes, a specialization that has attracted a significant client base from the industrial sector.

7.1.1 Project Requirements

The principal objective of data elicitation was to collect extensive, relevant data pertaining to past software projects conducted within the organization. This process can pose challenges, particularly given the limited time availability of stakeholders, necessitating advanced planning and organization. Privacy concerns further complicated the process, limiting the discussion of specific project details, including pricing, individuals involved, client details, and specific features and requirements.

Identifying the Data Source

Three past projects executed by the organization were discussed and presented. The process and methodology of data collection will be detailed in the subsequent chapter.

The three projects presented by the organization all held similarities and were developed for clients from the industrial sector. The principal aim for these clients was to digitize labor-intensive or time-consuming business processes.

Each project was a web application, necessitating the company to commence development largely from scratch, with no prior projects to base their code upon. However, certain components from previously developed software were incorporated into the development, including various forms that enabled CRUD (Create, Read, Update, Delete) operations and other useful elements. Given the company's expertise in .Net (C#) technologies, all projects were built using this technology.

As mentioned, the specifics of each project were avoided to respect privacy concerns. Nonetheless, efforts were made to collect all relevant information that could be utilized in evaluating the proposed estimation model. The complete data collection process and what was collected will be presented in the next section.

Data Collection Process

Multiple interviews were conducted to gather pertinent information about the software projects under investigation for this study. Additional communication, especially when further clarification was required from the participant, was conducted remotely via email or calls.

The majority of the communication was held directly with the CEO of the company, given their comprehensive understanding of the development process. However, detailed explanations of certain development aspects were discussed with developers as needed.

The core aspect of these discussions was to collect extensive information about their past projects, including the time required for their development. As previously mentioned, any data that could reveal private information was omitted from the discussions. The conversations also facilitated a broader understanding of cost estimation, introducing new perspectives to this study and the application of the proposed estimation model. The participant's reactions to the work proposed in this study will be detailed in the discussion section.

Upon the collection and analysis of information, the collected data were matched to the attributes utilized in this study, as presented in chapter 6. The description of the collected projects and their final mapping is shown in Table 7.1. In this table, each column represents a specific project, indexed under a unique project ID. Additionally, it includes the majority of relevant attributes used by the models to provide the effort estimation based on past projects, such as DBMS, programming language, documentation size, number of contributors, etc. The most crucial piece of data, however, is the final duration depicted in months, indicating the company's development timeline for each specific project. This data provided a basis for a clear comparison and evaluation of the proposed estimation model. Upon initial data collection and aggregation, the ML models described in chapter 6 were employed to render the final estimation. Consequently, the data was housed in a Google Sheet document and subsequently exported into a CSV file. This procedure facilitated the efficient extraction of relevant data, which served as the input for the estimation model.

In alignment with the proposed approach, ML techniques were utilized to identify the most similar repositories within the constructed dataset. The outcomes of the estimation process are presented in the subsequent section.

Attributes	ID: 1	ID: 2	ID: 3
availability_of_users_category	Medium	High	High
end_user_efficiency_category	Medium	Medium	High
ease_of_use_category	Medium	Medium	Medium
IndustrySector	INDUSTRIALS	INDUSTRIALS	INDUSTRIALS
SoftwareType	Systems software	Systems software	Systems software
IsNewDevelopment	TRUE	TRUE	TRUE
reusable_code	Medium	Medium	High
DocumentationSize	10	10	10
contributors_count	3	1	2
primaryLanguage	C#	C#	C#
DBMS	MongoDB	SQL	SQL
Actual duration (months)	36	3	8

Table 7.1: Industry-based projects: attributes and actual durations.

7.1.2 Results

As previously described, four ML approaches were utilized to perform effort estimation based on the most similar repositories within the dataset created for this study. Table 7.2 displays the outcome of each ML model for each collected project. The results in the table present each project by ID (refer to table 7.1) and for each project, its corresponding ML method is indicated. Values are represented in person-months (first row of each model) and the time required to develop the software in months (second row of each model). As demonstrated in table 7.1, the project with ID 1 required 36 months for completion, the project with ID 2 required 3 months, and the project with ID 3 required 8 months. The interpretation of the results provided will be addressed in the following chapter. Nevertheless, it is noticeable that the autoencoder was the most accurate compared to the other approaches employed. Meanwhile, the SVM returned identical repositories, which can be attributed to the similarities between the actual projects.

ID	Method	Autoencoder	KNN	Hierarchical Clustering	SVM
1	Average Effort (pm)	17,44	61,63	12,25	163,88
	Average Tdev (m)	6,33	8	5,5	11,5
2	Average Effort (pm)	4,44	31,63	464,5	163,88
	Average Tdev (m)	3,78	7,5	15,5	11,5
3	Average Effort (pm)	34,56	38,33	316,33	163,88
	Average Tdev (m)	8,67	8,44	14,67	11,5

Table 7.2: Industry-based case study results.

7.1.3 Analysis

As evidenced in table 7.2, the final results exhibit variance between each model and project. The closest estimation to Project 1 was yielded by SVM, although it significantly diverges from the actual duration, which was 36 months. The most accurate estimation for Project 2 was produced by the Autoencoder, which forecasted a development time of 3.78 months, whereas the actual time required was 3 months. Regarding Project 3, both Autoencoder and KNN provided comparable estimations of 8.67 and 8.55 respectively.

A pivotal point of consideration is that the organization rounded the total duration to whole months, suggesting that the actual duration might deviate slightly from the values utilized in this study.

Upon presentation of the results to the organization, the consensus was that such a model could potentially augment their development process, as well as those of other organizations. However, a stakeholder raised concerns regarding the substantial time and effort required to develop such a tool with more reliable results, which would adequately cover most aspects necessary for cost estimation, especially when taking into account data collection and development processes.

Regardless, the results demonstrated that the approach used in this study for cost estimation could serve as a promising point of departure for future research in this field.

7.2 Experiment

In addition to the case study, we have conducted another experimental assessment using publicly available data via the Freelancer.com platform.

Freelancer.com is a platform where freelancers can find job opportunities of various sizes and scopes, spanning a range of disciplines from writing and legal services to software development. When a job is posted, freelancers can place bids indicating their expected compensation for completing the task. Subsequently, the job poster can select a freelancer whom they deem credible and dependable.

Analogous to the case study performed, the primary objective of this experiment was to evaluate the proposed estimation model using real-world data.

7.2.1 Data Collection

In this investigation, multiple job postings on Freelancer.com were collected and mapped using factors identified in our research. Job postings were selected if they pertained to software development and featured an average bidding price exceeding $1000.

Upon implementing this pricing criterion, all projects falling beneath the specified cost were systematically excluded. Furthermore, the considered projects were necessitated to encompass comprehensive descriptions enabling the extraction of valuable factors. Moreover, we solely included projects pertaining exclusively to software development, which exploited technologies concurrent with our dataset. Given the disparate nature of project descriptions available on the platform, the selection process necessitated intricate filtering and manual curation of relevant data. This procedure culminated in the identification of 35 suitable projects.

When the projects were extracted, all relevant factors were cataloged in a spreadsheet, which included details such as the language, DBMS, software type, industry sector, and so on. Subsequently, the average bidding price for the listed job was also recorded.

Post the compilation of all pertinent factors and the estimated price, the average salary per country was extrapolated from the Glassdoor.com platform[2]. This data was leveraged to compute the average hourly wage for each country, thereby enabling us to estimate the development time for specific tasks premised on the price and hourly wage. The final effort was delineated in months for each project, premising on the assumption of 160 hours per month as a full-time project duration. The average monthly salary per country was then divided by 160 to obtain the hourly wage.

An additional critical consideration is the significant demographic of visitors to the Freelancer.com platform; data indicate a predominance of users from India (15,78%), Bangladesh (11,66%), followed by the USA (11,07%), Pakistan (6,47%), and Egypt (3,09%) (Similarweb, 2023). As a result, the majority of freelancers are primarily situated in India, Bangladesh, and neighboring countries exhibiting lower per capita income. Consequently, pricing standards on the platform are adjusted to accommodate these economic circumstances, often resulting in rates that are comparatively lower than those expected within the task's originating country. Therefore, the effort calculation factored in not only the average hourly wage of the country where the project was posted (Country of Origin) but also the hourly wages from the most represented countries on the platform: India, Pakistan, USA, and Bangladesh. Consequently, the final dataset encompassed five effort columns: one delineating the effort assuming development within the project's originating country and four others representing the effort assuming development in India, Pakistan, USA, and Bangladesh, respectively.

[2] https://www.glassdoor.com

Once all the required factors were collected, the data was extracted into a CSV document, similar to the approach taken in the previous case study. This data served as the input for the machine-learning techniques elaborated upon in the preceding chapter. These ML techniques were then employed to estimate the effort for each project. The estimated effort was subsequently compared to the effort calculated from the average bidding price, as discussed in the previous section.

7.2.2 Results and Evaluation

Table 7.3 encapsulates the conclusive results yielded by the machine learning (ML) models. This table encompasses data from 35 projects, which includes details such as the originating country of each project, the associated cost, and the calculated effort as explicated in the preceding section. The final quartet of columns delineate the estimations derived from each ML algorithm, where the effort is expressed in the time required for specific software development.

An examination of the first project, posted in Saudi Arabia with an average bidding cost of $4369, allows us to observe the calculated effort if developed in each corresponding country: the country of origin (Saudi Arabia), India, Bangladesh, USA, and Pakistan. The method for calculating the effort has been elaborated upon in the previous section. The final four columns indicate the estimations provided by each ML model. The Autoencoder model predicts an effort of 8,71 months, the KNN model estimates 4,3 months, Hierarchical Clustering forecasts 6,24 months, and the SVM model predicts 8,09 months.

From these results, we note that the Hierarchical Clustering model provides the closest estimation to the actual effort if the project was developed in Pakistan, and the KNN model performs similarly if the project was developed in India. Both models would provide close approximations if the project were developed in Bangladesh, with the Hierarchical Clustering model slightly overestimating and the KNN model slightly underestimating the actual effort.

The experimental outcomes were evaluated employing three of the most widely used evaluation methods identified in the Systematic Literature Review (SLR), detailed in chapter 3. These evaluative methods comprise the Mean Magnitude of Relative Error (MMRE), which is characterized as the mean of the absolute relative errors (Venkataiah, Mohanty, & Nagaratna, 2019), the Mean Absolute Error (MAE), defined as the mean of the absolute disparities between the predicted and actual values (Rai, Kumar, & Verma, 2020), and Pred(25%), defined as the ratio of estimated values that lie within 25% proximity to the actual values (Qi et al., 2017). The MMRE and MAE were computed employing the subsequent equations:

$$\text{MMRE} = \frac{1}{n}\sum_{i=1}^{n}\frac{\hat{E}_i - E_i}{E_i} \tag{7.1}$$

where n represents the number of samples, $\hat{E}_i$ denotes the predicted effort for the i-th instance or project, and E_i signifies the actual effort for the i-th instance or project.

$$\text{MAE} = \frac{1}{n}\sum_{j=1}^{n}|y_i - \bar{y}_t| \tag{7.2}$$

where n denotes the number of samples, y_i signifies the actual values, and $\bar{y}_i$ denotes the predicted values.

The table 7.4 presents the outcomes of the performed experiment for each country, complete with MMRE, PRED, and MAE values. Lower MMRE and MAE indicate greater accuracy,

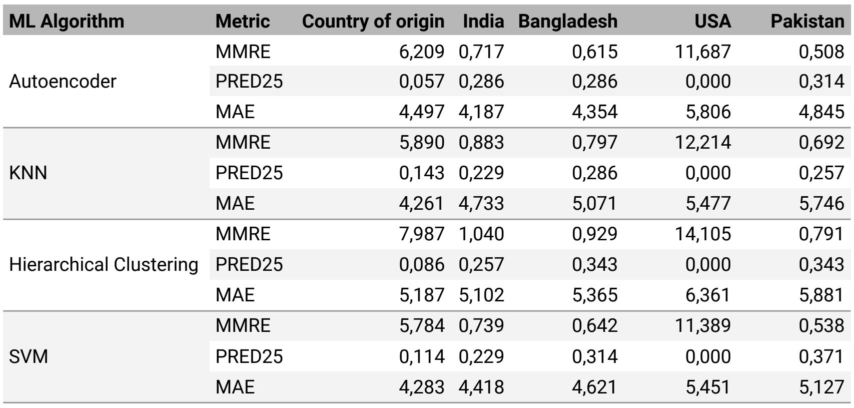
Proj.	Country	Cost ($)	Time/Effort (Country of Origin)	Time/Effort (India)	Time/Effort (Bangladesh)	Time/Effort (USA)	Time/Effort (Pakistan)	Autoencoder	KNN	Hierarchical Clustering	SVM
1	Saudi Arabia	4369	0,59	4,80	5,30	0,55	6,18	8,71	4,3	6,24	8,09
2	Turkey	3703	1,74	4,07	4,49	0,47	5,24	5,21	8,25	9,8	5,01
3	Australia	4123	0,62	4,53	5,00	0,52	5,83	6,51	3,86	7,59	6,82
4	Indonesia	2626	1,50	2,88	3,19	0,33	3,71	5,54	12,08	3,9	7,15
5	India	2.536,69	2,79	2,79	3,08	0,32	3,59	6,1	6,48	8,87	3,04
6	India	1.819,67	2,00	2,00	2,21	0,23	2,57	4,63	10,29	9,8	7,5
7	Saudi Arabia	1500	0,20	1,65	1,82	0,19	2,12	7,96	10,29	9,8	5,96
8	China	33956	7,87	37,30	41,21	4,27	48,01	5,61	6,18	7,59	5,72
9	UK	13879	2,88	15,24	16,84	1,74	19,63	5,44	6,47	7,74	5,71
10	China	14020	3,25	15,40	17,01	1,76	19,82	4,49	6,47	7,74	5,71
11	Colombia	11398	4,77	12,52	13,83	1,43	16,12	10,78	6,33	5,29	4,38
12	Italy	8417	2,97	9,25	10,21	1,06	11,90	5,2	2,27	4,47	5,54
13	Reunion	8395	2,40	9,22	10,19	1,06	11,87	7,37	6,73	4,49	5,3
14	India	6861	7,54	7,54	8,33	0,86	9,70	7,19	6,73	4,49	5,3
15	UK	5.084,87	1,05	5,59	6,17	0,64	7,19	6,56	2,85	7,59	7,18
16	UK	4046	0,84	4,44	4,91	0,51	5,72	5,35	2,89	4,47	8,14
17	USA	5045	0,63	5,54	6,12	0,63	7,13	7,04	3,59	7,86	7,17
18	India	4.792,83	5,27	5,26	5,82	0,60	6,78	8,5	5,03	7,72	5,68
19	Denmark	4669	0,42	5,13	5,67	0,59	6,60	7,73	4,96	4,89	7,36
20	Saudi Arabia	4350	0,59	4,78	5,28	0,55	6,15	5,3	5,7	8,48	4,31
21	India	4.325,06	4,75	4,75	5,25	0,54	6,12	5,29	6,27	4,32	4,46
22	Kenya	4177	2,01	4,59	5,07	0,53	5,91	9,35	5,03	7,72	5,68
23	USA	3972	0,50	4,36	4,82	0,50	5,62	4,76	6,45	5,29	5,03
24	Malaysia	3902	2,05	4,29	4,74	0,49	5,52	8,35	6,17	6,24	6,13
25	Egypt	3917	4,65	4,30	4,75	0,49	5,54	6,79	6,53	3,44	6,66
26	USA	3183	0,40	3,50	3,86	0,40	4,50	6,26	6,27	4,32	4,46
27	UK	2.855,47	0,59	3,14	3,47	0,36	4,04	8,09	6,53	3,44	6,66
28	UK	20.119,76	4,17	22,10	24,42	2,53	28,45	10,32	3,37	5,8	9,3
29	USA	7901	0,99	8,68	9,59	0,99	11,17	2,51	4,49	4,49	2,24
30	United Arab Emirates	7554	1,26	8,30	9,17	0,95	10,68	6,61	8,48	7,98	6,13
31	Argentina	5085	2,03	5,59	6,17	0,64	7,19	9,49	14,78	15,47	12,14
32	Peru	4494	2,34	4,94	5,45	0,56	6,35	4,26	6,58	6,58	5,89
33	Ireland	2400	0,41	2,64	2,91	0,30	3,39	6,58	6,02	2,61	10,16
34	Egypt	2401	2,85	2,64	2,91	0,30	3,40	3,9	4,51	7,04	7,41
35	USA	2301	0,29	2,53	2,79	0,29	3,25	7,61	6,65	27,24	5,54

Table 7.3: Results of the Experiment

while a higher PRED(25%) suggests that a larger percentage of the prediction will fall within 25% of the actual effort.

The experimental results are contrasted against the 'effort' value calculated for each respective country. This benchmark was established by considering the average wage for software developers within the specific nation in question. For instance, in the row corresponding to 'India', the results in the table were obtained by comparing our experimental results against the calculated effort based on the average salary of software developers working in India. A similar procedure was implemented for 'Bangladesh', and so forth for all the countries included in the study.

A comprehensive discussion of the results, implications, and potential risks will be addressed in the following subchapter. However, the full results of the experiment and the associated repositories are available in (Mrvar, 2023).



ML Algorithm	Metric	Country of origin	India	Bangladesh	USA	Pakistan
Autoencoder	MMRE	6,209	0,717	0,615	11,687	0,508
	PRED25	0,057	0,286	0,286	0,000	0,314
	MAE	4,497	4,187	4,354	5,806	4,845
KNN	MMRE	5,890	0,883	0,797	12,214	0,692
	PRED25	0,143	0,229	0,286	0,000	0,257
	MAE	4,261	4,733	5,071	5,477	5,746
Hierarchical Clustering	MMRE	7,987	1,040	0,929	14,105	0,791
	PRED25	0,086	0,257	0,343	0,000	0,343
	MAE	5,187	5,102	5,365	6,361	5,881
SVM	MMRE	5,784	0,739	0,642	11,389	0,538
	PRED25	0,114	0,229	0,314	0,000	0,371
	MAE	4,283	4,418	4,621	5,451	5,127

Table 7.4: Experiment Evaluation

7.2.3 Analysis

As illustrated in Table 7.4, the estimations are significantly more precise when the computed effort is matched with the actual effort calculated from hourly wages in countries such as India, Bangladesh, and Pakistan, compared to the effort derived from the country of origin or the USA. This outcome suggests that projects posted on Freelancer.com could be primarily intended for outsourcing to reduce costs, which consequently raises important questions about the reliability of the bidding prices associated with each project.

It is, however, crucial to consider the performance of the Machine Learning (ML) techniques utilized in this experiment. An examination of the evaluation metrics across the assembled test data allows us to identify the algorithms that yielded the most accurate results. Models that generate lower MMRE and MAE values, in addition to higher PRED25 scores, are indicative of superior performance.

The evaluation metrics reveal that the Autoencoder algorithm consistently registers the lowest MMRE among all methods, notably for countries like India, Bangladesh, and Pakistan. This suggests superior performance of the Autoencoder model in terms of the MMRE metric for these countries.

Upon observing the Pred(25%) metric, it is noteworthy that all models exhibit a value of 0 for the USA. This implies that none of the estimated projects fell within 25% of the actual values when the effort was compared with that of the USA. This finding supports the notion that the prices set on Freelancer.com are oriented towards outsourcing and cost reduction. However, for India, Bangladesh, and Pakistan, the Autoencoder model exhibits the highest Pred(25%) score, which further indicates its superior performance over the other models.

Considering the MAE, the models provide relatively weak results regarding the average absolute error. This could suggest that the input data may be insufficient or noisy, considering that the descriptions of the projects listed on Freelancer.com may not provide a comprehensive explanation of the project requirements.

Upon reflection, it is clear that the data collection process posed significant challenges. This is especially true considering that clients seeking freelancers tend to prefer the lowest bids. Furthermore, it is difficult to determine the generalizability of the effort calculations since they were based on the average hourly wage in a country, potentially leading to discrepancies across different levels of expertise. Finally, it is important to acknowledge that project descriptions on the platform are often vague and may not encompass the entire development process. As a result, extracting the right requirements can prove to be a complex task.

Detailed discussions on the results, the efficacy of the developed model, and reflections on the overall study will be further elaborated upon in Chapter 8.

Chapter 8

Discussion

This chapter aims to critically assess this study and its implications within software cost estimation research. Initially, the developed estimation model is rigorously investigated and discussed. Subsequently, potential threats to the validity of the results and the overall analysis are elucidated, along with relevant ethical considerations.

8.1 Final Estimation Model

The comprehensive process of utilizing machine learning (ML) techniques for estimation purposes manifests as a cyclic, ongoing endeavor. As such, there is no definitive guarantee of the model's precision or the effectiveness of the approach adopted in this study. This research study primarily targeted software cost estimation using GitHub repositories, wherein we relied upon Systematic Literature Review (SLR) and expert interviews to collect the factors essential for executing such an estimation. Consequently, it is plausible that the factors proposed in this study could be prone to a degree of bias; furthermore, we cannot assert with certainty that the selected factors are the most optimal for conducting effort estimation. The same applies to the ML techniques employed in the process. The SLR resulted in various ML techniques, making selecting the most appropriate one quite challenging.

It is vital to underscore that only 5699 repositories were incorporated into the dataset. This posed difficulties when evaluating, particularly when assessing smaller projects, given that merely 146 repositories in the dataset feature software development durations of less than one month. This limitation is primarily attributable to the dataset's size. However, it can also be ascribed to our focus on repositories with more than five stars on GitHub, which suggests that only larger repositories were considered. It is crucial to note that the objective of this study was to test the feasibility of data collection and estimation execution, not to test the completeness of the dataset. Another notable point concerning the dataset is that the data collected originates from open-source, publicly available projects. Consequently, it is challenging to assert how generalizable the results of the estimation model can be in the context of projects being developed within private organizations.

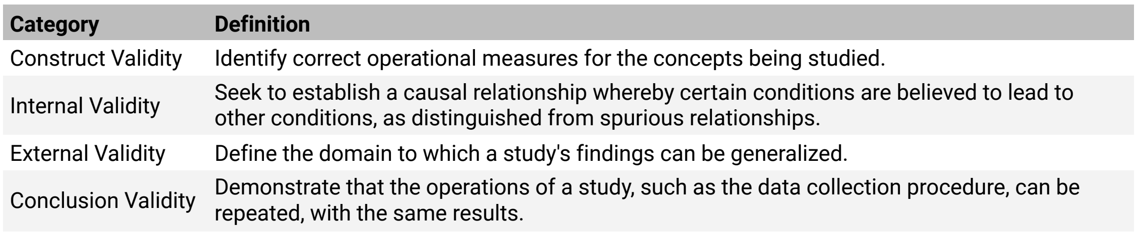
An additional challenge that became evident during the evaluation phase was that some factors chosen for performing the estimation could be challenging to articulate accurately. This factor could consume a considerable amount of time and result in biased decisions when executing the estimation.

Acknowledging all these issues, the procedural steps and code necessary to reproduce the process were made publicly available. We anticipate that future researchers in this domain will find this availability helpful, simplifying their efforts when conducting studies in this field. Therefore, we invite anyone to continue researching this complex and challenging field.

8.2 Threats to Validity

Evidence-Based Software Engineering (EBSE), proposed by (B. A. Kitchenham, Dyba, & Jorgensen, 2004), advocates guidelines for evidence-based software production. Consequently, the systematic literature review (SLR) has emerged as an essential technique for amassing evidence and insights about this research field. To certify the scientific value of the results yielded by the SLR, we must stringently assess its validity. Moreover, expert interviews, a case study, and the design of the final artifact together form a complex trajectory that must maintain a consistent level of scientific contribution from this study.

For these reasons, the validity of this study has been meticulously evaluated throughout its course. However, akin to all research, potential threats to validity warrant careful consideration when interpreting and applying the findings of this study. (X. Zhou, Jin, Zhang, Li, & Huang, 2016) examined the standard practices of Threats to Validity (TTVs) in 316 SLRs concerning the software engineering research field. Subsequently, they compiled a list of common threats to validity, illustrated in Table 8.1. The subsequent sections will explore these threats in the context of this research project's validity.



Category	Definition
Construct Validity	Identify correct operational measures for the concepts being studied.
Internal Validity	Seek to establish a causal relationship whereby certain conditions are believed to lead to other conditions, as distinguished from spurious relationships.
External Validity	Define the domain to which a study's findings can be generalized.
Conclusion Validity	Demonstrate that the operations of a study, such as the data collection procedure, can be repeated, with the same results.

Table 8.1: Definitions of validities based on (X. Zhou et al., 2016)

8.2.1 Construct Validity

The construct validity, which pertains to identifying accurate operational measures for the concepts discerned in the executed SLR, primarily hinges on the studies accumulated during the SLR and their collection methodology. One potential threat could arise from online libraries, where papers are collected, or un rigorous venues. Furthermore, the exclusion and inclusion criteria that we have established could be biased. Therefore, each step undertaken in the SLR was meticulously documented in Chapter 3. This research concentrated on methods prevalent in recent research papers, which implies that the concepts had to be recurrent for us to perceive them as significant.

Construct validity about expert interviews and the final artifact concerns whether the metrics and information used during the process accurately represent the phenomena they aim to measure. The methods and information employed during the research were precisely defined to enhance the construct validity, and data from the SLR were organized and elucidated. In addition, the questions posed during the interviews followed an interview protocol meticulously devised to capture pertinent information. Furthermore, the final artifact was designed and evaluated utilizing well-established methodologies in software engineering.

8.2.2 Internal Validity

Threats to internal validity are primarily associated with case studies and interviews, wherein our inherent bias could influence the selection of the participants, the organization, and the definition of the questions. Hence, the selection procedure of participants was executed rigorously, and risk was mitigated through systematic planning and consultation with all participants involved in the research.

Internal threats to validity are also evident in calculating factors using Natural Language Processing (NLP) and other tasks to derive a specific factor from a repository. As previously stated, achieving a perfect solution to such tasks is unlikely. Therefore, it is crucial to acknowledge that this aspect may carry a degree of bias.

Another threat to internal validity arises from potential bias from the researchers' side during the SLR, particularly during the screening phase, where the overview of the papers under evaluation was constrained. Consequently, all decisions and their rationale were documented in Chapter 3.

8.2.3 External Validity

The scope of this study is primarily concentrated on software development, rendering it fairly specific. Consequently, the findings of this study can be generalized exclusively to software development projects. This includes only coding projects; thus, extrapolating results to low-code platforms or projects with significantly divergent approaches, especially those outside software development, is complex and probably unfeasible.

Additionally, regarding the generalizability of the executed SLR, a minor percentage of studies were not accessible during this investigation. However, given that the majority of papers were accessible, we can reasonably conjecture that the missing papers would not considerably alter our findings.

Finally, the case study's nature and evaluation often circumscribe the generalizability of the findings. Therefore, additional research is necessitated to ascertain whether the results derived from this study can be extrapolated to other contexts.

8.2.4 Conclusion Validity

To illustrate and affirm the repeatability of the SLR results, each step was carefully documented, and a comprehensive description of those steps was elucidated in Chapter 3. Furthermore, the results of the SLR were made publicly available after the completion of the study, which will provide future researchers with additional insights into the execution of this study. The complete output of the executed SLR can be found in (Mrvar, 2023).

This also applies to the final estimation model and data collection procedure, with the final artifact available in (Mrvar, 2023). The code was made publicly accessible to facilitate understanding of how this study was conducted and its underlying logic for future researchers. This accessibility will also ease the replication of this study, should the need arise.

8.3 Ethical Considerations

Ethical considerations constitute a critical component of the research process. With regard to data privacy and security, all data that could potentially identify any individual were anonymized and secured. The actual names of the experts were concealed, and the organization participating in the experiment was also obscured to uphold privacy.

Concurrently, results and conclusions were articulated as accurately as possible. The methodologies employed were transparent and were made publicly available to foster reproducibility. This implies that future researchers in this area should be capable of replicating the study. In addition, we have strived to acknowledge all limitations encountered during this study, which should assist future researchers in identifying areas requiring further investigation.

Concerning intellectual property, all tools and papers utilized during this study were appropriately cited and acknowledged.

Chapter 9

Conclusion

This chapter aims to critically evaluate the research conducted during this thesis and elucidate its principal objectives. It responds to the research questions, assesses the achievement of goals, and recommends future research directions and targets in this field.

9.1 Cost Modeling and Estimation (MRQ)

To address the main research question, "How can a robust and systematic approach be developed to model the cost estimation of software projects, considering the essential factors of stakeholder requirements and priorities?" our research has developed an estimation model exhaustively documented within this thesis. The conceptual model, elaborated in Chapter 1, guided this process meticulously. Repositories were systematically scraped from GitHub and mapped to factors identified in the systematic literature review (SLR) and expert interviews, followed by dataset creation. The Constructive Cost Model (COCOMO) II (Boehm et al., 1995) computed effort for each repository based on lines of code. Finally, machine learning techniques were employed to find similar repositories based on the future project requirements, enabling effort estimation for the anticipated software development.

The main research question is supported by several sub-questions. Addressing these will provide a comprehensive picture and rationale behind the decisions made while crafting this model.

9.2 Literature on Cost Estimation (RQ1)

The state-of-the-art cost estimation methodologies presented in the literature primarily rely on existing datasets to perform effort estimation. This implies that the majority of approaches do not concentrate on gathering fresh data for effort estimation, instead, the focus is predominantly on improving the modeling aspect rather than creating a complete pipeline for effort estimation (encompassing both data extraction and modeling). This has been thoroughly elaborated in the findings of the literature review in Chapter 3.

Nevertheless, the systematic literature review uncovered 178 distinct models and methods. The most frequently encountered approaches in the literature include neural networks (Hammad & Alqaddoumi, 2018), K-nearest neighbors (KNN) (Shukla & Kumar, 2019), and multilayer perceptron (MLP) (Ali & Gravino, 2021), among others. Machine learning emerged as the most prevalent approach, followed by regression and neural networks.

With such a plethora of approaches, the decision on the most suitable one to address the effort estimation problem can pose a significant challenge.

9.3 Parameters in Estimation Approaches (RQ2)

In the systematic literature review (SLR), we observed that most approaches using machine learning (ML) techniques did not prioritize gathering data for effort estimation. However, a few methodologies did focus on manual data collection to assemble a dataset for effort estimation (Senevirathne & Wijayasiriwardhane, 2020). Moreover, numerous papers specifically referenced parameters that could potentially impact the cost of software development, even though some of these parameters are challenging to quantify or collect using extensive data methodologies. For instance, (Suliman & Kadoda, 2017) identified factors that include frequent changes in software requirements, political issues, management considerations, conflicts, and others.

Despite the challenges associated with gathering and identifying correct factors, we successfully gathered 254 factors and key performance indicators (KPIs). After organization and grouping, the final list comprised 106 factors. Additionally, we collected and analyzed datasets used by various effort estimation approaches and identified their attributes. To accomplish this, we extended the study conducted by (E. Mustafa & Osman, 2018), which resulted in a final tally of 48 attributes obtained from the collected datasets.

The complete list of identified factors and potential parameters derived from this study can be found in Chapter 3, and a comprehensive explanation is available in (Mrvar, 2023).

9.4 Data Acquisition Techniques (RQ3)

Throughout the project, we predominantly used the GitHub GraphQL API to extract repositories from GitHub. The procedure was primarily informed by the API documentation available on GitHub's official website, which also provides the GraphQL Explorer[1]. As outlined in Chapter 5, no other scraping techniques were employed in this project, except for two additional APIs that facilitated the extraction of lines of code and readme files for each repository.

9.5 Parameters for Cost Estimation Model (RQ4)

In order to answer the question, "Which parameters should be used to construct a cost estimation model for software projects?" we conducted an extensive literature review detailed in Chapter 3. As previously mentioned, we identified 106 factors through the SLR, all of which could serve as parameters for the cost estimation model. Consequently, we conducted several expert interviews to pinpoint the most valuable factors that could be parameters for the final estimation model.

The interviews enriched our understanding of the collected factors and introduced new potential factors for consideration. Furthermore, the list of factors was reduced, with less significant attributes being removed. The entire process and outcome of the interviews are presented in Chapter 4.

The final challenge involved determining which factors were feasible to acquire during data extraction and mapping. Accordingly, they were divided into two categories, automatic and manual factors. Manual elements were incorporated using the COCOMO II model, while automatic factors were incorporated through machine learning. The rationale and procedure behind these decisions are expounded in Chapters 4 and 5. It is essential to note that most

[1] https://docs.github.com/en/graphql/overview/explorer

decisions were taken in consultation with domain experts, but this does not preclude the possibility of residual bias in the results.

9.6 Extracting Project Requirements (RQ5)

Data extraction from GitHub occasionally made specific project requirements, indicative of the final effort, plainly visible. This includes parameters such as programming languages, lines of code, documentation size, standards, and others. We employed algorithmic techniques like natural language processing (NLP) to identify less evident factors. For instance, an NLP classifier was used to determine the industry sector and software type from the Readme file. Similarly, the database management system (DBMS) was identified by counting the occurrences of the most common DBMS in the readme file, thus indicating the probability of a particular DBMS being utilized in the corresponding repository. A comprehensive explanation of this process is provided in Chapter 5. However, it is essential to mention that using these techniques to extract specific attributes or requirements from a repository does not guarantee absolute accuracy.

9.7 Estimating Cost Based on Project Similarity (RQ6)

The systematic literature review (SLR) identified 178 unique models and methods for software cost estimation. However, this study aimed to seek similarities between the input data and the software projects within our knowledge base. As a result, we employed the K-nearest neighbors (KNN) and hierarchical clustering algorithms, given their long-standing and widespread use (Ogunleye, 2020). Additionally, we applied the Support Vector Machine (SVM) method due to its prevalence in the SLR. We also utilized the Autoencoder algorithm, a novel approach to software effort estimation, and proposed a variant of this approach (Qi et al., 2021), thereby adding an innovative aspect to our study.

The estimation was conducted based on the similarity between the input data and the projects within the knowledge base. This led to identifying the ten most similar repositories, from which outliers were removed, and the average effort was computed. A detailed explanation of the estimation process can be found in Chapter 6.

It should be noted that the final estimation depends on a multitude of factors, ranging from the input values and attributes used to ascertain similarity to the models employed. Nevertheless, this study's primary objective was not to find the best model but to demonstrate the feasibility of the complete process, which includes data extraction, mapping, and effort estimation. Consequently, while we cannot assert that the approach produced the best possible results, it does provide a valuable starting point for future research in this area.

9.8 Evaluating the Cost Estimation Model (RQ7)

Several methods are available for the evaluation of cost estimation models. Our extensive SLR identified the most prevalent existing methods, resulting in 86 evaluation methods. These findings are documented in Chapter 3.

Given that each evaluation method is unique and suited to different scenarios, selecting the best one can present a challenge. However, we employed the most popular evaluation methods for this project based on their frequency in the SLR. Consequently, Pred(25), MMRE, and MAE were the evaluation metrics used in this study.

9.9 Future Research Directions

The findings of this research offer a platform for future investigation in software cost estimation. One potential road lies in exploring and identifying novel factors that could improve the existing models established in this study. Similarly, the machine learning (ML) models utilized in this research could be replaced or augmented with alternative approaches to enhance cost estimation accuracy.

Another promising area of future research lies in the exploration of time estimation strategies that do not rely on the COCOMO II model for effort estimation per repository based on lines of code. A potential solution could involve leveraging repositories or issue-tracking systems that provide time frames for problem resolution. However, ensuring such tasks' reliability and adequate documentation can pose a significant challenge during data extraction.

Additionally, a more dynamic cost estimation model could be proposed that does not require pre-defined factors. Such a model could work directly with GitHub repositories or similar platforms, relying on comprehensive project descriptions.

Lastly, the rapidly evolving field of artificial intelligence continues to leave significant impacts across all technological areas. A potentially exciting research direction, in the context of software effort estimation, would be the exploration of emerging language models like the Generative Pre-training Transformer (GPT)[2]. However, it is essential to mention that these advanced models were not included in this study. We focused on employing open-source methodologies to ensure wider accessibility and reproducibility of our work.

[2] https://openai.com/gpt-4

Appendix A

Interview Protocol

The interview protocol employed in the expert interviews conducted within the scope of this study is outlined in Table A.1.

Step	Content
1.	A brief description of the project and the main goal of the interview
2.	Introductory questions
	How long have you worked in software development (or a similar field)? What kind of software is developed in your organization? (which area etc.) Is your organization using agile practices? Would you say cost estimation is an important part of software development? Have you heard of any cost estimation techniques?
3.	Cost estimation
	How are you currently performing cost estimation? How often are you performing cost estimation? Are you using any tools/platforms for cost estimation? How accurate are you currently with cost estimation? Which factors/KPIs are important for software cost estimation (especially in the beginning)? Considering the factors/KPIs we have gathered in the SLR, which ones would you say are the most important? Which factors would you consider to be useful in training the model? How do you imagine using such a prediction model? Would you expect to provide a list of requirements and then receive the prediction or maybe type in certain parameters to get the prediction? What type of estimation are you expecting?
4.	Closing
	What do you think about our work? Would you consider using a model for estimating the software cost/effort? May we contact you if we have any questions? Can we use your company's name in the scientific paper, or do you prefer an anonymous name? Do you have any questions or additional feedback?

Table A.1: Interview Protocol.

References

Albrecht, A., & Gaffney, J. (1983). Software function, source lines of code, and development effort prediction: A software science validation. *IEEE Transactions on Software Engineering*, *SE-9*(6), 639-648. doi: 10.1109/TSE.1983.235271

Albrecht, A. J. (1979). Measuring application development productivity. In *Proc. joint share, guide, and ibm application development symposium, 1979.*

Albrecht, A. J., & Gaffney, J. E. (1983). Software function, source lines of code, and development effort prediction: a software science validation. *IEEE transactions on software engineering*(6), 639–648.

Ali, A., & Gravino, C. (2021). Improving software effort estimation using bio-inspired algorithms to select relevant features: An empirical study. *Science of Computer Programming*, *205*, 102621. Retrieved from https://www.sciencedirect.com/science/article/pii/S0167642321000149 doi: https://doi.org/10.1016/j.scico.2021.102621

Aslam, W., Ijaz, F., Lali, M. I. U., & Mehmood, W. (2017). Risk aware and quality enriched effort estimation for mobile applications in distributed agile software development. *J. Inf. Sci. Eng.*, *33*(6), 1481–1500.

Bank, D., Koenigstein, N., & Giryes, R. (2020). Autoencoders. *arXiv preprint arXiv:2003.05991*.

Bilgaiyan, S., Mishra, S., & Das, M. (2016). A review of software cost estimation in agile software development using soft computing techniques. In *2016 2nd international conference on computational intelligence and networks (cine)* (p. 112-117). doi: 10.1109/CINE.2016.27

Boehm, B. (1984). Software engineering economics. *IEEE Transactions on Software Engineering*, *SE-10*(1), 4-21. doi: 10.1109/TSE.1984.5010193

Boehm, B., Abts, C., & Chulani, S. (2000). Software development cost estimation approaches—a survey. *Annals of software engineering*, *10*(1), 177–205.

Boehm, B., Clark, B., Horowitz, E., Westland, C., Madachy, R., & Selby, R. (1995). Cost models for future software life cycle processes: Cocomo 2.0. *Annals of software engineering*, *1*, 57–94.

Choetkiertikul, M., Dam, H. K., Tran, T., Pham, T., Ghose, A., & Menzies, T. (2019). A deep learning model for estimating story points. *IEEE Transactions on Software Engineering*, *45*(7), 637-656. doi: 10.1109/TSE.2018.2792473

Chow, T., & Cao, D.-B. (2008). A survey study of critical success factors in agile software projects. *Journal of systems and software*, *81*(6), 961–971.

Cohn, M. (2005). *Agile estimating and planning*. Pearson Education.

Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine learning*, *20*, 273–297.

Db-engines ranking. (2023). https://db-engines.com/en/ranking. (Accessed: 2023-5-28)

de A. Cabral, J. T. H., Oliveira, A. L., & da Silva, F. Q. (2023). Ensemble effort estimation: An updated and extended systematic literature review. *Journal of Sys-*

tems and Software, *195*, 111542. Retrieved from https://www.sciencedirect.com/science/article/pii/S0164121222002187 doi: https://doi.org/10.1016/j.jss.2022.111542

Effendi, A., Setiawan, R., & Rasjid, Z. E. (2019). Adjustment factor for use case point software effort estimation (study case: Student desk portal). *Procedia Computer Science*, *157*, 691-698. Retrieved from https://www.sciencedirect.com/science/article/pii/S1877050919311342 (The 4th International Conference on Computer Science and Computational Intelligence (ICCSCI 2019) : Enabling Collaboration to Escalate Impact of Research Results for Society) doi: https://doi.org/10.1016/j.procs.2019.08.215

Elmers, P. (2023). *github-repository-metadata.* https://github.com/pelmers/github-repository-metadata. (Accessed: 2023-6-5)

Farshidi, S. (2020). *Multi-criteria decision-making in software production* (Unpublished doctoral dissertation). Utrecht University.

Farshidi, S., Jansen, S., & van der Werf, J. M. (2020). Capturing software architecture knowledge for pattern-driven design. *Journal of Systems and Software*, *169*, 110714. Retrieved from https://www.sciencedirect.com/science/article/pii/S0164121220301552 doi: https://doi.org/10.1016/j.jss.2020.110714

Farshidi, S., Kwantes, I. B., & Jansen, S. (2023). Business process modeling language selection for research modelers. *Software and Systems Modeling*, 1–26.

Farshidi, S., & Zhao, Z. (2022). An adaptable indexing pipeline for enriching meta information of datasets from heterogeneous repositories. In J. Gama, T. Li, Y. Yu, E. Chen, Y. Zheng, & F. Teng (Eds.), *Advances in knowledge discovery and data mining* (pp. 472–484). Cham: Springer International Publishing.

Forward, A., & Lethbridge, T. C. (2008). A taxonomy of software types to facilitate search and evidence-based software engineering. In *Proceedings of the 2008 conference of the center for advanced studies on collaborative research: meeting of minds* (pp. 179–191).

Fylan, F. (2005). Semi-structured interviewing. *A handbook of research methods for clinical and health psychology*, *5*(2), 65–78.

Grenning, J. (2002). Planning poker or how to avoid analysis paralysis while release planning. *Hawthorn Woods: Renaissance Software Consulting*, *3*, 22–23.

Hai, V. V., Nhung, H. L. L. L., & Jasek, R. (2022). Toward applying agglomerative hierarchical clustering in improving the software development effort estimation. In *Software engineering perspectives in systems: Proceedings of 11th computer science on-line conference 2022, vol. 1* (pp. 353–371).

Hammad, M., & Alqaddoumi, A. (2018). Features-level software effort estimation using machine learning algorithms. In *2018 international conference on innovation and intelligence for informatics, computing, and technologies (3ict)* (p. 1-3). doi: 10.1109/3ICT.2018.8855752

Hevner, A., & Chatterjee, S. (2010). Design science research in information systems. In *Design research in information systems* (pp. 9–22). Springer.

Hughes, R. T. (1996). Expert judgement as an estimating method. *Information and Software Technology*, *38*(2), 67-75. Retrieved from https://www.sciencedirect.com/science/article/pii/0950584995010459 doi: https://doi.org/10.1016/0950-5849(95)01045-9

International software benchmarking standards group. (2022). https://www.isbsg.org/. (Accessed: 2022-12-26)

Jones, C. (2008). *Applied software measurement*. McGraw-Hill Education.

Kemerer, C. F. (1987). An empirical validation of software cost estimation models. *Communications of the ACM*, *30*(5), 416–429.

Kitchenham, B. (2004). Procedures for performing systematic reviews. *Keele, UK, Keele University*, *33*(2004), 1–26.

Kitchenham, B., Pfleeger, S. L., McColl, B., & Eagan, S. (2002). An empirical study of maintenance and development estimation accuracy. *Journal of systems and software*, *64*(1), 57–77.

Kitchenham, B. A., Dyba, T., & Jorgensen, M. (2004). Evidence-based software engineering. In *Proceedings. 26th international conference on software engineering* (pp. 273–281).

Kramer, O., & Kramer, O. (2013). K-nearest neighbors. *Dimensionality reduction with unsupervised nearest neighbors*, 13–23.

Kumar, G., & Bhatia, P. K. (2012). Impact of agile methodology on software development process. *International Journal of Computer Technology and Electronics Engineering (IJCTEE)*, *2*(4), 46–50.

Maxwell, K. D. (2002). Applied statistics for software managers. *Applied Statistics for Software Managers*.

Mendes, E., Mosley, N., & Counsell, S. (2003). Investigating early web size measures for web cost estimation. In *Proceedings of ease'2003 conference, keele* (pp. 1–22).

Meyer, D., & Wien, F. (2015). Support vector machines. *The Interface to libsvm in package e1071*, *28*, 20.

Miyazaki, Y., Terakado, M., Ozaki, K., & Nozaki, H. (1994). Robust regression for developing software estimation models. *Journal of Systems and Software*, *27*(1), 3–16.

Mrvar, G. (2023, July). *Software Cost Estimation Using Open Source GitHub Repositories.* Zenodo. Retrieved from https://doi.org/10.5281/zenodo.8174397 doi: 10.5281/zenodo.8174397

Mustafa, E., & Osman, R. (2018). An analysis of the inclusion of environmental cost factors in software cost estimation datasets. In *2018 ieee international conference on software quality, reliability and security companion (qrs-c)* (pp. 623–630).

Mustafa, E. I., & Osman, R. (2020). Seera: a software cost estimation dataset for constrained environments. In *Proceedings of the 16th acm international conference on predictive models and data analytics in software engineering* (pp. 61–70).

Ogunleye, J. O. (2020). Review of current data mining techniques used in the software effort estimation. In *Software engineering perspectives in intelligent systems: Proceedings of 4th computational methods in systems and software 2020, vol. 1 4* (pp. 393–408).

Peng, D., Cao, L., & Xu, W. (2011). Using json for data exchanging in web service applications. *Journal of Computational Information Systems*, *7*(16), 5883–5890.

Putnam, L. H. (1978). A general empirical solution to the macro software sizing and estimating problem. *IEEE transactions on Software Engineering*(4), 345–361.

Qi, F., Jing, X.-Y., Zhu, X., Jia, X., Cheng, L., Dong, Y., ... Feng, S. (2021). Heterogeneous software effort estimation via cascaded adversarial auto-encoder. In *Parallel and distributed computing, applications and technologies: 21st international conference, pdcat 2020, shenzhen, china, december 28–30, 2020, proceedings 21* (pp. 17–29).

Qi, F., Jing, X.-Y., Zhu, X., Xie, X., Xu, B., & Ying, S. (2017). Software effort estimation based on open source projects: Case study of github. *Information and Software Technology*, *92*, 145-157. Retrieved from https://www.sciencedirect.com/science/article/pii/S0950584917302239 doi: https://doi.org/10.1016/j.infsof.2017.07.015

Rai, P., Kumar, S., & Verma, D. K. (2020). A hybrid machine learning framework for prediction of software effort at the initial phase of software development. In *Advances* *in computing and data sciences: 4th international conference, icacds 2020, valletta, malta, april 24–25, 2020, revised selected papers 4* (pp. 187–200).

Ralph, P., Baltes, S., Bianculli, D., Dittrich, Y., Felderer, M., Feldt, R., . . . others (2020). Acm sigsoft empirical standards. arXiv.

Ramasubbu, N., & Balan, R. K. (2012). Overcoming the challenges in cost estimation for distributed software projects. In *2012 34th international conference on software engineering (icse)* (pp. 91–101).

Sayyad Shirabad, J., & Menzies, T. (2005). *The PROMISE Repository of Software Engineering Databases.* School of Information Technology and Engineering, University of Ottawa, Canada. Retrieved from http://promise.site.uottawa.ca/SERepository

Scott, E., & Pfahl, D. (2018). Using developers' features to estimate story points. In *Proceedings of the 2018 international conference on software and system process* (pp. 106–110).

Sehra, S. K., Brar, Y. S., Kaur, N., & Sehra, S. S. (2017). Research patterns and trends in software effort estimation. *Information and Software Technology*, *91*, 1-21. Retrieved from https://www.sciencedirect.com/science/article/pii/S0950584917304317 doi: https://doi.org/10.1016/j.infsof.2017.06.002

Senevirathne, D. S., & Wijayasiriwardhane, T. K. (2020). Extending use-case point-based software effort estimation for open source freelance software development. In *2020 international research conference on smart computing and systems engineering (scse)* (p. 188-194). doi: 10.1109/SCSE49731.2020.9313007

Shepperd, M., & Cartwright, M. (2001). Predicting with sparse data. *IEEE Transactions on Software Engineering*, *27*(11), 987–998.

Shepperd, M., & Kadoda, G. (2001). Comparing software prediction techniques using simulation. *IEEE Transactions on Software Engineering*, *27*(11), 1014–1022.

Shepperd, M., & Schofield, C. (1997). Estimating software project effort using analogies. *IEEE Transactions on Software Engineering*, *23*(11), 736-743. doi: 10.1109/32.637387

Shukla, S., & Kumar, S. (2019). Applicability of neural network based models for software effort estimation. In *2019 ieee world congress on services (services)* (Vol. 2642-939X, p. 339-342). doi: 10.1109/SERVICES.2019.00094

Silhavy, R., Silhavy, P., & Prokopova, Z. (2015). Algorithmic optimisation method for improving use case points estimation. *PloS one*, *10*(11), e0141887.

Similarweb. (2023). *Similarweb.* https://www.similarweb.com/. (Accessed: 2023-6-5)

Stankovic, D., Nikolic, V., Djordjevic, M., & Cao, D.-B. (2013). A survey study of critical success factors in agile software projects in former yugoslavia it companies. *Journal of systems and software*, *86*(6), 1663–1678.

Suliman, S. M. A., & Kadoda, G. (2017). Factors that influence software project cost and schedule estimation. In *2017 sudan conference on computer science and information technology (sccsit)* (p. 1-9). doi: 10.1109/SCCSIT.2017.8293053

Suresh Kumar, P., & Behera, H. (2020). Role of soft computing techniques in software effort estimation: an analytical study. In *Computational intelligence in pattern recognition* (pp. 807–831). Springer.

Tadayon, N. (2005). Neural network approach for software cost estimation. In *International conference on information technology: Coding and computing (itcc'05) - volume ii* (Vol. 2, p. 815-818 Vol. 2). doi: 10.1109/ITCC.2005.210

Tawosi, V., Al-Subaihin, A., Moussa, R., & Sarro, F. (2022). A versatile dataset of agile open source software projects. *arXiv preprint arXiv:2202.00979*.

Tiobe index. (2023). https://www.tiobe.com/tiobe-index/. (Accessed: 2023-5-28)

Usman, M., Mendes, E., & Börstler, J. (2015). 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* (pp. 1–10).

Venkataiah, V., Mohanty, R., & Nagaratna, M. (2019). Prediction of software cost estimation using spiking neural networks. In *Smart intelligent computing and applications: Proceedings of the second international conference on sci 2018, volume 2* (pp. 101–112).

Vinutha, H., Poornima, B., & Sagar, B. (2018). Detection of outliers using interquartile range technique from intrusion dataset. In *Information and decision sciences: Proceedings of the 6th international conference on ficta* (pp. 511–518).

Yin, R. K. (1981). The case study as a serious research strategy. *Knowledge*, *3*(1), 97–114.

Zhao, Y., Karypis, G., & Fayyad, U. (2005). Hierarchical clustering algorithms for document datasets. *Data mining and knowledge discovery*, *10*, 141–168.

Zhou, P., Han, J., Cheng, G., & Zhang, B. (2019). Learning compact and discriminative stacked autoencoder for hyperspectral image classification. *IEEE Transactions on Geoscience and Remote Sensing*, *57*(7), 4823–4833.

Zhou, X., Jin, Y., Zhang, H., Li, S., & Huang, X. (2016). A map of threats to validity of systematic literature reviews in software engineering. In *2016 23rd asia-pacific software engineering conference (apsec)* (pp. 153–160).

Ziauddin, S. K. T., & Zia, S. (2012). An effort estimation model for agile software development. *Advances in computer science and its applications (ACSA)*, *2*(1), 314–324.