Fine-SE: Integrating Semantic Features and Expert Features for Software Effort Estimation

Yue Li, Zhong Ren, Zhiqi Wang, Lanxin Yang, Liming Dong, Chenxing Zhong, He Zhang
State Key Laboratory of Novel Software Technology, Software Institute
Nanjing University, Nanjing, Jiangsu, China
yueli.dom@outlook.com,dutrenz@163.com,{502022320013,zhongcx}@smail.nju.edu.cn,
yang931001@outlook.com,lmdongmg@gmail.com,hezhang@nju.edu.cn

ABSTRACT

Reliable effort estimation is of paramount importance to software planning and management, especially in industry that requires effective and on-time delivery. Although various estimation approaches have been proposed (*e.g.*, planning poker and analogy), they may be manual and/or subjective, which are difficult to apply to other projects. In recent years, deep learning approaches for effort estimation that rely on learning expert features or semantic features respectively have been extensively studied and have been found to be promising. Semantic features and expert features describe software tasks from different perspectives, however, in the literature, the best combination of these two features has not been explored to enhance effort estimation. Additionally, there are a few studies that discuss which expert features are useful for estimating effort in the industry. To this end, we investigate the potential 13 expert features that can be used to estimate effort by interviewing 26 enterprise employees. Based on that, we propose a novel model, called Fine-SE, that leverages semantic features and expert features for effort estimation. To validate our model, a series of evaluations are conducted on more than 30,000 software tasks from 17 industrial projects of a global ICT enterprise and four open-source software (OSS) projects. The evaluation results indicate that Fine-SE provides higher performance than the baselines on evaluation measures (*i.e.*, mean absolute error, mean magnitude of relative error, and performance indicator), particularly in industrial projects with large amounts of software tasks, which implies a significant improvement in effort estimation. In comparison with expert estimation, Fine-SE improves the performance of evaluation measures by 32.0%-45.2% in *within*-project estimation. In comparison with the state-of-the-art models, Deep-SE and GPT2SP, it also achieves an improvement of 8.9%-91.4% in industrial projects. The experimental results reveal the value of integrating expert features with semantic features in effort estimation.

CCS CONCEPTS

• **Software and its engineering → Software creation and management**.

KEYWORDS

Effort estimation, AI for SE, deep learning

ACM Reference Format:
Yue Li, Zhong Ren, Zhiqi Wang, Lanxin Yang, Liming Dong, Chenxing Zhong, He Zhang. 2024. Fine-SE: Integrating Semantic Features and Expert Features for Software Effort Estimation. In *2024 IEEE/ACM 46th International Conference on Software Engineering (ICSE '24), April 14–20, 2024, Lisbon, Portugal.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3597503.3623349

1 INTRODUCTION

Effort estimation is a critical task of software development planning and management to ensure that the product is delivered on time and within budget. Reliable effort estimation can help companies integrate human resources [1], reduce development costs [30], accelerate delivery cycle [39], and improve software quality [31]. However, the Standish Group-CHAOS 2020 survey of software industry practitioners states that about 60% of software projects are delivered later than expected [19]. Effort estimation remains a challenging activity in software development [32, 49]. The difficulty primarily arises from the interplay of numerous factors, including rapid iterations of software development, constantly evolving technology, and diverse teams [5, 8]. In particular, the problem is exacerbated in industry [7]. Almutlaq et al. [7] executed a review study in which they identified the current issues and gaps in model-based methods for estimating effort. Specifically, they report two types of challenges related to model-based methods in software effort estimation, that is, performance challenges and dataset challenges. In addition, some researchers also pay attention to the effort estimation of enterprise projects. Seo et al. investigated the effect of outlier elimination on the accuracy of effort estimation using two industry datasets, the ISBSG Release 9 and the Korea's Bank dataset [48]. After that, Usman et al. conducted an exploratory case study in order to determine how effort estimation is carried out in large-scale distributed agile projects and to analyze how accurate it is [52].

Various approaches have been proposed for reliable effort estimation and they can generally be divided into expert-based methods and model-based methods [23, 53]. Expert-based methods rely on domain experts to estimate such as planning poker [29], expert estimation [22], and analogy [50]. Expert-based methods rely heavily on the experience of humans, which may introduce bias, and are difficult to reuse in other projects [5, 18, 44]. Over the past few years, model-based approaches have become increasingly popular [45, 46, 55]. Model-based approaches leverage data from old projects to make predictions about new projects. Deep Learning is widely used as a model-based approach for estimating effort due to its ability to mine the data and low human cost [44, 54]. A variety of state-of-the-art research has been proposed to boost the estimation tasks for software practices, and has made great progress with promising outcomes so far.

In the literature, state-of-the-art methods for effort estimation rely on deep learning techniques to build estimation models using semantic features or expert features. Semantic features are derived primarily from semantic information and syntactic structure contained in titles and descriptions of software tasks [34]. Expert features are defined based on the expert's understanding of effort estimation, as well as their expertise and experience, and are used as the input of estimation models [34]. In the literature, Xia et al. propose a set of expert features for GitHub commit count estimation (as a measure of the effort for the repositories) based on their experience [56]. Expert features are proposed because they are crucial to the development activities of GitHub projects. For the estimation of effort, Scott et al. proposed five developer-related features, including the total work capacity of developers [47].

The current state-of-the-art techniques for effort estimation are, however, limited in features and accuracy. For example, (1) The research on what expert features can be used to estimate effort is still in its infancy. Although some studies have suggested that several expert features can be used to estimate effort, no studies have investigated which features may be used to estimate effort in industry. (2) Expert features and semantic features represent different characteristics of software tasks from different dimensions, which have not yet been explored together for effort estimation. In a recent study, Tawosi et al. argue that semantic features are insufficient to differentiate software tasks based on their title or description according to the evaluation results [51]. Model-based methods still have poor performance, which results in rarely using them in practice. According to Moløkken's study [33], a lack of evidence that model-based methods are accurate or even useful in industry is an important reason why model-based methods are rarely used.

By integrating expert features and semantic features, it can be anticipated that effort estimation would be significantly improved. Generally, we hypothesize that effort estimation can be performed using an integrated model by integrating semantic features with expert features behind software tasks. Based on our hypothesis, we explore the importance of integrating semantic features with expert features for effort estimation, and present a novel model (called Fine-SE) with the widely used pre-trained model, BERT. In particular, this study aims to assist project managers in estimating efforts for long-term planning. According to our knowledge, our study is the first attempt to develop an automatic estimation model that leverages semantic features and expert features. Specifically, our study consists of four steps: (1) 26 interviews were conducted with industrial employees to identify initial set of expert features, (2) expert features and semantic features are extracted from software artifacts and their contexts with BERT, (3) integrated feature learning process was carried out using a fully connected layer, and (4) used the integrated features for effort estimation.

For the purpose of evaluating the effectiveness of Fine-SE, we compare its results with expert estimation and state-of-the-art models on 17 industrial projects and four OSS projects with more than 30,000 software tasks (which are referred to as Allocated Requirements (ARs) in the enterprise and Story Points (SPs) in OSS). As measured across industrial projects, Fine-SE outperforms expert estimation by 32.0%-45.2% in within-project estimation, and outperforms Deep-SE and GPT2SP by 8.9%-91.4% on evaluation measures (*i.e.*, mean absolute error, mean magnitude of relative error, and performance indicator).

The main contributions of this study can be concluded as follows:

- **Expert features**. Expert features were identified for effort estimation by conducting 26 interviews with software engineers in the enterprise. These expert features can be used to fill in the gap that expert features are not currently explored for effort estimation in the community.
- **Fine-SE**. To the best of our knowledge, this study is the first attempt to build an automatic model, Fine-SE, by leveraging semantic features and expert features for accurate effort estimation.
- **Evaluation of integrated features**. We evaluate the effectiveness of semantic features and expert features for effort estimation on more than 30,000 software tasks, and our results reveal that the integration of semantic and expert features is clearly more effective in effort estimation than the state-of-the-art models and the use of either feature alone.

2 BACKGROUND AND MOTIVATION

To the best of our knowledge, this is the first study to investigate effort estimation based on extensive interviews and large enterprise data. In this section, we first present the background related to this study, and then we discuss the motivation behind it in order to ease a better understanding.

2.1 Background

2.1.1 Expert Estimation in the Enterprise. The study is a collaboration with a global ICT enterprise that operates in more than 170 countries and regions. In general, enterprise software development is divided into three phases, which are design, development, and testing.

Design phase refers to software designers decomposing System Requirements (SRs) into fine-grained Allocated Requirements (ARs).

Development phase refers to the process in which the designed ARs are assigned to the developer(s), and the developer(s) understand the ARs, code, self-test, fix defects, and so on.

Testing phase refers to once ARs are assigned to testers, they start creating test cases to test those ARs accordingly. When a code commit arrives, the testers should do a test, and if any defects are detected, a defect report is generated. And then the relevant developers would receive specific bug reports.

Before the design phase, the project managers organize a kickoff meeting with designers, developers, and testers to determine SRs that will be included in the current project version. After the design phase, project managers are required to estimate the number of person-months needed to complete each AR from the beginning of the development phase to the completion of the test phase.

2.1.2 Data Collection. In this research, we concentrate on evaluating the performance of the proposed estimation approach on real-world projects. We collected 17 industrial projects from the enterprise and four OSS projects from the JIRA system. The evaluation of different community projects has the potential to provide a more comprehensive perspective on our work.

Enterprise projects. We collected data provided by our industrial collaborator, a leading global ICT enterprise with operations in over 100 countries and regions. To conduct our study, we selected only those enterprise projects that allow access to their software repositories and meet the following criteria:

(1) The projects are available in continuous release versions;
(2) The projects represent the main business lines of the enterprise (e.g., platform development, application development);
(3) The raw data of the projects relating to software artifacts can be obtained from associated software repositories;

In addition to applying the above criteria, we check the reliability of the data by interviewing the project managers and only include projects in which project managers confirm the reliability of data. As a result, 17 projects were selected from the enterprise repositories.

Table 1: Details of the selected projects

Project	Time Span	#AR	Effort					#E1	#E2	#Bug
			Min	Max	Mean	Median	Std			
DM	2014-03~2022-10	4320	0	195	7.91	3	16	92	91	395
ME	2014-02~2021-06	414	1	13	3.93	3	2.6	68	68	119
TD	2009-08~2023-01	73	0.5	40	8.25	8	7.4	19	15	1
US	2014-01~2017-12	100	1	8	3.31	3	1.2	21	21	62
Total	2014-01~2023-01	4907	-	-	-	-	-	200	195	577
P1	2021-04~2023-02	510	0.1	22.9	2.9	2.6	2.1	26	13	1352
P2	2021-12~2023-01	509	0.1	16.1	4.7	4.6	1.7	51	17	385
P3	2022-01~2023-01	553	0.1	9.0	5.9	6.7	2.4	8	6	1485
P4	2019-03~2023-02	4241	0.1	45.3	4.6	4.0	3.3	157	142	9630
P5	2022-02~2023-02	419	0.3	12.9	4.2	3.9	2.3	32	30	407
P6	2019-11~2022-07	3335	0.9	32.0	5.8	5.3	3.4	56	19	2736
P7	2019-02~2023-02	2374	0.7	38.3	15.0	14.7	8.4	37	12	1355
P8	2019-10~2022-12	7401	0.3	34.1	6.7	6.1	2.8	40	19	1859
P9	2020-11~2023-01	322	1.0	26.4	9.5	9.1	4.8	35	10	1592
P10	2020-09~2022-12	286	0.7	15.7	4.5	4.0	2.3	34	8	1697
P11	2021-05~2023-02	257	0.3	17.7	4.3	3.4	2.7	62	31	649
P12	2022-08~2023-02	261	1.3	21.7	8.2	8.1	3.5	63	39	16217
P13	2019-07~2022-12	605	0.1	29.0	5.1	4.6	2.9	47	20	13104
P14	2021-04~2022-12	241	1.9	35.9	15.5	13.9	7.8	32	11	1206
P15	2018-12~2022-12	1167	0.1	35.3	4.0	3.3	2.9	20	13	8251
P16	2018-11~2022-12	1794	0.1	48.9	9.1	5.9	7.9	14	14	36746
P17	2020-02~2023-02	1613	0.9	46.6	5.7	5.0	2.8	71	25	3784
Total	2018-11~2023-02	25888	-	-	-	-	-	785	429	102455

[1] E1 refers to creators in OSS projects, while developers in enterprise projects; E2 means reporters in OSS projects, while testers in enterprise projects

[2] The unit of the effort is the person-month.

We denote the selected projects from software repositories as P1 to P17. The details of the selected projects are listed in Table 1, including time span of projects and the number of major information (*i.e.*, #AR, effort). Overall, selected projects include between 241 and 7401 ARs that can be scaled across multiple industrial projects.

OSS Projects. The OSS projects are also used to evaluate the potential impact of our model on different community projects in order to provide researchers with a more comprehensive perspective for evaluating our work. Jira was selected as the issue tracking system of OSS projects that will be mined to obtain a dataset for analysis, which is one of the few widely-used issue tracking systems for managing agile projects and story point estimation [11, 18, 51]. As shown in Table 1, we obtained the data of four OSS projects from JIRA: Data Management (DM), Mesos (ME), Talend Data Quality (TD), and Usergrid (US). Expert features of OSS projects are available in the JIRA system: (1) *Creator_stories* is the number of stories created by the creator, (2) *Developer_stories* indicates the number of stories assigned to the developer, (3) *Tester_stories* refers to the number of stories tested by the tester. These expert features were obtained through interviews, and corresponding data were obtained from the JIRA system. We provide a detailed description of the source of these expert features in Section 3.

2.1.3 Expert Features and Semantic Features. **Expert features** are descriptive features defined by experts according to their professional knowledge and experience in measuring the effort of software tasks. Over the past few decades, several expert features for effort estimation have been defined at a change-level from a variety of dimensions in the literature. A number of features regarding the number of monthly commits, such as monthly_commit_comments and monthly_contributors, are defined by Xia et al. [56] to measure the development effort of a repository. Scott et al. use a predictive model to estimate effort based on x features of developers, such as the total amount of work capacity (effort), developer comments, and so on [47]. Unfortunately, the expert features have mainly been proposed by the authors based on their own experience or references from other fields but have not been recognized by experts or practitioners in effort estimation. Furthermore, these studies focus on OSS projects and do not discuss the features in the industry that can be used for effort estimation. To identify expert features for effort estimation, 26 software engineers were interviewed over a period of several months based on interview guidelines. As a result, finalized expert features derived from interviews play an important role in integrating features for estimating effort.

Semantic features are a theoretical unit of meaning-holding components that are used to represent a word's meaning [4]. Semantic features have been widely applied in software engineering (SE) to represent the intrinsic characteristics of text with deep learning techniques, such as BERT and GPT [20], by capturing both the meaning and context of tokens within the text (*e.g.*, semantic and syntactic structural information of software tasks). A pre-training model called BERT is proposed in recent research, which outperforms the state-of-the-art models in the tasks relating to question answering and language inference [15]. In this research, we extract semantic features from software tasks using an advanced pre-trained model, BERT, and fine-tune it for use in effort estimation.

2.2 Motivation

Effort estimation has been attracting increasing attention in the literature to boost software delivery on time, and it has actually achieved promising results for managers to automatically estimate the effort. However, effort estimation remains an open question due to various limitations (*e.g.*, incomplete expert features and independently applied features), which motivates us to conduct this study to advance its development.

Incomplete expert features. The state-of-the-art techniques of effort estimation are highly dependent on features mined in the related datasets [21, 47]. In recent years, semantic features have attracted more attention due to the rapid development of pre-trained models. However, the complexity of the estimation work exceeds the capabilities of the pre-trained models. Several researchers have explored expert features in their studies, but few of them are related to software tasks and the related features are incomplete. Xia et al. [56] proposed a set of features that would help to estimate the development effort, such as commits and comments in a month. However, these features are proposed by researchers in other fields such as code review, rather than being proposed by experts and practitioners in effort estimation. Scott et al. [47] proposed five developer-related features for the estimation of effort, including the number of issues assigned to each developer. The number of these features is limited and may not reflect the characteristics of the software task effectively. **To fill this gap, we interviewed 19 managers and 7 engineers about their perceptions of what expert features can be used for effort estimation, so as to mitigate the impact of incomplete expert features on estimation accuracy.**

Independently applied features. Expert features are defined based on the participants' knowledge and experience, which are the extrinsic understanding from the human perspective. Semantic features represent the intrinsic characteristics of software tasks and their contexts, which have been mined by practitioners to estimate the effort of stories using deep learning (*e.g.*, Deep-SE [4] and GPT2SP [18]). Recent research has presented that only relying on semantic features can not be effective in estimating effort, since it can not distinguish between different software tasks [51]. To improve effort estimation, the researchers suggest exploring and discovering more features in future work. As a matter of fact, expert features and semantic features represent characteristics of software tasks from the perspective of the intrinsic structure and expert knowledge, respectively. The combination of expert features and semantic features has produced more effective results than single-type features in a wide range of fields, such as the localization and prediction of defects [34]. However, state-of-the-art research has not yet explored the integration of expert features with semantic features for effort estimation. **We intuitively hypothesize that leveraging expert features and semantic features could be used to improve effort estimation.** The purpose of our study is to fill this gap by investigating the combination of expert features and semantic features to verify our intuitive hypothesis.

3 INTERVIEWS

By conducting interviews in the enterprise, we identified the expert features that can be used for effort estimation. As a result of interviewing 26 enterprise employees, we obtained 13 expert features that can be used to estimate effort. This section describes how the interviews were executed and the results.

3.1 Procedures of Interviews

In enterprises where financial gain is a priority, effort estimation is significantly important. Currently, there is uncertainty about what expert features can be used to estimate effort before software development begins. Therefore, we identify which features of the enterprise are recognized by interviewing the enterprise employees for estimating effort before proposing our approach.

3.1.1 Protocol. Interviews are a common qualitative method in empirical software engineering research that help researchers understand how interviewees think about a phenomenon. We conducted the interviews in reality to obtain reliable information from interviewees about what expert features can be used for effort estimation. We designed 21 basic open-ended questions for the interviews. Specifically, the answers to the first four questions provide us with a deeper understanding of the teams and work content (*e.g.*, size of the teams, roles, and responsibilities). The following 17 questions (*e.g.*, perspectives on expert features) are designed for investigating what expert features can be used for effort estimation. According to the answers provided by the interviewees, our questions were reordered and supplemented. The list of questions for our interviews is available online[1].

3.1.2 Participants. Interviewees were managers and software engineers from 17 industrial projects selected according to the selection criteria 1-3 described in Section 2.1.2. The interviewees were selected by the authors and the cooperation project manager, who provided assistance and advice for our study as well as the enterprise's background information, such as the various departments, data records, etc. To obtain a broad and sufficient perspective, we interviewed employees in two roles from different departments, including project managers and software engineers (*i.e.*, developers and testers). Over a period of three months, 26 employees (including 19 project managers and 7 software engineers) were interviewed regarding their perceptions of what expert features can be used for estimating effort. A total of 26 semi-structured interviews were conducted by three researchers, with each interview lasting approximately 30 minutes. Two experienced researchers asked the questions, while the third posed complementary questions and controlled the entire process. A total of 12 hours of recording of the interviews were included in the dataset, all of which were recorded with the permission of the interviewees.

[1]The list of questions for interviews can be found online at the following site: https://github.com/yueli-se/Fine-SE

Table 2: Expert features and quantitative results of interviews.

Features	ID	Description	Number of Employees and Ratio		
			Manager (n=19)	Engineer (n=7)	Total (n=26)
Creator_ARs	F1	Total number of ARs created by the creator	11, 57.9%	4, 57.1%	15, 57.7%
Developer_ARs	F2	Total number of ARs developed by the developer	16, 84.2%	6, 85.7%	22, 84.6%
Tester_ARs	F3	Total number of ARs tested by the tester	7, 36.8%	4, 57.1%	11, 42.3%
Developer_commits	F4	Total number of commits created by the developer	11, 57.9%	5, 71.4%	16, 61.5%
Developer_commits_reviews	F5	Total number of commit reviews created by the developer	11, 57.9%	5, 71.4%	16, 61.5%
Developer_modified_files	F6	Total number of files modified by the developer	8, 42.1%	3, 42.9%	11, 42.3%
Developer_created_MRs	F7	Total number of merge requests created by the developer	9, 47.4%	4, 57.1%	13, 50.0%
Developer_updated_MRs	F8	Total number of merge requests updated by the developer	9, 47.4%	4, 57.1%	13, 50.0%
Developer_fixed_defects	F9	Total number of defects fixed by the developer	12, 63.2%	5, 71.4%	17, 65.4%
Tester_detected_defects	F10	Total number of defects detected by the tester	8, 42.1%	6, 85.7%	14, 53.8%
Version_number	F11	Detail version number of ARs	12, 63.2%	3, 42.9%	15, 57.7%
Developer_rank	F12	Rank of the developer responsible for developing ARs	10, 52.6%	4, 57.1%	14, 53.8%
Contributors	F13	Total number of contributors related to ARs	10, 52.6%	4, 57.1%	14, 53.8%

3.1.3 *Data Analysis.*

To synthesize the findings of this study, quantitative and qualitative methods are applied to investigate what expert features can be used for effort estimation in the enterprise. A total of three researchers participated in the analysis and synthesis of the interview results. We conducted a qualitative coding [16] and thematic analysis [9] to process the recorded interviews using the following steps:

Transcription and coding. We recorded the interviews during the interview process. First, two researchers manually transcribed all interview recordings. Then, they read the manual transcripts and coded the interviews using NVivo qualitative analysis software. To ensure the quality of coding, the third researcher reviewed the coding results and provided suggestions for improvement. After adopting these suggestions, we generated a total of 252 coded cards. We discovered that the codes derived from the interviews were saturated towards the end of the qualitative coding process until no new code appeared, which indicates that the code set is stable. We present two examples below to illustrate the coding process. We originally recorded the following manual transcripts in the interview: Interviewee A mentioned that "When estimating the effort, I usually use the quantitative results of daily work, *i.e.*, requirements, as a means of assessing the output ability of developers." As a result of this, the researchers coded the transcript of the interview as a "requirement (E1)" and recorded the manual transcript. Interviewee B mentioned: "I usually consider the ability of the developer responsible for the allocation of ARs, which is mainly reflected in the historical number of ARs completed by the developer." Again, we coded the manual transcript of the interview as "AR (E2)" and recorded it.

Thematic analysis. Two researchers analyzed the qualitative codes separately and grouped the derived codes into potential themes using thematic synthesis [13]. To reduce bias caused by the two researchers, the third researcher reviewed and finalized the theme set. Any areas of disagreement were discussed until a consensus was reached. The "requirement" mentioned by interviewee A and the "AR" mentioned by interviewee B have the same meaning in this enterprise. During the thematic analysis of the coding cards of interviewees A and B, we integrated the coding results "requirements (E1)" and "AR (E2)" into the same theme, that is, "AR" is an expert feature used by project managers when estimating effort. The data analysis and synthesis phase took around four months, including iterations and revisions. Finally, we derived 13 expert features (as shown in Figure 2) from the interviews that can be used for effort estimation.

3.2 Results of Interviews

Table 2 illustrates the expert features that were obtained through the interviews, and the number of employees who proposed the feature and its percentage (compared to the total number of roles). The units represented by the expert features in Table 2 are per month. For example, *Creator_ARs* indicates the number of ARs created by the creator in one month.

As a result of interviews with 26 employees, we identified 13 expert features that can be used for effort estimation before software development. These are the 13 expert features obtained from the interview, as shown in Table 2: (1) *Creator_ARs* refers to the total number of ARs they have created, which can provide insight into the team's capabilities in organizing their work. Most of the creators in this enterprise are project managers, and a few are designers. (2) *Developer_ARs* is the total number of ARs developed by developers, and it is a reflection of a developer's capacity to develop ARs. (3) *Tester_ARs* indicates the number of ARs to test, it reflects a tester's capacity to test ARs. (4) *Developer_commits* shows the number of commits submitted by developers, which is a feature of the developer's ability to work continuously. Similarity to Developer_commits, (5) *Developer_commits_reviews* refers to the number of reviews of commits completed by developers in history. The review of commits in the enterprise is generally done by experienced developers in the project. (6) *Developer_modified_files* refers to the number of source code files that have been modified by developers. Furthermore, developers' efforts are reflected by the number of merge requests (MRs) that are (7) *Developer_created_MRs* and (8) *Developer_updated_MRs.* (9) *Developer_fixed_defects* refers to the number of defects that were fixed by developers. (10) *Tester_detected_defects* refers to the number of defects detected by testers, which interviewees view as a measure of testers' effort. Additionally, (11) *Version_number* is also considered an effort-related feature. (12) *Developer_rank* refers to the developers' level in the enterprise and reflects a degree of their capabilities. (13) *Contributors* refer to the number of contributors related to ARs.

The manager of an application development project proposed *"the number of ARs developed by a developer to be an indication of the developer's development capability and to be one of the important features used in estimating tasks"*. A total of 16 managers mentioned *Developer_ARs*, accounting for 84.2% of the interviewees. Six out of seven (accounting for 85.7%) software engineers consider the Developer_ARs and Tester_detected_defects to be the main features for effort estimation. A software engineer thinks that *"the number of ARs already developed can be used to estimate new effort. For a tester, the number of defects detected is equally useful for estimating new effort as the number of ARs tests completed"*. There are six engineers who mentioned the above features, accounting for 85.7% of the total number of engineers.

4 APPROACH

To examine the feasibility of our idea, we propose an integrated model, Fine-SE, which integrates semantic features with expert features for effort estimation. As shown in Figure 1, our approach consists of three main steps: (1) feature extraction involves the extraction of expert features and semantic features using the popular pre-trained model BERT, (2) integrated feature learning process is carried out using a fully connected layer, and (3) effort estimation is to estimate the effort of software tasks with previously learned features for both within-project and cross-project scenarios. The following subsections provide details about Fine-SE.

4.1 Features Extraction

The purpose of feature extraction is to convert statistical data and text into numerical representations that can be fitted to neural network models to capture distinguishing characteristics for a later estimation of effort. As a result of its high accuracy, BERT is used for semantic feature extraction in the community [12, 26]. In Fine-SE,

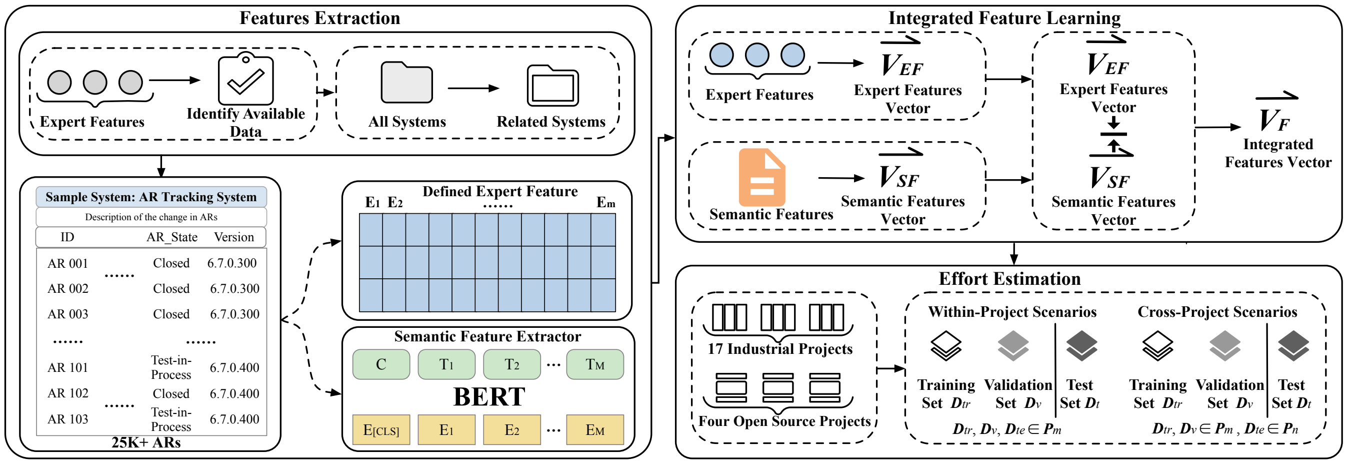

Figure 1: An overview of the proposed approach

the semantic feature extractor takes information (*e.g.*, the title and description of ARs) as input. When performing prediction tasks, the special token, *i.e.*, [CLS], is always inserted at the beginning of the input sequence, followed by the embeddings of the individual words or tokens. This allows BERT to capture important contextual information from the entire sequence [15]. We follow this workaround and add the token "[CLS]" before the title and the description of ARs. After this, the title and the description are labeled as a sequence of tokens, which are fed into a BERT model to generate an embedding vector for software task. Since BERT is designed for general natural language processing tasks, we use solution approaches that have been validated in previous work [58] to fine-tune it so that it can be applied to specific software engineering tasks, that is, effort estimation.

To extract expert features, we directly use 13 expert features derived from 26 semi-structured interviews and process enterprise data of expert features. To obtain data related to expert features, this process primarily involves pre-processing a large amount of enterprise data, including establishing relationships between different online systems, and extraction of related data from data repositories.

4.2 Integrated Feature Learning

Considering that semantic features and expert features are extracted differently, Fine-SE needs to further learn how these features are integrated. There has been no research attempt to integrate both types of features in effort estimation, but in other fields, such as defect prediction and localization [34, 40], there have been some studies that discuss how to integrate them together. Generally, there are two main ways to integrate features in the literature. The first is to align expert features with semantic features and concatenate them into a new feature [40, 57], while the second involves training semantic features in conjunction with expert features [35, 38]. Given the promising results obtained by aligning semantic features with expert features in recent years, we also adopted this way of integration. As noted above, the initial expert feature vector contains only 13 numeric elements extracted from the 13 feature factors, whereas the semantic feature vector generated by BERT contains 768 numeric elements. To prevent semantic features from overwhelming expert features and to treat the two kinds of features equally, we follow Alkhatib's study [6] to integrate semantic features and expert features. Based on their results the high-dimensional representation of the semantic vector is reduced to a low-dimensional representation vector using a fully connected layer. We adopt the same approach in our study to reduce 768-dimensional semantic feature vectors to 13-dimensional vectors by using fully connected layers. After that, the two 13-dimensional vectors are combined to form an integrated vector. The expert feature vector is denoted as V_{EF}, and the semantic vector with a reduced dimension is denoted as V_{SF}. Fine-SE combines the two vectors (V_{EF} and V_{SF}) to create a new one (V_F) which is used to fine-tune during training in conjunction with another fully connective layer.

4.3 Effort Estimation

The last step of Fine-SE is to train a model for estimating effort based on the learned integrated features. This process is achieved by first feeding the learned synthetic features into a hidden layer of the network. Hidden layer output is used as input to the activation function for the estimation of effort. To train the model, all training datasets are iterated over multiple times, the loss function is monitored, and the weights of feature relationships are optimized using back-propagation. Within-project estimation and cross-project estimation are the two scenarios of effort estimation. The former divides the data from the same projects into training, validation, and test sets in accordance with the ratio, while the latter refers to a scenario in which the training, validation, and test datasets originate from different projects.

5 EVALUATION

This section reports on the research questions, the studied datasets, the baselines, and the measures applied for evaluation in our experiments. The OSS dataset and the source code are available online[2].

5.1 Research Questions

The evaluation aims to answer the following research questions regarding the effectiveness of Fine-SE with integrated features on effort estimation:

[2]The OSS datasets and source code for this study are available online through https://github.com/yueli-se/Fine-SE

- **RQ1:** How effective is Fine-SE compared to the existing estimation methods for effort estimation?
  - **RQ1.1:** How effective is Fine-SE compared to the expert estimation and the state-of-the-art models for **within**-project estimation?
  - **RQ1.2:** How effective is Fine-SE compared to the state-of-the-art models for **cross**-project estimation?
- **RQ2:** What is the effect of integrating semantic and expert features on the performance of effort estimation?
  - **RQ2.1:** What is the effect of integrating semantic and expert features on the performance of **within**-project estimation?
  - **RQ2.2:** What is the effect of integrating semantic and expert features on the performance of **cross**-project estimation?

5.1.1 RQ1: Effectiveness of Fine-SE for Effort Estimation. For effort estimation, various advanced approaches have been proposed, using either expert features (*e.g.*, the total number of software tasks) or semantic features (*e.g.*, the title and the description of software tasks) to establish a prediction model in various scenarios (*i.e.*, *within*-project scenarios and *cross*-project scenarios). Fine-SE integrates expert features and semantic features for effort estimation. *Within*-project estimation refers to the process of training, validating, and testing models with the own project's dataset. Therefore, RQ1.1 investigates the effectiveness of Fine-SE in *within*-project estimation.

In the early stages of software development projects, estimating effort may be a difficult task due to the lack of shared knowledge among practitioners [18]. Additionally, Deep Learning approaches for effort estimation may be inaccurate due to the lack of training data. The solution to this is *cross*-project estimation, which involves training a model on other source project(s) and then applying it to the target project. Therefore, RQ1.2 investigates the effectiveness of Fine-SE in *cross*-project estimation.

Experimental Settings: There are three methods selected as the baselines for RQ1.1. In this study, the expert estimation and a pre-trained model (Fine-SE) are compared for the first time. Moreover, we select the state-of-the-art estimation methods Deep-SE and GPT2SP as the baselines. Considering the significant impact of time as proposed by Fu, we follow the same time-aware strategy to build the training data, the validation data, and the testing data from the dataset [18]. 60% of ARs in each project are treated as training data, 20% as validation data, and 20% as testing data. Finally, the training data from each project is combined into the target training data, as well as the validation data and the testing data.

Given no cross-project expert estimation is applied in the enterprise, two baselines (*i.e.*, Deep-SE and GPT2SP) are selected for the *cross*-project estimation (RQ1.2), in which we use the project with the closest number of ARs to the target project as a source project, train the model using the data from the source project, and test the model using the data from the target project. The training, validation, and testing data are collected using the same time-aware strategy as those used in RQ1.1. A total of 75% of ARs from the source project are used for training, 25% of ARs from the source project are used for validation, and 100% of ARs from the target project are used for testing.

5.1.2 RQ2: Effectiveness of Integrated Features. Expert features and semantic features are extracted from different aspects of ARs. Expert features represent ARs that have been derived from the professional knowledge and expert's experience, whereas semantic features represent the intrinsic characteristics of ARs that can be derived from their semantic and syntactic structural contexts. As a result, we explore how the integration of semantic and expert features influences the performance of effort estimation (RQ2). To be specific, we investigate how integrating semantic and expert features affects the performance for *within*-project estimation (RQ2.1), and how integrating semantic and expert features affects the performance for *cross*-project estimation (RQ2.2).

Experimental Settings: We set three training scenarios (*i.e.*, semantic features, expert features, and their integration) to train Fine-SE for assessing the effectiveness of integrated features in effort estimation. The evaluation dataset is set the same as the experiment of RQ1 (within-project estimation uses 60% of ARs for training, 20% for validation, and 20% for testing, while cross-project estimation uses 75% of ARs from the source project for training, 25% for validation, and 100% of ARs from the target project for testing).

5.2 Baselines

To systematically evaluate the performance of Fine-SE, the state-of-the-art models, Deep-SE [11] and GPT2SP [18], based on pre-trained models, have been considered as the important baselines of effort estimation for comparison. Deep-SE is a prediction model that is based on the combination of long short-term memory and recurrent highway networks for estimating story points. Without any manual feature engineering, Deep-SE can be trained from raw input data to prediction outcomes. GPT2SP model utilizes a pre-trained GPT-2 language model and a transformer-based architecture, which enables the model to better capture word relationships by considering the context surrounding each word and its position within the sequence.

In addition, the expert estimation provided by the project managers from the enterprise is also used as a baseline for *within*-project estimation. Expert estimation describes the estimation of effort based on the project manager's own experience for each software task in the enterprise. Furthermore, we use the effort of actual software tasks in the enterprise as the ground truth, specifically, the number of person-months required to complete a software task. The actual effort on the software tasks is recorded in the enterprise, and evaluation results are compared with the actual results.

5.3 Evaluation Measures

The measures that are widely used in effort estimation, *i.e.*, mean absolute error (*MAE*), mean magnitude of relative error (*MMRE*), and performance indicator (*PRED*) [28], are also applied in this evaluation.

MAE is widely used to assess the performance of process models. *MMRE* and *PRED* are the most commonly used measures in effort estimation [27].

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

where $actual_i$, $predicted_i$ refer to the actual effort of ARs and the predicted results, respectively, and n is the number of test instances.

MMRE is used to present the relative amount by which the estimated result is underestimated or overestimated compared to the actual result. *MMRE* is used as an evaluation criterion in research because of its unit-independent properties, such as person-hours, person-days, or person-months.

$$MMRE = \frac{1}{n}\sum_{i}^{n}\left(\frac{|actual_i - predicted_i|}{actual_i}\right) \tag{2}$$

$PRED(50)$ is defined as the proportion of projects with the estimated effort within 50% of the actual effort [24]. $PRED(50)$ are based on the magnitude of relative error measure ($MRE = \frac{|actual_i - predicted_i|}{actual_i}$).

$$PRED(50) = \frac{m}{n} \tag{3}$$

where m represents the number of MRE between the $actual_i$ and the $predicted_i$ that are less than 50%. In terms of MAE and $MMRE$ measures, a lower value is better. When it comes to $PRED(50)$, a higher value is better.

6 RESULTS

6.1 Effectiveness of Fine-SE for Effort Estimation (RQ1)

*6.1.1 Effectiveness of Fine-SE for **Within**-Project Estimation (RQ1.1).* Table 3 presents the evaluation results with the best results highlighted in bold. The results show that our Fine-SE outperforms all the baseline methods on average considered for *within*-project estimation on enterprise projects. For industrial projects, Fine-SE achieves the best results with regard to three evaluation measures: MAE at 0.559, MMRE at 0.497, and PRED(50) at 0.786. When compared with expert estimation, Fine-SE on average achieves 32.0% improvement on MAE, 43.6% improvement on MMRE, and 45.2% improvement on PRED(50). When compared with the state-of-the-art models, Fine-SE on average achieves 41.9%, 27.6%, and 65.2% improvements compared to Deep-SE, and 16.2%, 37.4%, and 91.4% improvements compared with GPT2SP on all the three measures—MAE, MMRE, and PRED(50) respectively.

To further test the differences, Wilcoxon signed-rank test is applied on the three measures. Wilcoxon signed-rank test is a non-parametric hypothesis test used for determining whether results are significantly different between groups, which is used in effort estimation [11], fault localization [10], and other fields [17]. A p-value less than 0.05 indicates that the null hypothesis is rejected, that is, there is no significant difference between the two groups, and therefore, there is a significant difference between the two groups at a 95% confidence level. As shown in Table 3, Fine-SE outperforms the baselines on all the three measures for *within*-project estimation on industrial projects, with a statistically significant difference at a 95% confidence level.

*6.1.2 Effectiveness of Fine-SE for **Cross**-Project Estimation (RQ1.2).* The evaluation results (shown in Table 4 indicate that Fine-SE on average performs the best on all performance measures for *cross*-project estimation on the industrial projects. In particular, Fine-SE achieves 0.940, 0.803, and 0.474 in terms of MAE, MMRE, and PRED (50) respectively on the enterprise projects. When compared with Deep-SE, Fine-SE achieves an average 8.9% improvement on

Table 3: Comparison of Fine-SE with baselines for within-project estimation.

Scenario	Approach	Project	DM	ME	TD	US	Avg. (Imp.)	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12	P13	P14	P15	P16	P17	Avg. (Imp.)
Within-Project	Expert Estimation	MAE	-	-	-	-	-	0.600	0.470	0.580	0.581	0.550	0.867	1.590	0.719	0.956	0.764	0.944	0.806	0.308	1.835	0.580	1.340	0.480	0.822 (-32.0%)
		MMRE	-	-	-	-	-	1.380	0.440	1.530	0.801	0.670	0.568	0.743	0.585	0.695	0.833	3.268	0.434	0.309	0.618	0.930	0.770	0.410	0.881 (-43.6%)
		Pred (50)	-	-	-	-	-	0.370	0.860	0.580	0.600	0.550	0.389	0.567	0.356	0.827	0.464	0.273	0.708	0.875	0.077	0.520	0.400	0.790	0.542 (-45.2%)
	Deep-SE	MAE	4.992	1.398	**4.108**	**0.426**	**2.731(+29.1%)**	0.405	**0.315**	0.766	0.754	0.767	1.586	2.632	1.869	2.297	0.785	0.440	0.614	0.320	0.884	0.355	**1.166**	0.395	0.962 (-41.9%)
		MMRE	3.492	0.509	1.030	**0.105**	1.284(-44.7%)	0.856	0.282	**0.533**	0.688	0.688	1.042	0.895	1.340	0.965	0.730	1.252	0.364	0.312	0.349	0.516	0.613	**0.256**	0.687 (-27.6%)
		Pred (50)	0.336	0.631	**0.438**	**1.000**	**0.601(+9.4%)**	0.582	0.817	0.603	0.239	0.167	0.011	0.067	0.000	0.038	0.071	0.676	0.800	**0.895**	**0.926**	0.685	0.609	0.904	0.476 (-65.2%)
	GPT2SP	MAE	**4.569**	**1.322**	6.424	0.472	3.197(+10.3%)	**0.340**	0.500	0.530	0.570	0.610	0.686	1.899	0.332	0.818	0.504	0.426	0.623	0.387	0.934	**0.320**	1.420	0.440	0.667 (-16.2%)
		MMRE	1.520	0.462	0.766	0.124	0.718(-1.1%)	1.010	0.740	0.610	1.009	1.010	0.829	1.596	0.507	0.507	1.024	**1.033**	0.465	0.418	0.342	1.020	0.820	0.560	0.794 (-37.4%)
		Pred (50)	**0.343**	**0.723**	0.133	**1.000**	0.550(+0.9%)	0.000	0.540	0.580	0.000	0.000	0.511	0.367	0.743	0.731	0.000	0.000	0.792	0.692	0.808	0.000	0.510	0.710	0.411 (-91.4%)
	Fine-SE	MAE*	4.892	1.518	5.542	2.156	3.527	0.400	0.320	**0.390**	**0.445**	**0.390**	**0.663**	**1.486**	**0.316**	**0.686**	**0.325**	**0.385**	**0.607**	**0.305**	**0.749**	0.350	1.300	**0.390**	**0.559**
		MMRE*	**1.053**	**0.448**	**0.692**	0.647	**0.710**	**0.810**	**0.250**	0.760	**0.559**	**0.580**	**0.456**	**0.642**	**0.244**	**0.449**	**0.324**	1.283	**0.322**	**0.302**	**0.243**	**0.490**	**0.480**	0.260	**0.497**
		Pred (50)*	0.292	0.687	0.200	1.000	0.545	**0.590**	**0.940**	**0.850**	**0.714**	**0.770**	**0.644**	**0.633**	**0.871**	**0.865**	**0.786**	**0.697**	**0.917**	0.885	0.885	**0.740**	**0.660**	**0.920**	**0.786**

[1] The lower of MSE and MMRE, the better. For PRED (50), a higher value is better.
[2] The best results on evaluation measures are highlighted in bold.
[3] * Indicates that the p-value of Wilcoxon signed-rank test is less than 0.05 on evaluation measures in industrial projects.

Table 4: Comparison of Fine-SE with baselines for cross-project estimation.

Scenario	Approach	Target	DM	ME	TD	US	Avg. (Imp.)	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12	P13	P14	P15	P16	P17	Avg. (Imp.)
		Source	ME	US	US	TD		P2	P1	P1	P6	P2	P4	P6	P4	P10	P12	P12	P11	P3	P11	P17	P17	P16	
Cross-Project	Deep-SE	MAE	**6.510**	**1.691**	**5.985**	3.332	4.380(-2.2%)	0.577	0.465	**0.772**	1.475	0.489	1.011	2.459	0.507	**1.356**	**0.647**	0.870	1.140	0.507	**2.703**	0.583	**1.272**	0.723	1.033 (-8.9%)
		MMRE	2.079	**0.456**	1.047	1.069	1.163(-49.2%)	1.783	**0.431**	**0.814**	2.581	0.833	1.041	0.671	0.425	0.634	**0.840**	1.732	0.562	**0.625**	**0.692**	1.026	0.734	0.644	0.945(-15.0%)
		Pred (50)	0.342	0.694	**0.324**	0.149	0.377(-18.9%)	0.290	0.719	0.321	0.263	0.551	0.326	0.223	**0.865**	**0.275**	**0.392**	**0.327**	0.378	**0.704**	**0.124**	0.436	**0.511**	0.589	0.429 (-10.5%)
	GPT2SP	MAE	6.515	1.714	6.118	1.723	**4.017(+6.6%)**	0.601	0.604	0.896	0.807	0.630	**0.686**	**2.209**	**0.485**	1.713	0.883	0.969	1.434	0.556	3.116	0.788	1.305	0.493	1.069 (-12.0%)
		MMRE	1.099	0.457	0.984	0.601	0.785(-24.7%)	**1.420**	1.012	1.002	1.285	1.205	0.912	0.681	0.584	0.967	1.429	**1.570**	0.991	0.897	0.982	1.452	0.873	0.650	1.054 (-23.8%)
		Pred (50)	**0.351**	**0.715**	0.315	0.490	**0.468(+4.2%)**	0.105	0.000	0.000	0.405	0.262	**0.548**	0.224	0.678	0.000	0.368	0.317	0.000	0.525	0.000	0.237	0.325	0.695	0.276 (-71.9%)
	Fine-SE	MAE*	6.726	2.149	7.534	**0.720**	4.282	**0.572**	**0.456**	0.774	**0.557**	**0.426**	0.709	2.425	0.505	1.361	0.837	**0.817**	**1.114**	**0.473**	2.825	**0.480**	1.276	**0.380**	**0.940**
		MMRE*	**0.883**	0.466	**0.821**	**0.194**	**0.591**	1.757	0.434	0.838	**1.023**	**0.720**	**0.737**	**0.669**	**0.415**	**0.609**	1.123	1.618	**0.550**	0.690	0.721	**0.802**	**0.698**	**0.251**	**0.803**
		Pred (50)	0.208	0.563	0.082	**0.940**	0.448	**0.293**	**0.755**	**0.357**	**0.540**	**0.589**	0.532	**0.231**	0.855	0.241	0.278	0.323	**0.407**	0.684	0.086	**0.498**	0.510	**0.880**	**0.474**

Table 5: Comparison with different features for within-project estimation.

Scenario	Approach	Project	DM	ME	TD	US	Avg. (Imp.)	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12	P13	P14	P15	P16	P17	Avg. (Imp.)
Within-Project	EF	MAE	5.141	2.798	7.397	2.395	4.433(-20.4%)	0.600	0.470	0.580	0.581	0.550	0.867	1.590	0.719	0.956	0.764	0.618	1.462	1.697	3.087	0.607	2.994	0.918	1.121 (-56.2%)
		MMRE	**0.908**	0.730	1.039	0.726	0.851(-16.5%)	1.380	0.440	1.530	0.801	0.670	0.568	0.743	0.585	0.695	0.833	**0.826**	0.721	1.581	1.056	0.678	1.758	0.617	0.911 (-41.3%)
		Pred (50)	0.278	0.096	0.000	0.000	0.094(-482.6%)	0.370	0.860	0.580	0.600	0.550	0.389	0.567	0.356	0.827	0.464	0.152	0.167	0.000	0.000	0.400	0.000	0.259	0.385 (-40.2%)
	SF	MAE	**4.863**	**1.348**	**4.904**	**0.407**	**2.881(+22.4%)**	0.410	**0.290**	0.460	**0.443**	**0.390**	0.670	1.983	0.327	0.714	0.332	0.401	**0.590**	0.519	0.837	0.360	1.480	**0.390**	0.623 (-6.4%)
		MMRE	3.688	0.475	0.704	**0.105**	1.243(-42.9%)	0.820	0.270	0.990	0.566	0.610	0.481	1.429	0.249	0.470	0.329	1.372	0.333	0.919	0.332	0.530	0.880	**0.250**	0.637 (-14.0%)
		Pred (50)	**0.307**	0.651	**0.400**	0.950	**0.577(+8.7)**	**0.590**	0.880	0.760	0.705	0.730	0.633	0.400	0.822	**0.865**	0.768	0.636	0.875	0.510	0.846	0.680	0.270	0.910	0.699 (-8.7%)
	EF+SF	MAE*	4.892	1.518	5.542	2.156	3.527	**0.400**	0.320	**0.390**	0.445	0.390	**0.663**	**1.486**	**0.316**	**0.686**	**0.325**	**0.385**	0.607	**0.305**	**0.749**	**0.350**	**1.300**	0.390	**0.559**
		MMRE*	1.053	**0.448**	**0.692**	0.647	**0.710**	**0.810**	**0.250**	**0.760**	**0.559**	**0.580**	**0.456**	**0.642**	**0.244**	**0.449**	**0.324**	1.283	**0.322**	**0.302**	**0.243**	**0.490**	**0.480**	0.260	**0.497**
		Pred (50)*	0.292	**0.687**	0.200	**1.000**	0.545	0.590	**0.940**	**0.850**	**0.714**	**0.770**	**0.644**	**0.633**	**0.871**	0.865	**0.786**	**0.697**	**0.917**	**0.885**	**0.885**	**0.740**	**0.660**	**0.920**	**0.786**

[1] EF refers to estimation using only expert features, SF refers to using only semantic features, and EF+SF refers to using both expert and semantic features.

Table 6: Comparison with different features for cross-project estimation.

Scenario	Approach	Target	DM	ME	TD	US	Avg. (Imp.)	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12	P13	P14	P15	P16	P17	Avg. (Imp.)
		Source	ME	US	US	TD		P2	P1	P1	P6	P2	P4	P6	P4	P10	P12	P12	P11	P3	P11	P17	P17	P16	
Cross-Project	EF	MAE	7.167	4.085	**5.385**	5.518	5.539(-22.7%)	1.631	0.863	**0.769**	1.266	0.840	2.131	**1.401**	1.735	**1.209**	1.506	1.672	1.192	0.745	**2.305**	0.958	1.285	2.280	1.399 (-45.9%)
		MMRE	0.888	1.099	0.989	1.043	1.005（-41.2%）	2.394	0.993	**0.784**	**0.902**	0.785	1.201	**0.628**	1.234	0.655	1.227	**1.315**	0.807	0.708	1.176	0.803	**0.678**	1.532	1.048 (-24.5%)
		Pred (50)	0.108	0.223	0.089	0.011	0.027（-316.0%)	0.000	0.068	0.206	0.042	0.074	0.000	**0.296**	0.001	0.228	0.008	0.000	0.121	0.364	0.010	0.050	0.290	0.001	0.103 (-37.1%)
	SF	MAE	**6.526**	**1.684**	6.010	2.650	**4.218(+1.5%)**	**0.521**	**0.432**	0.824	0.647	0.461	1.360	2.342	**0.501**	1.343	**0.800**	0.931	1.154	0.517	2.829	0.537	**1.257**	0.480	0.996 (-5.6%)
		MMRE	2.111	**0.466**	1.087	0.945	1.152(-48.7%)	**1.639**	**0.410**	0.813	1.210	0.795	1.442	0.656	0.426	0.617	**1.065**	1.844	0.564	0.773	**0.720**	0.928	0.702	0.417	0.884 (-8.0%)
		Pred (50)	**0.335**	**0.676**	**0.315**	0.140	0.367(-22.1%)	**0.345**	**0.775**	0.289	0.444	0.576	0.188	0.279	**0.873**	**0.319**	**0.278**	0.250	0.356	0.585	0.078	0.481	**0.524**	0.723	0.433 (-4.1%)
	EF+SF	MAE	6.726	2.149	7.534	**0.720**	4.282	0.572	0.456	0.774	**0.557**	**0.426**	**0.709**	2.425	0.505	1.361	0.837	**0.817**	**1.114**	**0.473**	2.825	**0.480**	1.276	**0.380**	**0.940**
		MMRE	**0.883**	**0.466**	**0.821**	**0.194**	**0.591**	1.757	0.434	0.838	1.023	**0.720**	**0.737**	0.669	**0.415**	**0.609**	1.123	1.618	**0.550**	**0.690**	0.721	**0.802**	0.698	**0.251**	**0.803**
		Pred (50)	0.208	0.563	0.082	**0.940**	**0.448**	0.293	0.755	**0.357**	**0.540**	**0.589**	**0.532**	0.231	0.855	0.241	0.278	**0.323**	**0.407**	**0.684**	**0.086**	**0.498**	0.510	**0.880**	**0.474**

MAE, 15.0% improvement on MMRE, and 10.5% improvement on Pred(50). Compared to GPT2SP, Fine-SE achieves an average 12.0% improvement on MAE, 23.8% improvement on MMRE, and 71.9% improvement on Pred(50).

Again Wilcoxon signed-rank test is applied on the three measures to test the differences. As shown in Table 4, Fine-SE outperforms Deep-SE and GPT2SP on MAE and MMRE of the Wilcoxon signed-rank test in enterprise projects. The results of the Wilcoxon signed-rank test show that Fine-SE has a statistically significant difference from the baselines.

Answer to RQ1: Fine-SE outperforms the expert estimation and the state-of-the-art models for both within-project estimation and cross-project estimation in the industrial projects, especially achieving the overwhelming results on MAE and MMRE.

6.2 Effectiveness of Integrated Features (RQ2)

*6.2.1 Effectiveness of Integrated Features for **Within**-project Estimation (RQ2.1).* Table 5 presents the comparisons with the best results highlighted in bold for each approach according to the following three different settings: EF refers to using only Expert Features, SF means using only Semantic Features, and EF+SF refers to using both expert and semantic features. In industrial projects, compared to the single expert feature (EF), the integrated feature (EF+SF) improves on average by 56.2% on MAE, 41.3% on MMRE, and 40.2% on Pred(50), while compared to the single semantic feature (SF), the integrated feature EF+SF improves on average by 6.4% on MAE, 14.0% on MMRE, and 8.7% on Pred(50).

Table 5 shows the results of Wilcoxon signed-rank test on the three measures, which indicate that Fine-SE outperforms the baselines on all these measures for *within*-project estimation in the enterprise projects.

*6.2.2 Effectiveness of Integrated Features for **Cross**-Project Estimation (RQ2.2).* The results of this comparison are presented in Table 6, with the best performing approach for each project highlighted in bold in *cross*-project estimation. According to the evaluation results, Fine-SE produces significant improvements across industrial projects. When comparing the integrated feature (EF+SF) with the single expert feature (EF), the integrated feature improved MAE by 45.9%, MMRE by 24.5%, and Pred(50) by 37.1% on average. When compared to the single semantic feature (SF), the integrated feature (EF+SF) has an average improvement of 5.6% on MAE, 8.0% on MMRE, and 4.1% on Pred(50).

Based on the results, the following conclusions can be drawn: 1) Expert and semantic features have their own respective advantages when building estimation models. 2) Semantic features appear to offer a better understanding of the characteristics of software effort than expert features. 3) Fine-SE can benefit from both expert features and semantic features by utilizing them to achieve higher performance for both *within*-project estimation and *cross*-project estimation.

Answer to RQ2: Expert features and semantic features both have advantages in estimating effort. Fine-SE is able to achieve better results by integrating both types of features to reinforce the effort estimation than the model fed with single features.

7 DISCUSSION

To the best of our knowledge, this is the first study that leverages expert features and semantic features for effort estimation based on a large-scale enterprise dataset. This section discusses the implications of this study for researchers and practitioners.

7.1 Implications for Researchers

Expert features: The expert features are proposed by project managers with rich working experience, and are extracted from the project historical data in the enterprise. The estimation with expert features combines expert knowledge and historical data. The evaluation results shown in Table 5 provide the evidence that by integrating expert features, the model has an average improvement of 6.4-14.0% on evaluation measures compared with using semantic features alone. **Semantic features**: The extraction of semantic features is performed by powerful pre-trained models. Compared to expert features, semantic features perform better for effort estimation across multiple projects.

According to Table 5 and Table 6, the integrated features improve the measures by 4.1% to 56.2% compared with single expert features and single semantic features on the enterprise projects. This study provides researchers with an initial reference for improving the accuracy of effort estimation by leveraging expert features and semantic features.

In industrial scenarios, due to the diversity of teams and data, effort estimation remains a challenging task. Figure 2 illustrates the differences in ARs estimated per month for different projects. It is common that project managers are required to estimate more than 40 ARs per month on average, and in some extreme cases, project managers are required to estimate more than 120 ARs per month. Due to this, effort estimation remains difficult in industry, and even expert estimation is less effective. Table 3 illustrates that as the estimated ARs increase, the expert estimation suffers from performance degradation, whereas the automated method can remain relatively stable and perform well. Accordingly, we suggest that researchers can explore more effective methods for effort estimation. The findings of this study reveal that integrating expert features and semantic features is observed as a potential strategy for effort estimation.

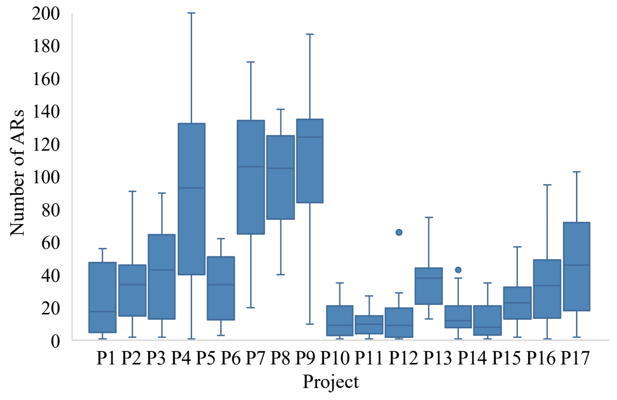

Figure 2: Monthly #ARs of the projects

7.2 Implications for Practitioners

The evaluation results of expert estimation indicate that estimating effort remains a challenging task in the enterprise. From practitioner's perspective, although several project managers in the enterprise have more than ten years of experience in the industry, the expert estimations still suffer poor performance according to the evaluation results. As the amount of data increases, expert estimation may become less effective, whereas the automated methods (*e.g.*, Fine-SE) remain relatively stable and provide decent results (as shown in Table 3). In this sense, we recommend that project managers could use automated methods, such as Fine-SE, to assist them in estimating effort.

Additionally, we invited project managers to check their estimation results, The project manager confirmed their estimation results and agreed that their estimates were indeed inaccurate. After consulting with the project manager, we discovered that subjective misestimation is one of the main causes of inaccurate estimates. In the enterprise, project managers sometimes may submit an estimate with a smaller size than normal to the online system, so as to enforce a higher delivery pressure on the developers involved. The impact may not be accurately captured by semantic and expert features, since it might be also influenced by the other managerial considerations out of estimation. Model-based estimations may be less effective if the project manager makes inaccurate estimations on purpose.

Several project managers do not use online systems to manage software tasks in the enterprise, but instead use local documents. In the enterprise, the use of local documents is a common practice. These managers consider that localized documents are more flexible and not limited by the rules of online systems when managing software tasks. Model-based estimation may be more challenging in projects that use local documents to manage tasks. For instance, change logs cannot be recorded in local documents, making it impossible to verify the accuracy of the data later on. In addition, since different teams use different table structures to manage tasks in the enterprise, estimation models need to be adapted to different documents. It is recommended that project managers use online systems to manage software tasks instead of localized documents in order to facilitate the traceability of data and the introduction of automated tools.

8 THREATS TO VALIDITY

In this section, we discuss the potential threats to the validity of the study and our efforts to mitigate their impacts.

Internal validity. The internal threat arises from potential mistakes in the implementation of our approach and the baselines. To mitigate this threat, we used the original source code shared by the baselines [11, 18], as well as the hyperparameter settings used in the baselines. Moreover, we invited multiple experts to jointly check the implementations to ensure their validity.

External validity. The external validity of this study is primarily concerned with the ability to generalize our research to the external. In this study, the 17 enterprise projects and expert features are acquired from one enterprise, which poses a threat to external validity. The main structure of Fine-SE is based on BERT, which is widely used in many studies. Fine-SE was implemented and evaluated on 17 enterprise projects and four OSS projects. We obtained expert features from different perspectives (*i.e.*, project managers, developers, and testers) and lines of business (*e.g.*, platform development) in order to mitigate the impact of context on them.

Construct validity. The main threat to construct validity is posed by the evaluation measures we select, that is, ensuring that the measures selected can be matched with the approach and design of the study in order to produce reliable results. To minimize this threat, we use the most commonly used measures in effort estimation, *i.e.*, MMRE and Pred (50) [27], as well as MAE, which has received increased attention in recent years.

9 RELATED WORK

Effort estimation has attracted substantial attention from SE researchers since reliable effort estimation can benefit companies on several levels, *e.g.*, identifying the resources and project scope, monitoring project progress, and improving software quality [25, 43].

Machine learning is commonly used in a variety of fields, including effort estimation, since it can acquire and integrate knowledge from large-scale observations and be improved by learning new information [42]. The first research on effort estimation using ML was published in 2011 by Abrahamsson et al. [2]. They construct a predictor that extracts the number of characters and the occurrences of the 15 most frequently used keywords. Support Vector Machine (SVM) presents the best results in two enterprise agile teams using stories to write in a structured way. After that, Porru et al. [41] and Scott et al. [47] estimated effort for OSS projects by using Term Frequency Inverse Document Frequency (TFIDF) derived from issue descriptions and SVM, respectively. The results show that both the classifier and the SVM estimator outperform the baselines.

Deep learning methods significantly improve the state-of-the-art in many domains such as object recognition and prediction, and have been gradually applied in effort estimation [11]. To improve the stability of effort estimation, Pasuksmit [37] conducted an empirical study of mixed-methods to investigate the potential impacts of unstable effort, in which deep learning methods were used to identify information about change. Panda et al. [36] compared different performances of Neural Networks for effort estimation (e.g., General Regression Neural Network and Cascade-Correlation Neural Network). The experiment results that the cascade network outperformed other networks. Ahmad et al. [3] evaluated the effectiveness of Long Short Term Memory networks (LSTM) methods on a Chinese dataset with 499 projects and the Kitchenham dataset with 145 projects. The results show that LSTM achieves the best results on both datasets. Pre-trained models become landmarks in artificial intelligence and have achieved great success in many fields [20], including effort estimation [11, 14]. Choetkiertikul et al. [11] propose a pre-trained model called Deep-SE for estimating effort using two powerful deep learning architectures that are LSTM and recurrent highway network. Empirical studies on 16 OSS projects show that their models outperform the baselines. Following that, Tawosi et al. [51] conducted a replication study on Deep-SE and compared it to TFIDF, and the results show that the Deep-SE did not outperform TFIDF-SE statistically. In recent years, Fávero et al. [14] constructed a generic pre-trained model that is fine-tuned for context-less and contextualized approaches. They used the generated models as inputs in the applied deep learning architecture. The results indicate that the pre-trained model with fine-tuned achieves better performance than the baseline models

In contrast to previous studies, this paper obtains expert features through the interviews that can be used to estimate effort. In addition, this paper is the first attempt to combine expert and semantic features to enhance effort estimation.

10 CONCLUSION

In this paper, we propose an integrated model, Fine-SE, which leverages expert features and semantic features to construct an automatic integrated model for effort estimation. Based on the interviews conducted within a global ICT enterprise, we proposed comprehensive expert features that could be used to estimate the effort in advance of development. To verify the effectiveness of Fine-SE, we compare it with the expert estimation and the state-of-the-art models (*i.e.*, Deep-SE and GPT2SP) on 17 projects in the enterprise. The evaluation results demonstrate the effectiveness of our proposed method, as we achieve better results than expert estimation. As compared to the state-of-the-art baseline methods, ours show an improvement of 16.2%-91.4% on *within*-project estimation, and an improvement of 8.9%-71.9% on *cross*-project estimation.

ACKNOWLEDGMENTS

This work is supported by the National Natural Science Foundation of China (No.62072227, No.62202219, No.62302210), the Key Research and Development Program of Jiangsu Province (No.BE2021002-2), the National Key Research and Development Program of China (No.2019YFE0105500), and the Innovation Project and Overseas Open Project of State Key Laboratory for Novel Software Technology (Nanjing University) (ZZKT2022A25, KFKT2022A09).

REFERENCES

[1] Tarek K. Abdel-Hamid. 1990. On the Utility of Historical Project Statistics for Cost and Schedule Estimation: Results from a Simulation-Based Case Study. *Journal of Systems and Software* 13, 1 (1990), 71–82.

[2] Pekka Abrahamsson, Ilenia Fronza, Raimund Moser, Jelena Vlasenko, and Witold Pedrycz. 2011. Predicting Development Effort from User Stories. In *Proceedings of the 2011 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM '11)*. IEEE, Banff, AB, Canada, 400–403.

[3] Farah B. Ahmad and Laheeb M. Ibrahim. 2022. Software Development Effort Estimation Techniques Using Long Short Term Memory. In *Proceedings of the 17th International Conference on Computer Science and Software Engineering (CSASE'20)*. IEEE, Duhok, Kurdistan Region, Iraq, 182–187.

[4] Maha Al-Yahya, Hend Al-Khalifa, Alia Bahanshal, and Iman Al-Oudah. 2011. Automatic Generation of Semantic Features and Lexical Relations Using OWL Ontologies. In *Proceedings of the 16th International Conference on Natural Language Processing and Information Systems (NLDB'11)*. Springer-Verlag, Alicante, Spain, 15–26.

[5] Mohammed Alhamed and Tim Storer. 2021. Playing Planning Poker in Crowds: Human Computation of Software Effort Estimates. In *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE '21)*. IEEE, Madrid, Spain, 1–12.

[6] Wael Alkhatib, Christoph Rensing, and Johannes Silberbauer. 2017. Multi-label Text Classification Using Semantic Features and Dimensionality Reduction with Autoencoders. In *Proceedings of the 1st Language, Data, and Knowledge (LDK '17)*, Jorge Gracia, Francis Bond, John P. McCrae, Paul Buitelaar, Christian Chiarcos, and Sebastian Hellmann (Eds.). Springer, Galway, Ireland, 380–394.

[7] Ayman Jalal Hassan Almutlaq and Dayang N. A. Jawawi. 2020. Missing Data Imputation Techniques for Software Effort Estimation: A Study of Recent Issues and Challenges. In *Emerging Trends in Intelligent Computing and Informatics*, Faisal Saeed, Fathey Mohammed, and Nadhmi Gazem (Eds.). Springer, Johor, Malaysia, 1144–1158.

[8] Ilona Bluemke and Agnieszka Malanowska. 2021. Software Testing Effort Estimation and Related Problems: A Systematic Literature Review. *Comput. Surveys* 54, 3, Article 53 (apr 2021), 38 pages.

[9] Virginia Braun and Victoria Clarke. 2006. Using thematic analysis in psychology. *Qualitative Research in Psychology* 3, 2 (2006), 77–101.

[10] An Ran Chen, Tse-Hsun (Peter) Chen, and Shaowei Wang. 2021. Demystifying the Challenges and Benefits of Analyzing User-Reported Logs in Bug Reports. *Empirical Software Engineering* 26, 1 (2021), 1–30.

[11] Morakot Choetkiertikul, Hoa Khanh Dam, Truyen Tran, Trang Pham, Aditya Ghose, and Tim Menzies. 2019. A Deep Learning Model for Estimating Story Points. *IEEE Transactions on Software Engineering* 45, 7 (2019), 637–656.

[12] Agnieszka Ciborowska and Kostadin Damevski. 2022. Fast Changeset-Based Bug Localization with BERT. In *Proceedings of the 44th International Conference on Software Engineering (ICSE'22) (ICSE '22)*. ACM, Pittsburgh, Pennsylvania, 946–957.

[13] Daniela S. Cruzes and Tore Dyba. 2011. Recommended Steps for Thematic Synthesis in Software Engineering. In *Proceedings of the 2011 International Symposium*

on Empirical Software Engineering and Measurement (ESEM '11). IEEE, Alberta, Canada, 275–284.

[14] Eliane Maria De Bortoli Fávero, Dalcimar Casanova, and Andrey Ricardo Pimentel. 2022. SE3M: A model for software effort estimation using pre-trained embedding models. *Information and Software Technology* 147 (2022).

[15] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *CoRR* (2018), 1–17.

[16] Kerry Dhakal. 2022. NVivo. *Journal of the Medical Library Association* 110, 2 (2022), 270–272.

[17] Jinhao Dong, Yiling Lou, Qihao Zhu, Zeyu Sun, Zhilin Li, Wenjie Zhang, and Dan Hao. 2022. FIRA: Fine-Grained Graph-Based Code Change Representation for Automated Commit Message Generation. In *Proceedings of the 44th International Conference on Software Engineering (ICSE'22)*. IEEE, Pittsburgh, PA, USA, 970–981.

[18] Michael Fu and Chakkrit Tantithamthavorn. 2023. GPT2SP: A Transformer-Based Agile Story Point Estimation Approach. *IEEE Transactions on Software Engineering* 49, 2 (2023), 611–625.

[19] The Standish Group. 2020. *The CHAOS report*.

[20] Xu Han, Zhengyan Zhang, Ning Ding, Yuxian Gu, Xiao Liu, Yuqi Huo, Jiezhong Qiu, Yuan Yao, Ao Zhang, Liang Zhang, Wentao Han, Minlie Huang, Qin Jin, Yanyan Lan, Yang Liu, Zhiyuan Liu, Zhiwu Lu, Xipeng Qiu, Ruihua Song, Jie Tang, Ji-Rong Wen, Jinhui Yuan, Wayne Xin Zhao, and Jun Zhu. 2021. Pre-trained models: Past, present and future. *AI Open* 2 (2021), 225–250.

[21] Mohamed Hosni, Ali Idri, and Alain Abran. 2021. On the value of filter feature selection techniques in homogeneous ensembles effort estimation. *Journal of Software: Evolution and Process* 33, 6 (2021), 1–38.

[22] Magne Jorgensen. 2004. A review of studies on expert estimation of software development effort. *Journal of Systems and Software* 70, 1 (2004), 37–60.

[23] Magne Jorgensen and Martin Shepperd. 2007. A Systematic Review of Software Development Cost Estimation Studies. *IEEE Transactions on Software Engineering* 33, 1 (2007), 33–53.

[24] Magne Jørgensen. 2019. Evaluating probabilistic software development effort estimates: Maximizing informativeness subject to calibration. *Information and Software Technology* 115 (2019), 93–96.

[25] Rahul Krishna, Amritanshu Agrawal, Akond Rahman, Alexander Sobran, and Tim Menzies. 2018. What is the Connection between Issues, Bugs, and Enhancements? Lessons Learned from 800+ Software Projects. In *Proceedings of the 40th International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP '18)*. ACM, Gothenburg, Sweden, 306–315.

[26] Jinfeng Lin, Yalin Liu, Qingkai Zeng, Meng Jiang, and Jane Cleland-Huang. 2021. Traceability Transformed: Generating More Accurate Links with Pre-Trained BERT Models. In *Proceedings of the 43th International Conference on Software Engineering (ICSE'21)*. IEEE, Madrid, ES, 324–335.

[27] Yasir Mahmood, Nazri Kama, and Azri Azmi. 2020. A systematic review of studies on use case points and expert-based estimation of software development effort. *Journal of Software: Evolution and Process* 32, 7 (2020), 1–20.

[28] Yasir Mahmood, Nazri Kama, Azri Azmi, Ahmad Salman Khan, and Mazlan Ali. 2022. Software effort estimation accuracy prediction of machine learning techniques: A systematic performance evaluation. *Software: Practice and Experience* 52, 1 (2022), 39–65.

[29] Viljan Mahnič and Tomaž Hovelja. 2012. On using planning poker for estimating user stories. *Journal of Systems and Software* 85, 9 (2012), 2086–2095.

[30] Emilia Mendes. 2012. Using Knowledge Elicitation to Improve Web Effort Estimation: Lessons from Six Industrial Case Studies. In *Proceedings of the 34th International Conference on Software Engineering (ICSE'12)*. IEEE, Zurich, Switzerland, 1112–1121.

[31] Tim Menzies, Andrew Butcher, David Cok, Andrian Marcus, Lucas Layman, Forrest Shull, Burak Turhan, and Thomas Zimmermann. 2013. Local versus Global Lessons for Defect Prediction and Effort Estimation. *IEEE Transactions on Software Engineering* 39, 6 (2013), 822–834.

[32] Tim Menzies, Ye Yang, George Mathew, Barry Boehm, and Jairus Hihn. 2017. Negative Results for Software Effort Estimation. *Empirical Software Engineering* 22, 5 (2017), 2658–2683.

[33] K. Moløkken and M. Jorgensen. 2003. A review of software surveys on software effort estimation. In *Proceedings of 2003 International Symposium on Empirical Software Engineering (ISESE '03)*. IEEE, Rome, Italy, 223–230.

[34] Chao Ni, Wei Wang, Kaiwen Yang, Xin Xia, Kui Liu, and David Lo. 2022. The Best of Both Worlds: Integrating Semantic Features with Expert Features for Defect Prediction and Localization. In *Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE '22)*. ACM, Singapore, Singapore, 672–683.

[35] Shengyi Pan, Lingfeng Bao, Xiaoxue Ren, Xin Xia, David Lo, and Shanping Li. 2021. Automating Developer Chat Mining. In *Proceedings of the 36th IEEE/ACM International Conference on Automated Software Engineering (ASE '21)*. IEEE, Melbourne, Australia, 854–866.

[36] Aditi Panda, Shashank Mouli Satapathy, and Santanu Kumar Rath. 2015. Empirical Validation of Neural Network Models for Agile Software Effort Estimation based on Story Points. *Procedia Computer Science* 57 (2015), 772–781.

[37] Jirat Pasuksmit. 2021. Investigating Documented Information for Accurate Effort Estimation in Agile Software Development. In *Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE'21)*. ACM, Athens, Greece, 1605–1609.

[38] Rahul Paul, Samuel H. Hawkins, Yoganand Balagurunathan, Matthew Schabath, Robert J. Gillies, Lawrence O. Hall, and Dmitry B. Goldgof. 2016. Deep Feature Transfer Learning in Combination with Traditional Features Predicts Survival among Patients with Lung Adenocarcinoma. *Tomography* 2, 4 (2016), 388–395.

[39] Passakorn Phannachitta. 2020. On an optimal analogy-based software effort estimation. *Information and Software Technology* 125 (2020), 1–11.

[40] Chanathip Pornprasit and Chakkrit Kla Tantithamthavorn. 2021. JITLine: A Simpler, Better, Faster, Finer-grained Just-In-Time Defect Prediction. In *Proceedings of the IEEE/ACM 18th International Conference on Mining Software Repositories (MSR '21)*. IEEE, Madrid, Spain, 369–379.

[41] Simone Porru, Alessandro Murgia, Serge Demeyer, Michele Marchesi, and Roberto Tonelli. 2016. Estimating Story Points from Issue Reports. In *Proceedings of the 12th International Conference on Predictive Models and Data Analytics in Software Engineering (PROMISE '16)*. ACM, Ciudad Real, Spain, Article 2, 10 pages.

[42] Przemyslaw Pospieszny, Beata Czarnacka-Chrobot, and Andrzej Kobylinski. 2018. An effective approach for software project effort and duration estimation with machine learning algorithms. *Journal of Systems and Software* 137 (2018), 184–196.

[43] R. C. Sandeep, Mary Sánchez-Gordón, Ricardo Colomo-Palacios, and Monica Kristiansen. 2022. Effort Estimation in Agile Software Development: A Exploratory Study of Practitioners' Perspective. In *Proceedings of 6th International Conference on Lean and Agile Software Development (LASD '22)*, Adam Przybyłek, Aleksander Jarzębowicz, Ivan Luković, and Yen Ying Ng (Eds.). Springer, Gdańsk, Poland, 136–149.

[44] Federica Sarro, Rebecca Moussa, Alessio Petrozziello, and Mark Harman. 2022. Learning From Mistakes: Machine Learning Enhanced Human Expert Effort Estimates. *IEEE Transactions on Software Engineering* 48, 6 (2022), 1868–1882.

[45] Federica Sarro and Alessio Petrozziello. 2018. Linear Programming as a Baseline for Software Effort Estimation. *ACM Transactions on Software Engineering and Methodology* 27, 3 (2018), 1–28.

[46] Federica Sarro, Alessio Petrozziello, and Mark Harman. 2016. Multi-Objective Software Effort Estimation. In *Proceedings of the 38th International Conference on Software Engineering (ICSE '16)*. ACM, Austin, Texas, 619–630.

[47] Ezequiel Scott and Dietmar Pfahl. 2018. Using Developers' Features to Estimate Story Points. In *Proceedings of the 2018 International Conference on Software and System Process (ICSSP'18)*. ACM, Gothenburg, Sweden, 106–110.

[48] Yeong-Seok Seo, Kyung-A Yoon, and Doo-Hwan Bae. 2008. An Empirical Analysis of Software Effort Estimation with Outlier Elimination. In *Proceedings of the 4th International Workshop on Predictor Models in Software Engineering (PROMISE '08)*. ACM, Leipzig, Germany, 25–32.

[49] Martin Shepperd and Steve MacDonell. 2012. Evaluating Prediction Systems in Software Project Estimation. *Information and Software Technology* 54, 8 (2012), 820–827.

[50] Martin John Shepperd and Chris Schofield. 1997. Estimating software project effort using analogies. *IEEE Transactions on Software Engineering* 23, 11 (1997), 736–743.

[51] Vali Tawosi, Rebecca Moussa, and Federica Sarro. 2022. Deep Learning for Agile Effort Estimation Have We Solved the Problem Yet? *IEEE Transactions on Software Engineering* (2022), 1–17.

[52] Muhammad Usman, Ricardo Britto, Lars-Ola Damm, and Jürgen Börstler. 2018. Effort estimation in large-scale software development: An industrial case study. *Information and Software Technology* 99 (2018), 21–40.

[53] Muhammad Usman, Emilia Mendes, Francila Weidt, and Ricardo Britto. 2014. Effort Estimation in Agile Software Development: A Systematic Literature Review. In *Proceedings of the 10th International Conference on Predictive Models in Software Engineering (PROMISE '14)*. ACM, Turin, Italy, 82–91.

[54] Jianfeng Wen, Shixian Li, Zhiyong Lin, Yong Hu, and Changqin Huang. 2012. Systematic literature review of machine learning based software development effort estimation models. *Information and Software Technology* 54, 1 (2012), 41–59.

[55] Peter A. Whigham, Caitlin A. Owen, and Stephen G. Macdonell. 2015. A Baseline Model for Software Effort Estimation. *ACM Transactions on Software Engineering and Methodology* 24, 3 (2015), 11 pages.

[56] Tianpei Xia, Rui Shu, Xipeng Shen, and Tim Menzies. 2022. Sequential Model Optimization for Software Effort Estimation. *IEEE Transactions on Software Engineering* 48, 6 (2022), 1994–2009.

[57] Meng Yan, Xin Xia, Yuanrui Fan, Ahmed E. Hassan, David Lo, and Shanping Li. 2022. Just-In-Time Defect Identification and Localization: A Two-Phase Framework. *IEEE Transactions on Software Engineering* 48, 1 (2022), 82–101.

[58] Sarim Zafar, Muhammad Zubair Malik, and Gursimran Singh Walia. 2019. Towards Standardizing and Improving Classification of Bug-Fix Commits. In *Proceedings of the 2019 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM '19)*. IEEE, Porto de Galinhas, Brazil, 1–6.