JOSSE: A Software Development Effort Dataset Annotated with Expert Estimates

First Author
Department
Institution
City, Country
email@host.com

Second Author
Department
Institution
City, Country
email@host.com

***Abstract*—The JIRA Open-Source Software Effort (JOSSE) dataset consists of software development and maintenance tasks collected from the JIRA issue tracking system for Apache, JBoss, And Spring open-source projects. All the issues annotated with actual effort and 19% of them annotated with expert estimates. JOSSE is a task-based dataset with a textual attribute represented as a task description for each data point. This paper explains how the data were collected and details six data quality refinement procedures of the data points.**

I. Introduction

Datasets of software task efforts are essential for the empirical evaluation of novel methods of software effort estimation (SEE). This requirement is particularly critical for methods based on Machine Learning (ML), since such methods typically require large amounts of data for training and evaluation. Researchers may develop their own datasets to suit their specific needs, or reuse datasets that have been published in association with previous studies. According to Machine Learning SEE surveys [30, 3, 4, 16, 28], the top five datasets are COCOMO [6], Desharnais [13], NASA [21], ISBSG [1], and Zia's dataset [31]. The PROMISE repository [27] keeps some more of those datasets publicly available (20 were listed on the date of submission). Typical attributes of task items in these data sets include total lines of code (LOC), team experience, project length and actual effort. Aside from reducing the cost of experimentation, reuse of existing datasets also eases the process of comparison between SEE methods.

One limitation of these existing popular data sets is a lack of textual data, such as titles and descriptions for task items. This makes it impossible to use these datasets to evaluate methods based on the features. Other datasets that are less frequently used in the literature, such as Deep-SE [10] and Porru's [25] do include textual attributes. However, these datasets only contain actual effort reports that can be used as a ground truth and lack expert estimates that may be used as a baseline.

In addition, several studies [29, 7, 9, 5, 19, 24] have noted the lack of consideration given to the assessment of SEE dataset quality. One aspect here may be the lack of guidance or methods for assessing the suitability of datasets for evaluation of effort estimation methods. However, Kitchenham and Mendes [19] have demonstrated that effort prediction studies may be invalid if the quality of the dataset adopted for evaluation is not considered. Of the available work in this area, most studies focus in a single dimension of quality including *inconsistency* [24], *relevancy* [20], *correlation* and *distribution* [5]. More recently, Bosu and Macdonell [9] offered a quality taxonomy that cover multiple dimensions of SEE dataset quality, including *accuracy*, *relevance* and *provenance* with dimension divided further into sub-categories. While Bosu and Macdonell [9]'s taxonomy offers insights to design a holistic quality framework for datasets, its dimensions and measures were focused on project-based datasets with numeric attributes and thus dataset with textual attribute has a limited usage of the taxonomy. For example, Bosu and Macdonell [9] quantified *accuracy -¿ noise* may need a language-model such as BERT [14] to be determined reliably.

Contribution: We introduce the JIRA Open-Source Software Effort (JOSSE) dataset to address these considerations. The dataset consists of 23,184 software development and maintenance tasks belong to 371 projects from three open-source communities Apache, JBoss, And Spring. The dataset offers recent (2004 - 2019) and textual-based attributed data points annotated with actual (100% of data points) and expert estimated (19% of data points) efforts. Further, the data set is evaluated against six quality dimensions derived from the literature, specifically, *quantity, outliers, Dissension, readability, discretising, heterogeneity and origin*. Those dimensions were identified along with their implications on the dataset as refinement scenarios.

The next section explains the dataset collection method. Section III details the dataset refinement method that incorporates six quality dimensions. In addition, the JOSSE dataset is evaluated by adopting Bosu and Macdonell [9] benchmark in Section IV . Finally, Section V sums up the paper highlights future research.

II. Collection

Our approach is inspired by Ortu et al. [23], who proposed the mining of multiple open source software projects to generate a large and general dataset of software task efforts. Following the same method, we sampled software tasks from three open-source communities that annotate their issues with expert estimates and use the JIRA issue tracking system: Apache, JBoss, and Spring to produce the JOSSE dataset.

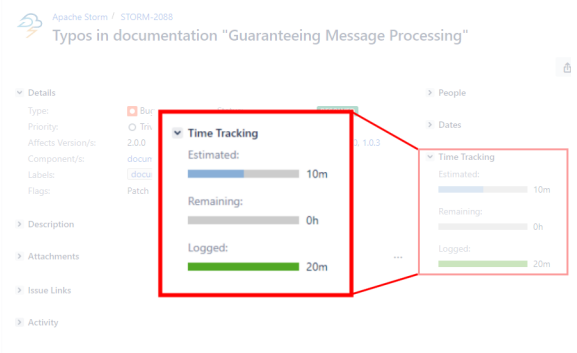


Fig. 1: A screenshot of JIRA issue tracking system. Inside the red box are details of the time-tracking information for an issue.

Attribute	DB Field	Missing Records
Issue key	id	0
Description	corpus	0
Number of Comments	num_comment	0
Number of Activities	num_activities	0
Expert Estimate	expert_estimated_effort	18857
Actual Effort	actual_effort	0
URL	reference	1118

TABLE I: List of JOSSE attributes, their DB corresponding fields and number of missing records for each attribute.

Sampling was performed using the JIRA search interface. Results were exported to multiple CSV files depending on the community record limits.

Two criteria were used to find relevant issues, these are *Spent Time* and *Status*. Spent time is referred to as Logged Time and it represents the actual effort spent on finishing an issue work. Person-hour is the unit that is used for spent time on the JIRA tracking system. Figure 1 illustrates an example of how JIRA system presents time tracking information. The dataset consists of 23,184 issues that were all annotated with actual effort. The collection activities of the data-points took place in 2019.

A Python script was then developed to aggregate the CSV files and extract a number of attributes as described below. The data was stored in a SQLite database comprising a single *Case* table.

Following a similar approach to Ortu et al. [23] we extracted issue ID,title and description. The original issue key is used as an identifier in the dataset. All the issues have a text corpus that is produced by combining the issue title with its description. Additional *derived* attributes for each issue were the total number of comments and number of activities for the issue and the total number of events that happened to the issue, such as comments or changes to attribute values, as recorded in the issue change log.

We extracted the actual cost (recorded for all issues) and the expert estimated cost where available (4,327 issues). We define this value as *expert estimation* because they were produced by

Dataset	# of Projects	# of Records	Has Corpus	Publish Year	# of Expert Estimates (%)
JOSSE	38	23184	yes	2022	4327 (18.7%)
Deep-SE	16	23313	yes	2019	0 (0%)
Porru	8	4682	yes	2016	0 (0%)

TABLE II: Summary details of the three datasets that are tasked-based with textual attribute and publicly accessible

Dataset	# of Records	Unit	Min-Max	Mean, Median	STD	Skew.	Kurt.
JOSSE	16979	P/I	2 - 2640	136 , 60	248	5.06	33
Deep-SE	23313	SP	1 - 100	6 , 4	10	6	45.69
Porru	4682	SP	1 - 6765	5 , 3	99	68.1	4652.2

TABLE III: Distribution of effort in the three datasets. Effort unit abbreviations stand for the following: P/I = person-minuteand SP = story point. Skew. stands for Skewness and Kurt. stands for Kurtosis

one or members of the open source community contributing to the project. Estimation in this approach relies on the judgement of the software team, based on their understanding of the task and the wider project context. We validated our assumption as to the source of this information by contacting several community members and asking them to describe their estimation method.

Finally, for traceability, each issue is supplied with a reference link that refers to the issue web page. The dataset consists of issues belonging to different projects from each community. Table I lists the dataset attributes along with their DB fields. I also shows the number missing records of each attribute.

To provide some context, Table II shows a simple comparison between JOSSE and other similar datasets that have public access. Table III also gives more statistical specifications of the three datasets including, mean, median, Skewness and Kurtosis of the actual effort.

The JOSSE dataset has been stored in a GitHub repository that is publicly accessible[1]. The repository contains all the necessary scripts to replicate and reproduce the dataset from its raw data.0

The JOSSE dataset consists of all the collected issues (23,184) without any further refinement or consideration of quality issues. This 'default' version may need to be refined for use in particular experiments. The next section will discuss several refinement options that can be applied to the dataset and report on the impact of the refinements in terms of dataset quality. However, it is up to the researchers who are planning to use the JOSSE dataset as part of their research to decide which refinement should be applied given the context of the dataset usage. For example, those doing experiments involving NLP techniques, such as BERT [14], may require a readability refinement, since BERT has been trained on a human language

[1]https://github.com/JOSSE-dataset/JOSSE-dataset.git

text corpus for a language such as English. Another example is dataset noise. In some cases, researchers generate artificial noise to evaluate their model robustness and examine how reliably the model can determine its confidence level.

III. Refinement Procedures

After the collection of the raw data points and depending on the research context, some data refinement options are necessary to eliminate any undesirable data points that may negatively impact the research outcomes. Inspired by Bosu and MacDonell's taxonomy, six refinement procedures were applied to enhance accuracy, relevance, and provenance. The procedures are: project-based quantification, outlier detection, assurance of data point cohesion, assessing corpus readability, discretising actual effort and its estimates and tracing data origin and reproducibility. Table IV lists all the dataset projects along with a summary details of the refinement procedure impacts after applying all the procedures. The following subsections explain each phase and evaluate the impact on associated quality metrics.

A. Quantity of Data Points Per Project

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

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

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

B. Dataset Outliers

The dataset's outlier data points are detected using David and Tukey [12]'s method to identify data points outside the lower and upper bounds. The outlier detection has been done on a project basis, i.e. data points are grouped based on their projects, then the outliers are identified. If outlier removal is required, two phases may need to be considered: project-based and individual-based removal.

The first phase identifies outliers as a percentage of the whole project data points. Any project with outlier data points

Project	# of Expert Estimates (%)	# of Outlier Records (%)	# of Inconsistent Records (%)
ACCUMULO	6 (0.5%)	149 (13.3%)	30 (2.7%)
AEROGEAR	185 (90.2%)	9 (4.4%)	4 (2%)
AMBARI	41 (2.4%)	123 (7.2%)	34 (2%)
ARROW	18 (1.1%)	174 (10.5%)	41 (2.5%)
ARTEMIS	0 (0%)	9 (10%)	1 (0.8%)
BATCH	290 (89.5%)	33 (10.2%)	4 (1.3%)
BEAM	15 (1.1%)	112 (8.2%)	26 (1.9%)
CALCITE	2 (1.5%)	7 (5.1%)	2 (1.5%)
CARBONDATA	15 (0.9%)	138 (8%)	34 (2%)
DAFFODIL	0 (0%)	15 (8.5%)	2 (1.2%)
EXOJCR	489 (69%)	52 (7.3%)	13 (1.9%)
FLINK	7 (1%)	47 (7%)	12 (1.8%)
GEODE	0 (0%)	119 (9.4%)	24 (1.9%)
GTNPORTAL	166 (80.6%)	14 (6.8%)	3 (1.5%)
HDDS	0 (0%)	22 (13.9%)	2 (1.3%)
IGNITE	2 (0.4%)	110 (21.4%)	12 (2.3%)
INT	385 (93.2%)	47 (11.4%)	7 (1.6%)
JBAS	33 (33.3%)	6 (6.1%)	2 (2%)
JBEAP	4 (2.9%)	24 (17.6%)	5 (3.8%)
JBESB	5 (4%)	25 (20.2%)	1 (0.7%)
JBFORUMS	0 (0%)	14 (12.8%)	5 (4.2%)
JBLAB	45 (19.6%)	27 (11.7%)	4 (1.8%)
JBPORTAL	25 (16.7%)	22 (14.7%)	2 (1.3%)
JBTM	114 (59.1%)	24 (12.4%)	6 (3.2%)
METRON	0 (0%)	5 (4.7%)	2 (1.6%)
MNG	113 (78.5%)	21 (14.6%)	5 (3.8%)
MXNET	0 (0%)	24 (7.9%)	4 (1.4%)
NETBEANS	1 (0.3%)	24 (8.2%)	6 (1.9%)
NIFI	4 (2.3%)	15 (8.6%)	4 (2.4%)
RF	236 (84.3%)	162 (57.9%)	4 (1.3%)
SLING	1 (0.9%)	5 (4.6%)	3 (2.9%)
SPR	9 (5%)	25 (13.9%)	4 (2.3%)
STDCXX	178 (89.9%)	28 (14.1%)	4 (2%)
STORM	4 (0.5%)	87 (10.4%)	16 (1.9%)
STS	98 (39.7%)	35 (14.2%)	5 (2.2%)
SWS	5 (5.6%)	14 (15.7%)	3 (3.2%)
TS	0 (0%)	22 (6.8%)	7 (2.2%)
ZOOKEEPER	6 (2.8%)	20 (9.3%)	4 (1.8%)
Total	2502 (14.7%)	1809 (10.7%)	347 (2%)

TABLE IV: Summary details of JOSSE dataset after Appling refinement scenarios

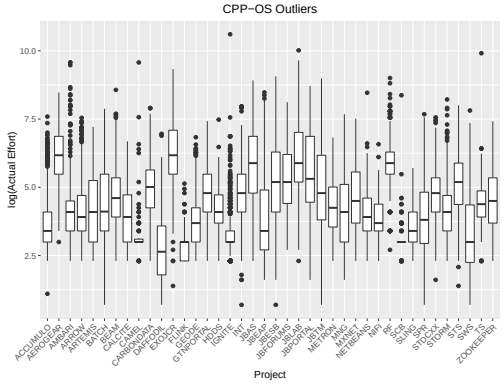
Community	Projects with $<$ 100 DP Count	%	Median	IQR	Skewens
RedHat	86	89%	5	19.25	1.91
Spring	21	81%	8	26	1.58
Apache	226	91%	4	10.5	2.70

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

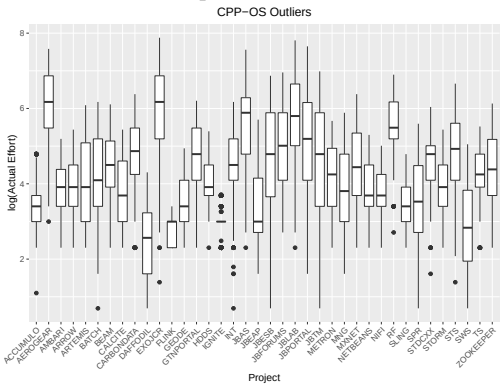
that represent more than a certain threshold is removed. Then, an individual-based removal of data points should follow to purify projects that have a minor number of outliers. Such a removal scenario is based on an assumption that the outliers represent a minority of the whole project data points, and if they are not a minority then the project may not be suitable to draw patterns from.

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

To visualise the difference before and after outlier removal, Figure 2 shows box plot charts of the dataset before and after outlier removal. The upper part of the figure (2a) shows a large number of outliers, as represented by the black dots outside the whisker range (David and Tukey [12]'s fences). On the other hand, a significant reduction of outliers is illustrated in the lower part. Some projects still have a few outliers shown in the box plot, which represent the outliers of the new distribution after the removal of the original outliers.



(a) JOSSE box plot before outlier removal



(b) JOSSE box plot after outlier removal

Fig. 2: Two box plots representing the dataset before and after the outlier removal.

C. Dataset Dissension

As mentioned in several studies, data *dissension* has been referred to as effort inconsistency [24] and data inconsistency [9]. It could impact prediction accuracy if it is present. According to Bosu and Macdonell [9], inconsistency is a lack of data point harmony in terms of their property values. Phannachitta et al. [24] refers to inconsistency when the assumption that similar projects have similar efforts is violated. In other words, if data points have similar effort, they should exhibit similar property values (similar to each other).

While the term inconsistency has been used differently in the literature, this chapter uses data dissension to refer to how cohesive the data points belonging to a given project are. The challenge with the JOSSE dataset is that the main attribute is a corpus, and measuring similarity between different corpuses must consider lexical similarity as well as semantic similarity. Thus, Phannachitta et al. [24]'s method may not suit JOSSE since it has been designed for datasets with numeric properties. Nevertheless, Phannachitta et al. [24] provided insights to inspect data dissension in the JOSSE dataset.

Text similarity is a concern of Natural Language Processing (NLP) research, and BERT [14] is among the state-of-the-art advancements in that field. Devlin et al. [14] proposed a pre-trained deep learning data model that can be used to encode a given text into word vectors. The lexical and semantic meaning of each word is embedded in the word vector. The BERT model has been trained on a large amount of literature, so one word may have different word vectors based on the context.

For the case of assessing the JOSSE dataset cohesion, issue corpuses were converted to BERT embeddings and then a cosine similarity between issue vectors was calculated. To illustrate the harmony between data points (issues), heat maps were created for each project, see Figure 3. Each heat map square represents a 25-issue random sample from each project. The dissension is represented in a colour range from yellow (100% similarity) to navy blue (83.5% similarity). Overall, the JOSSE dataset is cohesive, that is, the data point properties exhibit high similarity, except for a negligible number of data points per project (less than 5% dissension). From 1% to 5% of the data points were are not similar (have a BERT cosine similarity beyond David and Tukey [12]'s fences) to the rest of the data points. For instance, project ZOOKEEPER has 5% dissension, while BATCH has a 2.6% dissension; Table IV gives the dissension percentage for each project in the dataset.

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

D. Dataset Readability

Since the main property of the data points in the dataset is a corpus, it is necessary to examine the content of that

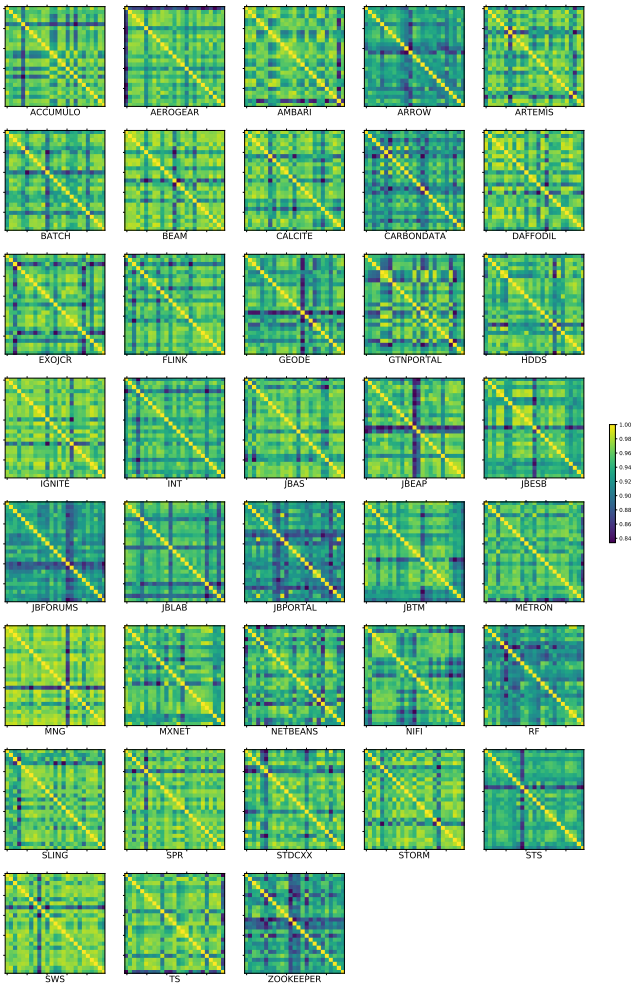


Fig. 3: JOSSE dissension heat maps. Each map represents a sample of 25 data points that are selected randomly for each project.

corpus. Some ML models may make assumptions about the corpus, e.g. written in readable text and following a given language's grammar. There are several readability assessment techniques, such as Flesch–Kincaid [18], however, they might not be useful for assessing a text corpus with grammar errors or code snippets that contains a stack trace.

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

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

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

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

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

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

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

E. Discretising Software Effort Estimates

Taking inspiration from the Planning Poker method, software effort estimation can be discretised and grouped into categorical times. Estimate categories often adopt a metaphor that suggests increasing uncertainty with estimate magnitude. For example, Grenning [15] suggests using a Fibonacci sequence to indicate the margin of error between estimate sizes as they increase in magnitude. Cohn [11] states that approximate person-effort categories are more appropriate because it is often unrealistic to expect person-hour precision estimates to be accurate for software tasks. Therefore, classification can be used to build machine learning models instead of regression.

Further, Cohn [11] argues that teams eventually develop a tacit interpretation of the relationship between the relative categorical estimate and actual person-time costs, as the completed tasks are compared to the team's available person-hour budget over several sprints. Similarly, Menzies and Shepperd [22] asserts that data discretisation concentrates signals in datasets, which significantly enhances the ML model's performance. The discretisation is an essential consideration,

since the datasets that are adopted attribute expert-based effort estimates.

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

Using a Planning Poker model as a strategy helps narrow the gap between expert-based and ML-based prediction, as Jørgensen et al. [17] concluded that such a combination is one step towards better estimation. The Planning Poker model inspires the research in designing a new discretisation category that avoids some drawbacks, such as discretisation noise [26], and brings in the concept of magnitude of error to discretisation. Thus, this discretisation is referred to as magnitude discretisation in this paper.

F. Dataset Domain and Origin

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

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

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

For each data point in the dataset, there is a reference link where further information about the data point can be acquired. Data point references also provide provenance and trustworthiness. The whole dataset can be reproduced from the reference links if necessary.

IV. JOSSE Evaluation

Following the quality criteria of Bosu and Macdonell [9]'s taxonomy, Table VII shows the JOSSE assessment against each criterion of the taxonomy. Three criteria have been adjusted to fit in the context of the JOSSE dataset, including noise and outliers. Noise has been replaced by data dissension, since the noise method that is used by Bosu and Macdonell [9] is not applicable to the JOSSE text corpus properties. In addition, data dissension is not necessarily a bad sign for the dataset, as described in Bosu and Macdonell [9]'s taxonomy when they discussed *noise*. Further details about data dissension and inconsistency, outliers, amount of data, accessibility, and provenance can be found in sections III-C, III-B, and III-F accordingly.

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

		RedHat	Spring	Apache
Dissension*		2.15%	2.12%	1.99%
Outliers*		10%	12%	16%
Amount of data		3,953	1,808	17,427
Timeliness	Year*	2004-18	2007-11	2004-19
	Dates	No	No	No
Inconsistency		No	No	No
Incompleteness		Yes	Yes	Yes
Redundancy		No	No	No
Heterogeneity		Yes	Yes	Yes
Commercial Sensitivity		No	No	No
Accessibility		Yes	Yes	Yes
Provenance/ Trustworthiness		Yes	Yes	Yes

TABLE VII: JOSSE evaluation against Bosu and Macdonell [9]'s quality taxonomy. Columns with * are adjusted to fit in the JOSSE context.

Interpreting the JOSSE assessment results listed in Table VII in the context of Bosu and Macdonell [9]'s assessment of the 13 SEE datasets and using their approach, JOSSE has less noise, more data points, and is more recent. The average dissension (reported as noise in Bosu and Macdonell [9]'s assessment) of JOSSE is 2.1%, which equals to the lowest noise reported among the 13 datasets. The average noise of all the 13 datasets is 17%. On average, the JOSSE dataset has a slightly higher outlier percentage (12.6%) than the 13 datasets (9.6%). The median number of records in the 13 datasets is 62, with one dataset (ISBSG16) that has 7518 records. The JOSSE dataset has 23,188 records. The 13 datasets have records about software projects, whereas the JOSSE dataset records are software development tasks, and this is a key difference between the datasets which explains the large number of JOSSE records. The records of the 13 datasets dated back to the year of 1989, and the most recent records were created in 2015. The JOSSE dataset records were created between 2004 and 2019.

Generally, JOSSE offers less noise, more granularity, more data, and more recent records of open-source software development activities compared with the 13 datasets evaluated by Bosu and Macdonell [9].

V. Conclusion

Whilst datasets are important bases for researchers, few studies pay attention to collecting and maintaining SEE datasets. In fact, the most popular datasets that are used in the recent ML SEE research are quite old. Therefore, this paper analysed research datasets more closely and introduced the JOSSE dataset as a task-based dataset with a textual attribute, actual and estimated efforts. It also reviewed dataset quality

studies, such as Bosu and Macdonell [9]'s comparative study of different datasets. Then, it reflected the learned lessons in the JOSSE dataset to ensure its quality and fitness to be used as training data for ML SEE models.

As a future work, JOSSE may be used to evaluate recent advancement in Natural Language Processing and whither a textual attribute may offer more correlated features then numeric attributes. For instance, BERT can be used for both feature extraction and model training. More interestingly, is to compare BERT as a feature extraction with a more simple method such as TF-IDF to evaluate the effect of context-aware features on the reliability of effort predictions.

REFERENCES

[1] [n.d.]. Software Project Benchmarking - Home Page - ISBSG. https://www.isbsg.org/

[2] [n.d.]. Spell and Grammar Checker. https://languagetool.org/

[3] Asad Ali and Carmine Gravino. 2019. A systematic literature review of software effort prediction using machine learning methods. *J. Softw. Evol. Process.* 31, 10 (2019). https://doi.org/10.1002/smr.2211

[4] Mohit Arora, Sahil Verma, Kavita, and Shivali Chopra. 2020. A Systematic Literature Review of Machine Learning Estimation Approaches in Scrum Projects. *Advances in Intelligent Systems and Computing* (2020), 573586. https://doi.org/10.1007/978-981-15-1451-7$_5$9

[5] Amid Khatibi Bardsiri, Seyyed Mohsen Hashemi, and Mohammadreza Razzazi. 2015. Statistical Analysis of The Most Popular Software Service Effort Estimation Datasets. *Journal of Telecommunication, Electronic and Computer Engineering (JTEC)* 7, 1 (2015), 87–96.

[6] Barry W. Boehm. 2002. Software Engineering Economics. *Software Pioneers* (2002), 641686. https://doi.org/10.1007/978-3-642-59412-0$_3$8

[7] G Boetticher. 2001. Using machine learning to predict project effort: Empirical case studies in data-starved domains. In *Model Based Requirements Workshop*. 17–24.

[8] Michael Franklin Bosu and Stephen G. MacDonell. 2013. A Taxonomy of Data Quality Challenges in Empirical Software Engineering. In *22nd Australian Conference on Software Engineering (ASWEC 2013), 4-7 June 2013, Melbourne, Victoria, Australia*. IEEE Computer Society, 97–106. https://doi.org/10.1109/ASWEC.2013.21

[9] Michael F. Bosu and Stephen G. Macdonell. 2019. Experience: Quality Benchmarking of Datasets Used in Software Effort Estimation. *Journal of Data and Information Quality* 11, 4, Article 19 (2019), 38 pages. https://doi.org/10.1145/3328746

[10] Morakot Choetkiertikul, Hoa Khanh Dam, Truyen Tran, Trang Pham, Aditya Ghose, and Tim Menzies. 2019. A Deep Learning Model for Estimating Story Points. *IEEE Trans. Software Eng.* 45, 7 (2019), 637–656. https://doi.org/10.1109/TSE.2018.2792473

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

[12] F. N. David and John W. Tukey. 1977. Exploratory Data Analysis. *Biometrics* 33, 4 (Dec 1977), 768. https://doi.org/10.2307/2529486

[13] Jean-Marc Desharnais. 1989. Analyse statistique de la productivite des projets de developpement en informatique a partir de la technique des points de fonction. *Master's thesis, Univ. du Quebec a Montreal* (1989).

[14] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).

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

[16] Ali Idri, Mohamed Hosni, and Alain Abran. 2016. Systematic literature review of ensemble effort estimation. *J. Syst. Softw.* 118 (2016), 151–175. https://doi.org/10.1016/j.jss.2016.05.016

[17] Magne Jørgensen, Barry W. Boehm, and Stan Rifkin. 2009. Software Development Effort Estimation: Formal Models or Expert Judgment? *IEEE Softw.* 26, 2 (2009), 14–19. https://doi.org/10.1109/MS.2009.47

[18] J. Peter Kincaid, Richard Braby, and John E. Mears. 1988. Electronic authoring and delivery of technical information. *Journal of Instructional Development* 11, 2 (Jun 1988), 813. https://doi.org/10.1007/bf02904998

[19] Barbara A. Kitchenham and Emilia Mendes. 2009. Why comparative effort prediction studies may be invalid. In *Proceedings of the 5th International Workshop on Predictive Models in Software Engineering, PROMISE 2009, Vancouver, BC, Canada, May 18-19, 2009*, Thomas J. Ostrand (Ed.). ACM, 4. https://doi.org/10.1145/1540438.1540444

[20] Ekrem Kocaguneli, Gregory Gay, Tim Menzies, Ye Yang, and Jacky W. Keung. 2010. When to use data from other projects for effort estimation. In *ASE 2010, 25th IEEE/ACM International Conference on Automated Software Engineering, Antwerp, Belgium, September 20-24, 2010*, Charles Pecheur, Jamie Andrews, and Elisabetta Di Nitto (Eds.). ACM, 321–324. https://doi.org/10.1145/1858996.1859061

[21] Emilia Mendes, Nile Mosley, and Steve Counsell. 2005. Investigating Web size metrics for early Web cost estimation. *J. Syst. Softw.* 77, 2 (2005), 157–172. https://doi.org/10.1016/j.jss.2004.08.034

[22] Tim Menzies and Martin J. Shepperd. 2012. Special issue on repeatable results in software engineering prediction. *Empir. Softw. Eng.* 17, 1-2 (2012), 1–17. https://doi.org/10.1007/s10664-011-9193-5

[23] Marco Ortu, Giuseppe Destefanis, Bram Adams, Alessandro Murgia, Michele Marchesi, and Roberto Tonelli. 2015. The JIRA Repository Dataset: Understanding Social Aspects of Software Develop-

ment. In *Proceedings of the 11th International Conference on Predictive Models and Data Analytics in Software Engineering, PROMISE 2015, Beijing, China, October 21, 2015*, Ayse Bener, Leandro L. Minku, and Burak Turhan (Eds.). ACM, 1:1–1:4. https://doi.org/10.1145/2810146.2810147

[24] Passakorn Phannachitta, Jacky Keung, Kwabena Ebo Bennin, Akito Monden, and Kenichi Matsumoto. 2016. Filter-INC: Handling Effort-Inconsistency in Software Effort Estimation Datasets. *2016 23rd Asia-Pacific Software Engineering Conference (APSEC)* (2016). https://doi.org/10.1109/apsec.2016.035

[25] Simone Porru, Alessandro Murgia, Serge Demeyer, Michele Marchesi, and Roberto Tonelli. 2016. Estimating Story Points from Issue Reports. In *Proceedings of the The 12th International Conference on Predictive Models and Data Analytics in Software Engineering, PROMISE 2016, Ciudad Real, Spain, September 9, 2016*. ACM, 2:1–2:10. https://doi.org/10.1145/2972958.2972959

[26] Gopi Krishnan Rajbahadur, Shaowei Wang, Yasutaka Kamei, and Ahmed E. Hassan. 2021. Impact of Discretization Noise of the Dependent Variable on Machine Learning Classifiers in Software Engineering. *IEEE Trans. Software Eng.* 47, 7 (2021), 1414–1430. https://doi.org/10.1109/TSE.2019.2924371

[27] J. Sayyad Shirabad and T.J. Menzies. 2005. The PROMISE Repository of Software Engineering Databases. School of Information Technology and Engineering, University of Ottawa, Canada. http://promise.site.uottawa.ca/SERepository

[28] Pinkashia Sharma and Jaiteg Singh. 2017. Systematic Literature Review on Software Effort Estimation Using Machine Learning Approaches. *2017 International Conference on Next Generation Computing and Information Systems (ICNGCIS)* (Dec 2017). https://doi.org/10.1109/icngcis.2017.33

[29] Martin J. Shepperd and Chris Schofield. 1997. Estimating Software Project Effort Using Analogies. *IEEE Trans. Software Eng.* 23, 11 (1997), 736–743. https://doi.org/10.1109/32.637387

[30] Jianfeng Wen, Shixian Li, Zhiyong Lin, Yong Hu, and Changqin Huang. 2012. Systematic literature review of machine learning based software development effort estimation models. *Inf. Softw. Technol.* 54, 1 (2012), 41–59. https://doi.org/10.1016/j.infsof.2011.09.002

[31] Shahid Kamal Tipu Ziauddin and Shahrukh Zia. 2012. An effort estimation model for agile software development. *Advances in computer science and its applications (ACSA)* 2, 1 (2012), 314–324.