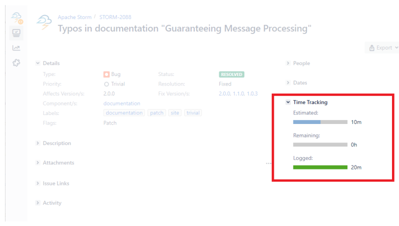
JOSSE: A Software Development Effort Data Set Annotated with Expert Estimates

Mohammed Alhamed
School of Computing Science
The University of Glasgow
Glasgow, Scotland
m.alhamed.1@research.gla.ac.uk

Tim Storer
School of Computing Science
The University of Glasgow
Glasgow, Scotland
timothy.storer@glasgow.ac.uk

***Abstract*—The JIRA Open Source Software Effort (JOSSE) data set consists of software development and maintenance tasks collected from the JIRA issue tracking system for three open source communities, including Apache, JBoss, And Spring. Issued from JOSSE data set has been used in the crowd planning poker (CPP) research. CPP's replication pack has also been supplied and explained in this essay.**


Apache Storm / STORM-2088
Typos in documentation "Guaranteeing Message Processing"

Details
Type: Bug
Status: RESOLVED
Priority: Trivial
Resolution: Fixed
Affects Version/s: 2.0.0
Fix Version/s: 2.0.0, 1.1.0, 1.0.3
Component/s: documentation
Labels: documentation patch site trivial
Flags: Patch

Time Tracking
Estimated: 10m
Remaining: 0h
Logged: 20m

Fig. 1. A screenshot of JIRA issue tracking system. Inside the red box, details of time tracking information of an issue.

I. Introduction

As part of the research work that investigates crowds' ability to predict an expert-comparable estimate for software development and maintenance tasks, it was crucial to find such tasks annotated by expert estimates and descriptions. In the same time, it was essential for those tasks to be open so that the public crowd can predict them. Up to our knowledge, the literature has no such data set. This abstract will discuss the JIRA Open Source Software Effort (JOSSE) Data set and the Crowd Planing Poker (CPP) research replication pack.

Fortunately, some open-source software projects and communities have open issues tracker systems, e.g. JIRA, that contains software development and maintenance issues (tasks) annotated with expert estimates. From which, JOSSE data was collected. Then a subset of the 30 issues was used in the experimental work of CPP.

The abstract is organized around two topics, collection and transforming JOSSE data set, and describing the replication pack of CPP experimental work.

II. JOSSE Data Set

Using the JIRA user interface for a couple of open-source projects, it was possible to collect many issues that meet the main research criteria. The criteria are software development tasks annotated with description, actual cost, and expert estimate. The collected data points gathered in one databased and named as JIRA Open Source Software Effort (JOSSE) Data set.

The literature has several software effort data sets publicly available such as those at PROMISE repository [1]. While most of those data sets are projects based, they also tend to have few data points and associate only numeric attributes. However, Ortu et al. [2] proposed a large and general dataset collected from JIRA for several open-source software projects. While Ortu et al. [2] proposed method of mining JIRA is inspiring, their data set does not include expert estimates. Similarly, other data set have been proposed [3] but have no expert estimates.

JIRA of three open source communities including Apache, JBoss, and Spring was used to search for software issues with text descriptions and annotated by an actual time and expert estimate. The next subsection will discuss the filtration process. Then, more details will be illustrated about the collected data. Finally, a discussion about the data transformation and access will follow.

A. Filtering and Collecting JOSSE Data

JIRA interface provides a feature to filter out software issues using several criteria. Two of them were selected including ”Spent Time” and ”Status”. These two criteria were toned to find relevant issues. JIRA interface refers to Spent time as Logged time. It represents the actual effort that is spent working on an issue. Person-second is the unit used in spent time, see Figure 1 for an example of how JIRA system presents time tracking information. Status filter represents the issue status to reflect on its progressive milestones, e.g. New.

Spent time was set to be greater than or equal 150 seconds (5 minutes). However, the Status has a couple of values that tell the issue has finished. Different communities have a different set of status values. Nevertheless, the values used are: Fixed, Resolved, Done, Completed, and Closed.

The search resulted in 23,184 issues. After an initial analysis, only 4,327 were annotated with an estimated effort and actual effort. All the issues were kept even those with no expert effort to widen the data set support for other experiments.

Later, during data transformation, those issues without expert estimates will be easily distinguished.

Finally, the filtered issues were retrieved using the export feature of JIRA. The system limits the number of issues that can be extracted at one time, and thus, the issues were retrieved in thousand-issue batches. Comma-Separated Values (CSV) was the export format that is selected to download the issue batches.

B. Data Transformation and Storage Format

The downloaded data includes many issue attributes, and thus, it was necessary to filter out the targeted attributes that serve the purpose of CPP research. Therefore, the data have gone through a transformation process to extract issue Key, Corpus, Actual Effort, Expert Estimate, Features, and Reference.

Issue Corpus produced by combining issue title with its description. Issue Features consists of the issue's total comments and activities number. The activities number was extracted from the issue log. It represents a sum of all the events that happened for a given issue. Issue Reference is a link that refers to the issue web page. It is extracted for the reason of traceability of each issue using the Issue Key. The key that is used is JIRA original issue key.

Instead of using CSV, the research team opted for an SQL-based format and SQLite was selected to store the extracted data. The SQLite DB consist of a single table named as "case", and its fields named after the extracted data as explained above, except for the Issue Key named as "id".

The raw data and transformation Python code are available in the same repository of the dataset under a folder named "dataset_replication". The code contains functions that read CSV files and parse the data of interest.

JOSSE dataset stored in a public Github repository that is publicly accessible. The repository URL is https://github.com/crowd-planning-poker/JOSSE-Dataset.git. Zenodo generated a DOI for the dataset, and it is 10.5281/zenodo.4459154.

III. Crowd Planning Poker (CPP) Replication Pack

CPP experimental work used JOSSE data set to randomly select issues that meet the experiment design, as explained in the research publication [4]. Thirty issues have been used and annotated with supplementary information.

The pack consists of two parts, experiment data and calculation code. More details about the data and the code are in the following subsections.

A. CPP Experiential Data

CPP experiment aims to evaluate crowd workers ability to predict an expert-comparable estimate for software development tasks. Thus, JOSSE data set has been used to find software issues that fit the experimental design.

In the replication pack, there are four data files. Three of them are in CSV format, and one uses Microsoft Excel format. The experiment input data listed in "30_issues_data.csv", and its output stored in "crowd_estimates.csv". The third CSV file is "direct_url_to_all_issues.csv". For convenience, the input and output data are copied and stored as Microsoft Excel. Sometimes, CSV files may not be readable by the conventional reader since the issues contain code snippets that confuse conventional reader tools. The fields and the content of the input and output files are explained in the "readme.md" file of the replication repository.

B. CPP Experiential Code

The source code contains functions that are used to calculate CPP output, such as CPP round Kappa. It is written in Python, and it only requires one package, the NLTK agreement package. The list of functions inside the file are explained as the following:

- calculate_round_kappa(): this function calculates the agreement between crowd workers in each round. It uses Fleiss' Kappa implementation in the NLTK package.
- aggregate_crowd_estimates(): this function aggregates crowd estimate for each round and overall the estimation session.
- Time formatting functions: two functions that are used to help time format. The first one: estimates_hours_format() to group estimates into estimate category and estimates_time_category_format() to return the category name.

C. Replication storage and Access

Both CPP data and code are available in the replication repository, a public Github repository accessible via https://github.com/crowd-planning-poker/cpp-30-issue-replication-pack. Zenodo generated a DOI for the replication pack, and it is 10.5281/zenodo.4459186.

References

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

[2] M. Ortu, G. Destefanis, B. Adams, A. Murgia, M. Marchesi, and R. Tonelli, "The JIRA repository dataset," in *Proceedings of the 11th International Conference on Predictive Models and Data Analytics in Software Engineering - PROMISE '15*. ACM Press, 2015. [Online]. Available: https://doi.org/10.1145/2810146.2810147

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

[4] M. Alhamed and T. Storer, "Playing planning poker in crowds: Human computation of software effort estimates," 2021.