A DATA EXTRACTION ALGORITHM FROM OPEN SOURCE SOFTWARE PROJECT REPOSITORIES FOR BUILDING DURATION ESTIMATION MODELS: CASE STUDY OF GITHUB

Donatien K. Moulla[1, 2], Alain Abran[3] and Kolyang[4]

[1]Faculty of Mines and Petroleum Industries, University of Maroua, Cameroon
[2]LaRI Lab, University of Maroua, Cameroon
[3]Department of Software Engineering and Information Technology, École de Technologie Supérieure, 1100, rue Notre-Dame Ouest Montréal, Canada
[4]The Higher Teachers' Training College, University of Maroua, Cameroon

ABSTRACT

Software project estimation is important for allocating resources and planning a reasonable work schedule. Estimation models are typically built using data from completed projects. While organizations have their historical data repositories, it is difficult to obtain their collaboration due to privacy and competitive concerns. To overcome the issue of public access to private data repositories this study proposes an algorithm to extract sufficient data from the GitHub repository for building duration estimation models. More specifically, this study extracts and analyses historical data on WordPress projects to estimate OSS project duration using commits as an independent variable as well as an improved classification of contributors based on the number of active days for each contributor within a release period. The results indicate that duration estimation models using data from OSS repositories perform well and partially solves the problem of lack of data encountered in empirical research in software engineering.

KEYWORDS

Effort and duration estimation, software project estimation, project data, data extraction algorithm, GitHub repository

1. INTRODUCTION

Software project estimation has always been a challenge for software engineering communities [1, 2]. One of the factors which influence the development of software project estimation is project data, which is often incomplete and at times inconsistent. The performance of an estimation model depends on the characteristics of the software project data such as size, missing values and outliers, as well as the validation techniques used (leave-one-out cross validation, holdout, n-fold cross validation) and evaluation criteria [1]. Some organizations such as the International Software Benchmarking Standard Group (ISBSG) and PROMISE provide worldwide repositories of software projects [3, 4]. These repositories provide data useful for multiple purposes, including project productivity comparison and estimation of the effort required to build productivity models [5]. However, different software projects have their own characteristics, such as application domain, type of system, development platform, type of language, etc. In addition, data collected may present quality problems due to noise, outliers, incompleteness, even in data collected by mature organizations [6, 7, 8, 9]. In addition, it is considerably more difficult to get both large datasets and diverse datasets. The amount of data available for building estimation models affects the performance of the generated models, especially with small datasets where results may not be generalizable.

An OSS source code management repository is an online community platform that makes project history data available to IT practitioners and researchers by enabling developers to host and review code, manage projects, share knowledge and work together to build OSS. Such OSS code repositories do not hold direct information on the total duration and effort of any specific project. Due to the quantity and diversity of the raw data (such as commits, contributors, LOC, etc.), extracting and processing the data for either duration or effort estimation is a challenge.These OSS repositories usually contain the following information:

- Delivered software source code with corresponding dates.
- Activities of the contributors who submitted their codes to a project (commits).
- Meta-data (commit ID, date of commitment, release tag, etc.).
- Contributor performing the commit.

All of this information can be used directly as independent variables, or through derived variables, to develop duration estimation models.

In this paper, we focus on the design of an algorithm to extract data from the GitHub source code management repository to design duration estimation models. Three measures of activity are explored as independent variables:

(1) The number of changes to source code (commits) per contributor in each release.
(2) The number of active days per contributor in each release, i.e., the days in which a contributor performs at least one commit to the code base.
(3) The number of lines of code added per contributor in each release. We define LOC as the total number of lines of code, not including blank spaces and comments.
(4) Type of contributors, where the type of contributors is an indirect variable defined within our research methodology.

Data from the WordPress project, extracted from the source code management repository GitHub, were used to test our algorithm and design the duration estimation models. The rationale behind selecting the WordPress project is related to its OSS economic model, data availability and its significance within the Open Source community: WordPress is one of the most used content management system (CMS) open source software in the world and its economic model is based on service and hosting.

The paper is structured as follows. Section 2 presents related work on duration and effort estimation with a focus on project datasets. Section 3 describes the proposed algorithm. Section 4 details the test results. Section 5 concludes with a summary of the key findings, the study limitations and directions for future work.

2. RELATED WORK

This section presents related work on empirical software engineering datasets and duration and effort estimation with a focus on OSS projects.

Qi et al. [10] provided a method to collect sufficient data for solving the problem of lack of training data when building an effort estimation model using GitHub data. Although their method can be used to collect sufficient data for solving the problem of lack of training data when building an effort estimation model, it is subject to certain threats to validity. They only used the java open source projects of GitHub to calculate effort as a derived variable. To identify an active contributor within their study, they adopted the following definitions:

- Active contributor: in a period of consecutive active days (with a of default 30 days), an active contributor has an average of committing one record per active day, and whose total contributions exceed the average contributions of the project;
- Inactive contributor: otherwise.

Badashian et al. [11] tracked a variety of metrics such as commits, Pull Reqs, Projects Watched, Issue Comments, Followers, and others to study the activities of developers.

MacDonell et al. [12] provided a transparent and consistent means of collection and evaluation of data that could lead to the use of higher quality data in software engineering experiments. They assessed the quality of 13 empirical software engineering datasets with the aim of benchmarking them against the data quality taxonomy proposed by Bosu and MacDonell [13] based on three groups: accuracy, relevance and provenance.

According to Gencel et al. [14], inconsistent results are obtained when software effort estimation models are built using benchmark repositories for two reasons:

- the lack of common standards and vocabulary;
- the differences in definitions and categories of attributes of the different repositories.

Cheikhi and Abran [15] surveyed the PROMISE and ISBSG repositories with the objective of making them easier for researchers to understand the data in them and thus more readily use the data.

Kocaguneli et al. [16] proposed a solution for importing relevant data from other organizations to build effort estimation models. They explored cross versus within data “sources”, where the data is divided by the values of any single feature. With in studies are localized to one subset while a cross study trains from some subsets and tests on others. The datasets used for their research were COCOMO81, nasa93, Desharnais, Finnish, Kemerer and Maxwell datasets. They found that the *cross* performance results are no worse than *within*.

Shihab et al. [17] performed an empirical study on four OSS projects (namely JBoss, Spring, Hibernate and Mule) and compared the use of LOC to other code variables (in particular code complexity). To obtain code complexity, they required the source code of the changed files. They used the list of file names to download the corresponding revisions of files from the JIRA database. They used correlation tests and linear regression models to investigate the categories of variables as a substitute for effort. Since effort was available in their dataset, they investigated the used code complexity and LOC as a substitute measure of effort, a unique feature of this study. However, in their dataset the effort data was at the “issues” level but the variables used for statistical analysis were at another level (e.g., at the “files” level). A number of assumptions had to be made to assign effort across levels (e.g., from issues to files). Doing so, of course, introduces a number of unknown distortions without the ability quantify them and analyze their impact on the reported findings. While the authors report that there are correlations between the indirect effort measures with actual effort, these results are quite low and provide little support for the relevance of the indirect measures of effort in other related works for OSS effort and duration estimation purposes.

In summary, of the studies on empirical software engineering datasets and effort and duration estimation very few are OSS projects. The size and choice of the datasets presents a challenge for research on OSS project estimation. The keys findings of these specific studies show that there are datasets that have been used extensively in research on software effort estimation but most of them do not contain sufficient information to enable researchers to evaluate their accuracy (outliers, inconsistency, incompleteness, etc.), relevance (amount of data, heterogeneity and timeliness) and provenance (trustworthiness, accessibility and commercial sensitivity). In addition, related OSS work suffers from a lack of data on effort itself and must be approximated indirectly through other variables without knowing their variability and the impact of such variability on the estimation models themselves. To date, OOS studies by researchers for estimating effort are based on substitute variables that do not rely on reliable methodological bases [17]. As long as reliable substitute of effort measures are not available, it is not wise to build estimation models based on these very weak substitutes. However, for duration estimation, this variable is directly calculable from direct data of project calendar dates and our research relates exclusively to duration estimation models.

3. PROPOSED ALGORITHM

Data extraction selects, within a project, the number of changes made by each contributor within a specific period. These changes are related to the total number of commits, LOC added, LOC removed and active days. The algorithm to generate data from GitHub is presented next.

```
Algorithm: Extraction of data
Input: Project P
Outputs: Contributors, Commits, LOC added, LOC removed, Active days
Begin
{
   1. Write the range of date on which data extraction must be done
   2. Read (beginning date, end date)
3.For this range of date:
   4. List all contributors who made a change
   5.For each contributor:
6.       { commit<- total number of changes to source code
  7.insert <- total number of LOC added
  8.        delete <- total number of LOC removed
  9.        day  <- total number of active days
10.      write in output (contributor \t commit \t LOC added \t LOC      removed \t
active days \n)
11.}
12.End for
}
End
```

The input of the algorithm is the project P and the outputs are the total number of contributors per release, their commits, LOC added, LOC removed and active days. The first step is to enter the beginning date and end date of the release. The second step is to generate the list of contributors for this release. A temporary file is created for this purpose. Finally, for each contributor, the algorithm calculates the total number of LOC added, LOC removed, commits and active days.

The algorithm then writes the results to a file. The implemented script for this algorithm is presented in Appendix A.

4. TEST RESULTS

4.1. Data Extraction

The proposed algorithm was implemented in C including GitHub commands.

Table 1 shows a statistical analysis of the historical data on 21 releases of the WordPress project, from February 2015 to May 2019 based on data extracted on June 23, 2019 from the GitHub repository using our algorithm. In Table 1, |LOC net| = |LOC added - LOC removed|.

Table 1.WordPress project: descriptive statistics of the 21 releases

Release	# Commits	# LOC added	#LOC removed	|LOC net|
4.2.4	227	7984	3384	4618
4.2.5	1319	28696	20281	8415
4.2.6	225	5435	3062	4409
4.2.7	589	12018	5835	8989
4.2.8	913	23886	6488	17398
4.3	876	12044	6446	5944
4.3.1	988	42575	35829	6746
4.3.2	1100	20861	8760	12825
4.3.3	1599	32842	6207	26635
4.3.4	559	29788	26392	3686
4.4.1	408	4100	2041	2257
4.4.2	518	16154	4823	11335
4.4.3	806	17548	5990	11558
4.5	487	5823	2995	3142
4.6	855	54932	37178	22192
4.7	1108	73845	35945	48848
4.8	718	42282	22272	41862
4.9	1001	126472	47171	126100
5.0	854	125568	105561	124716
5.1	586	921228	668282	920642
5.2	421	145968	81814	145547

4.2. Classification of OSS Contributors

We observed that contributions to OSS projects were not equally distributed across contributors. Several contributors were responsible for the majority of the commits, while the majority contributed only a few [18, 19]. In our study, instead of using a threshold [19], we propose an improved approach using the total number of active days for each contributor in the release duration for classifying the contributors as full-time, part-time and occasional.
For the purpose of estimating OSS project duration, we defined the following:

i) Active day for a contributor = a day in which the contributor performs at least one commit to the code base.

ii) Release duration is calculated from the time required to develop each release, that is, duration = (end date of the release) – (beginning date of the release).

iii) Approach for classifying contributors:

a) Occasional: any contributor who has been active at most one third (1/3) of the release duration.
b) Part-time: any contributor who has been active more than one third (1/3) and at most two thirds (2/3) of the release duration.
c) Full-time: any contributor who has been active more than two thirds (2/3) of the release duration.

After identifying the different contributors per category in each release according to the above definitions, the total number of commits per category of contributors per release is obtained by adding all the commits of the different contributors for each category. For example, when a category of full-time contributors consists of three contributors who made respectively, 10, 20 and 30 commits over respectively 8, 13 and 16 active days, the maximum duration release for this category of contributors is 16 days and the total number of commits for this category of contributors in this release is 60. In summary, the maximum duration by category of contributors for this release corresponds to the largest number of active days.

4.3. Estimation Models with Commits by Categories of Contributors

We investigated the contribution of the number of commits in the estimation of the duration of OSS releases using linear regression since linear regression models are widely used in the software estimation literature [20, 21, 22, 23]. In addition, linear regression models are more suitable for models that use one independent variable. Also, it is much easier to represent the findings graphically and for management to understand.

Three linear regression models were constructed using the total number of commits by category of contributors for each release as the independent variable. The three models correspond to the duration estimation models for full-time, part-time and occasional contributors of the WordPress project.

4.3.1. Data Preparation and Descriptive Statistics

Three major steps are recommended for building estimation models based on historical data: data preparation, application of statistical tools and data analysis. Models built using simple parametric regression techniques require [3]:

- A normal distribution of the input variables (for both the dependent variable (e.g., project duration) and independent variable (e.g., number of commits));
- No outlier that may unduly influence the model;
- A large enough dataset (e.g., typically 20 to 30 data points for each independent variable included in the model).

Here, we focus on the quality of the data, not on its quantity. In order to build good models, we must have good input values: that is the rationale to remove outliers for all categories of contributors to avoid potentially biasing the findings (see the discussion on outliers in chapter 5 of [3]). The total number of commits per category of contributors per release, excluding outliers and the maximum duration per category of contributors per release, where duration is expressed in calendar days (work days) are given in the appendix. There were full-time contributors in 10 releases, part-time contributors in 15 releases and occasional contributors in 21 releases. According to the definitions in the research approach section, there were:

- No full-time contributors in releases 4.2.6, 4.2.8, 4.3.4, 4.5, 4.6, 4.7, 4.8, 4.9, 5.0, 5.1 and 5.2.
- No part-time contributor in releases 4.6, 4.8, 4.9, 5.0, 5.1 and 5.2.

To analyze for the presence of outliers by category of contributors in these new subsets (commits per category of contributors and duration per category of contributors), we applied the Grubbs test on both the independent variable 'total commits per category of contributors per release' and the dependent variable 'maximum duration per category of contributors per release':

- There were no statistical outliers for total commits per category of full-time, part-time and occasional contributors.
- There was one statistical outlier for maximum duration per category of occasional contributors in the release 4.8.

4.3.2. Estimation Models

Table 2 from [24] presents the duration estimation models built on commits per category of contributors for each release and the performance criteria for each model, respectively, for full-time, part-time and occasional contributors:

- The MMRE for full-time, part-time and occasional contributors were 23%, 20% and 49%, respectively.
- In terms of R^2, MMRE and PRED (25%), the duration estimation model for part-time contributors was better compared to the models for full-time and occasional contributors. For instance, the PRED value obtained for full-time contributors is excellent (PRED (25%) = 93%), which means that the MRE was within ± 25% for 93% of the data points.

It is to be noted that the 'x' variable in the equations of Table 2 represents the number of commits. When extrapolations are made from 1/3 and 2/3 of the estimated duration, respectively, for the models for occasional and part-time contributors, the estimates are good compared to the model for full-time contributors [24].

4.4. Discussion of the Estimation Models based on 'Commits' and Industrial Datasets

Table 3 presents the industrial datasets used to compare the duration estimation models with regression techniques. The rationale behind selecting these datasets is related to their significance within the empirical software engineering research: the COCOMO81, Desharnais and Kemerer datasets have been the most widely used in software effort estimation [25].

Table 3. Descriptive statistics of the WordPress and industrial datasets

Datasets	**Number of records**	**Size (measure unit) (independent variable)**	**Measure unit**	**Attribute measured (dependent variable)**
Desharnais	81	Function points (Adjusted)	Month	Duration
Kemerer	15	KSLOC	Month	Duration
COCOMO81	63	KLOC	Month	Duration
WordPress	21	Commits	Workday	Duration

To date, studies by researchers for estimating effort based on substitute variable do not rest on reliable methodological bases [17] and, as long as reliable substitute of effort measures are not available, it is not wise to build estimation models based on these very weak substitutes.

However, for duration estimation this variable is directly calculable from direct data of calendar dates, and our research relates exclusively to duration estimation models.

Table 4 summarizes the performance criteria resulting from the duration estimation models with Desharnais [26], Kemerer [27], COCOMO81 [28] datasets and the duration estimation model with WordPress datasets. The performance criteria of duration estimation models using Desharnais, Kemerer COCOMO81 and WordPress datasets are given in Table 4.

Table 4.Performance of duration estimation models based on WordPress and industrial datasets

Datasets	**Number of records**	**Equation of the models**	**Performance criteria**		
			MMRE (%)	**PRED (25%)**	**R^2**
Desharnais	81	$y = 0.0293x + 2.8687$	49	47	0.51
Kemerer	15	$y = 0.014x + 11.648$	53	40	0.06
COCOMO81	63	$y = 0.0696x + 14.73$	55	37	0.51
WordPress	10 (full-time)	$y = 0.0151x + 14$	23	70	0.36
	15 (Part-time)	$y = 0.0223x + 7$	**20**	**93**	**0.77**
	20 (Occasional)	$y = 0.0392x + 2$	49	20	0.60

In Table 4 the highlighted numbers indicate the most accurate estimates among the four types of datasets. The duration estimation models with the WordPress OSS datasets using 'commits' as the independent variable gave better estimates than duration models built from Desharnais, Kemerer and COCOMO81 datasets with lower MMRE, and highest PRED(25) and R^2.

5. CONCLUSION

One of the factors which influence software project estimation is project data, which is often incomplete and inconsistent. This paper proposes an algorithm to extract relevant data from the GitHub repository for building duration estimation models. Our algorithm written in C includes GitHub commands. Data from WordPress projects, extracted from the source code management repository GitHub, were used to test our algorithm and design the duration estimation models. Statistical regression techniques were used to construct duration estimation models using as the independent variable the number of commits per categories of contributors. We proposed an improved approach which uses the total number of active days for each contributor in the release duration for classifying the contributors as full-time, part-time and occasional. The duration estimation models with the WordPress OSS datasets using 'commits' as the independent variable gave better estimates than duration estimation models built from Desharnais, Kemerer and COCOMO81 datasets. The results show that the duration estimation models built from data collected from Open Source repositories perform as well than those built with datasets most widely used in software duration estimation. All our duration estimation models can be considered white-box models since all the detailed data used to build them are documented and available for independent replication studies.

All empirical studies, such as those reported here, are subject to certain threats to validity. The recommended size of the dataset sample required for meaningful statistical regression results is between 20 to 30 data points. In this study, the sample size used to construct the duration estimation models per category of contributors, with commits as the independent variable, was relatively small (21 data points). We proposed criteria for an objective classification to categorize contributors as full-time, part-time or occasional. However, some assumptions have been made in the design of our duration estimation models. For instance, we assumed that the time spent by each contributor in a release corresponds to the number of days in which the contributor performs at least one commit to the code base.

We plan to build duration estimation models using different statistical techniques and analyze additional OSS projects. In future work, we plan to use some additional factors as criteria to collect data, such as application domain, type of system, development platform, type of language, etc. We also plan to take into account outliers for a more in-depth analysis. Subsequent research could also explore other variables for the estimation of project duration such as algorithm complexity, team experience and code quality, but all these variables together will only help to predict the part of the duration not yet explained by the commits.

REFERENCES

[1] A. Idri, M. Hosni, A. Abran, "Systematic Literature Review of Ensemble Effort Estimation", Journal of Systems & Software, doi: 10.1016/j.jss.2016.05.016, 2016.

[2] S.K. Sehra, Y.S. Brar, N. Kaur, S.S. Sehra, "Research Patterns and Trends in Software Effort Estimation", Information and Software Technology, doi: 10.1016/j.infsof.2017.06.002, 2017.

[3] https://www.isbsg.org/

[4] T. Menzies, B. Caglayan, E. Kocaguneli, J. Krall, F. Peters, and B. Turhan, "The promise repository of empirical software engineering data," 2012. [Online]. Available: http://promise.site.uottawa.ca/SERepository/

[5] A. Abran, Software Project Estimation: The Fundamentals for Providing High Quality Information to Decision Makers. John Wiley & Sons Inc., Hoboken, New Jersey, 2015.

[6] M. Fernãndez-Diego, F.G.-L. De-Guevara, "Potential and limitations of the ISBSG dataset in enhancing software engineering research: A mapping review", Information and Software Technology 56 (6), 527– 544. doi:10.1016/j.infsof.2014.01.003, 2014.

[7] D. Gray, D. Bowes, N. Davey, Y. Sun, and B. Christianson, "Reflections on the NASA MDP data sets", IET Softw. 6 6, 549–558. DOI:10.1049/iet-sen.2011.0132, 2012.

[8] M.J. Shepperd, Q. Song, Z. Sun, and C. Mair, "Data quality: Some comments on the NASA software defect datasets", Software Engineering. IEEE Trans. Softw. Eng. 39, 9, 1208–1215. DOI:10.1109/TSE.2013.11, 2013.

[9] K. Deng and S.G. MacDonell, "Maximising data retention from the isbsg repository", 12th International Conference on Evaluation and Assessment in Software Engineering. Italy. 2008.

[10] F. Qi, X Y. Jing, X. Zhu, et al., "Software effort estimation based on open source projects: Case study of Github", Information and Software Technology, 92: 145-157, 2017.

[11] A.S. Badashian, A. Esteki, A. Gholipour, A. Hindle, and E. Stroulia, "Involvement, contribution and influence in github and stackoverflow", 24th Annual International Conference on Computer Science and Software Engineering. Markham, Ontario, 2014.

[12] M.F. Bosu and S. G. Macdonell, "Experience: Quality Benchmarking of Datasets Used in Software Effort Estimation", Journal of Data and Information Quality, vol. 11, n° 4, Article 19, 38 pages, 2019. https://doi.org/10.1145/3328746

[13] M.F. Bosu and S.G. MacDonell, "A taxonomy of data quality challenges in empirical software engineering", 22nd Australian Conference on Software Engineering.97–106.2013a, DOI:10.1109/ASWEC.2013.21.

[14] C. Gencel, L. Buglione, and A. Abran, "Improvement opportunities and suggestions for benchmarking", Software Process and Product Measurement.Springer, Berlin (Germany), 144–156, 2009.

[15] L. Cheikhi and A. Abran, "Promise and ISBSG software engineering data repositories: A survey", In Joint Conference of the 23nd International Workshop on Software Measurement and the 8th International Conference on Software Process and Product Measurement, 17–24, 2013, DOI:10.1109/IWSM-Mensura.2013.13

[16] E. Kocaguneli, T. Menzies, "How to find relevant data for effort estimation?",International Symposium on Empirical Software Engineering and Measurement, Banff, Canada, 2011, pp. 255–264. doi:10.1109/ESEM. 2011.34.

[17] E. Shihab, Y. Kamei, B. Adams, et al., "Is lines of code a good measure ofeffort in effort-aware models?", Information and Software Technology, 55(11): 1981-1993, 2013.
[18] S. Koch, "Effort Modeling and Programmer Participation in Open Source Software Projects",Information Economics and Policy, 20(4), pp.345-355, 2008.
[19] G. Robles, J.M. González-Barahona, C. Cervigón, A. Capiluppi, D. Izquierdo-Cortázar, "Estimating Development Effort in Free/Open Source Software Projects by Mining Software Repositories: A Case Study of OpenStack", 11th Working Conference on Mining Software Repositories, May 31 – June 01 2014, pp.222-231.
[20] S. Chatterjee and A.S. Hadi, Regression analysis by example, John Wiley & Sons, 2015.
[21] J. Fernandez-Ramil, D. Izquierdo-Cortazar, and T. Mens, "What does it take to develop a million lines of Open Source code?", In: Boldyreff C., Crowston K., Lundell B., Wasserman A.I. (eds) Open Source Ecosystems: Diverse Communities Interacting. OSS 2009. IFIP Advances in Information and Communication Technology, Springer, Berlin, Heidelberg, vol 299, pp. 170-184, 2009.
[22] M. Jorgensen, "Regression models of software development effort estimation accuracy and bias", Empirical Software Engineering, vol. 9, pp. 297–314, 2004.
[23] D.C. Montgomery, E. A. Peck, G. G. Vining, Introduction to Linear Regression Analysis, 5th ed. Wiley, Hoboken, NJ. 2012.
[24] D.K. Moulla, A. Abran and Kolyang, "Duration Estimation Models for Open Source Software Projects", International Journal of Information Technology and Computer Science (IJITCS), Vol.12, No.5, In production.
[25] C. Mair, M.J. Shepperd and M. Jørgensen, "An analysis of data sets used to train and validate cost prediction systems", Workshop on Predictor Models in Software Engineering (PROMISE'05).1–6, 2005. DOI:10.1145/1083165.1083166.
[26] J.M. Desharnais, Analyse Statistique de la productivité des projets informatique partir de la technique des Point des Fonction, Université de Montréal (Canada), Masters thesis, 1989.
[27] Chris F. Kemerer, "An empirical validation of software cost estimation models",Communication of the ACM, Vol.30, No.5, pp.416–429, 1987.
[28] J. SayyadShirabad, T.J. Menzies, "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.

AUTHORS

Dr.Donatien K. Moulla is currently a Lecturer in computer science at the Faculty of Mines and Petroleum Industries, University of Maroua, Cameroon. He received his PhD in Computer Science from the University of Ngaoundéré, Cameroon. He earned his Master's and Bachelor's degrees from the Department of Mathematics and Computer Science in the same University. He has more than eight years of teaching and research experience in information systems development and Software Engineering. He is also member of COSMIC International Advisory Council. His research interest includes software estimation, software measurement, software quality, Automation of COSMIC functional size measurement and Open Source Software projects.

Alain Abran, PhD, is a Professorat École de technologie supérieure, Université du Québec, Canada. He is also Chairman of the Common Software Measurement International Consortium. He was the international secretary for ISO/IEC JTC1 SC7. Dr. Abran has over 20 years of industry experience in information systems development and software engineering.

Prof. Dr.-Ing. habil. Kolyang is a Professor at the Higher Teachers' Training College, University of Maroua, Cameroon. His research area includes Software Engineering, E-learning, and ICT for Development.

APPENDIX A

Code

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define LG_LIGNE     512
int main(int argc, char* argv[]){
if(argc == 2){
printf("number of arguments not sufficient\n");
return 0;
}
char ligne [LG_LIGNE];
char* auteur=(char*)malloc(sizeof(char*)*50);
char* nb_commit=(char*)malloc(sizeof(char*)*50);
char* token=(char*)malloc(sizeof(char*)*50);
char cmd[LG_LIGNE];
if(argc >= 2)
snprintf(cmd, sizeof(cmd), "git shortlog -sn --since=%s --
before=%s | awk '{print $2, $3}' | sort -
d > tmp.txt", argv[1], argv[2]);
else
snprintf(cmd, sizeof(cmd), "git shortlog -
sn | awk '{print $2, $3}' | sort -d > tmp.txt");
FILE * f = popen(cmd, "r");
pclose(f);
FILE * fichier = fopen("tmp.txt", "r");
FILE * result = fopen("resultats.txt", "w+");
while (fgets (ligne, LG_LIGNE, fichier) != NULL) {
auteur = strtok_r(ligne, "\n", &token);
printf("%s\n", auteur);
fprintf(result, "%s\t", auteur);
char cmd[LG_LIGNE];
//number of commits
if(argc >= 2)
snprintf(cmd, sizeof(cmd), "git shortlog -sn --author=\"%s\" --
since=\"%s\" --
before=\"%s\" | awk '{print $1}' ", auteur, argv[1], argv[2]);
else
snprintf(cmd, sizeof(cmd), "git shortlog -sn --
author=\"%s\" | awk '{print $1}' ", auteur);

char commit[20];
FILE * ct = popen(cmd, "r");
fgets(commit, 20, ct);
fprintf(result, "%d\t", atoi(commit));
pclose(ct);
//number of insertion
        if(argc >= 2)
            snprintf(cmd, sizeof(cmd), "git log --stat --author=\"%s\" -
-since=\"%s\" --
before=\"%s\" | grep insertions | awk 'BEGIN{ som=0 } { som=som+$4} END{
 print som }' ", auteur, argv[1], argv[2]);
        else
            snprintf(cmd, sizeof(cmd), "git log --stat --
author=\"%s\" | grep insertions | awk 'BEGIN{ som=0 } { som=som+$4} END{
 print som }' ", auteur);
        char insert[20];
        FILE * fi = popen(cmd, "r");
        fgets(insert, 20, fi);
        fprintf(result, "%d\t", atoi(insert));
        pclose(fi);

        //number of deletions
        if(argc >= 2)
            snprintf(cmd, sizeof(cmd), "git log --stat --author=\"%s\" -
-since=\"%s\" --
before=\"%s\" | grep deletions | awk 'BEGIN{ som=0 } { som=som+$(NF-
1)} END{ print som }' ", auteur, argv[1], argv[2]);
        else
            snprintf(cmd, sizeof(cmd), "git log --stat --
author=\"%s\" | grep deletions | awk 'BEGIN{ som=0 } { som=som+$(NF-
1)} END{ print som }' ", auteur);
        char delete[20];
        FILE * fd = popen(cmd, "r");
        fgets(delete, 20, fd);
        fprintf(result, "%d\t", atoi(delete));
        pclose(fd);

        //number of active days
        if(argc >= 2)
            snprintf(cmd, sizeof(cmd), "git log --stat --author=\"%s\" -
-since=\"%s\" --before=\"%s\" --date=short | grep Date | sort -u | sed -
n '$=' ", auteur, argv[1], argv[2]);
        else
            snprintf(cmd, sizeof(cmd), "git log --stat --author=\"%s\" -
-date=short | grep Date | sort -u | sed -n '$=' ", auteur);

        char date[20];
        FILE * dt = popen(cmd, "r");
        fgets(date, 20, dt);
        fprintf(result, "%d\n", atoi(date));
        pclose(dt);
    }
    FILE * rm = popen("rm  tmp.txt ", "r");
    pclose(rm);
    fclose(result);
    fclose(fichier);

    return EXIT_SUCCESS;}
```

APPENDIX B

WordPress datasets

Table 1. WordPress project: commits per category of contributors
N= 21 releases - excluding outliers

Release	Total number of commits per category of contributors		
	Full-time contributors	Part-time contributors	Occasional contributors
4.2.4	95 (2)	21 (2)	111 (7)
4.2.5	554 (2)	531 (6)	305 (7)
4.2.6	-	81 (3)	90 (9)
4.2.7	35 (1)	51 (1)	142 (16)
4.2.8	-	570 (8)	116 (7)
4.3	251 (2)	328 (4)	297 (11)
4.3.1	218 (3)	49 (1)	163 (11)
4.3.2	905 (5)	154 (5)	41 (5)
4.3.3	500 (4)	259 (5)	196 (7)
4.3.4	-	328 (7)	231 (22)
4.4.1	94 (2)	237 (6)	77 (13)
4.4.2	51 (1)	280 (6)	187 (13)
4.4.3	287 (2)	351 (5)	168 (14)
4.5	-	196 (3)	291 (21)
4.6	-	-	275 (23)
4.7	-	431 (5)	677 (27)
4.8	-	-	592(30)
4.9	-	-	306(26)
5.0	-	-	403(29)
5.1	-	-	197(20)
5.2	-	-	101(20)
Average	**299**	**258**	**236**
Standard deviation	**278**	**171**	**162**

In Table 1, the number of contributors per release is in parenthesis.

Table 2. WordPress project: duration per category of contributors
N= 21 releases - excluding outliers

Release	Maximum duration per category of contributors			Total duration per release (in work days)
	Full-time contributors (in work days)	Part-time contributors (in work days)	Occasional contributors (in work days)	
4.2.4	8	3	2	8
4.2.5	33	20	9	33
4.2.6	-	10	5	14
4.2.7	13	8	6	19
4.2.8	-	16	5	24
4.3	21	17	10	31
4.3.1	17	8	6	20
4.3.2	19	9	5	19
4.3.3	26	12	6	26
4.3.4	-	14	8	24
4.4.1	17	12	6	21
4.4.2	14	11	5	19
4.4.3	19	14	5	23
4.5	-	14	8	27
4.6	-	-	21	90
4.7	-	52	25	79
4.8	-	-	41	132
4.9	-	-	20	114
5.0	-	-	27	280
5.1	-	-	12	55
5.2	-	-	13	53

Table 3. WordPress project: Actual and estimated total duration per release per occasional contributors (with extrapolation)

Release	Total commits for all occasional contributors per release	Actual duration (in work days)	Estimated duration (in work days)	|RE|
4.2.4	111	8	19	1.38
4.2.5	305	33	42	0.27
4.2.6	90	14	17	0.18
4.2.7	142	19	23	0.19
4.2.8	116	24	20	0.18
4.3	297	31	41	0.32
4.3.1	163	20	25	0.26
4.3.2	41	19	11	0.43
4.3.3	196	26	29	0.12
4.3.4	231	24	33	0.38
4.4.1	77	21	15	0.28
4.4.2	187	19	28	0.47
4.4.3	168	23	26	0.12
4.5	291	27	40	0.49
4.6	275	90	38	0.57
4.7	677	79	86	0.08
4.8	592	132	76	0.43
4.9	306	114	42	0.63
5.0	403	280	53	0.81
5.1	197	55	29	0.47
5.2	101	53	18	0.66
The MMRE for the model = 42%				
The median |RE| = 38%				

Table 4. WordPress project: Actual and estimated total duration per release per part-time contributors (with extrapolation)

Release	**Total commits for all part-time contributors per release**	**Actual duration (in work days)**	**Estimated duration (in work days)**	**|RE|**
4.2.4	21	8	9	0.09
4.2.5	531	33	37	0.13
4.2.6	81	14	12	0.14
4.2.7	51	19	10	0.45
4.2.8	570	24	40	0.65
4.3	328	31	26	0.16
4.3.1	49	20	10	0.49
4.3.2	154	19	16	0.15
4.3.3	259	26	22	0.15
4.3.4	328	24	26	0.08
4.4.1	237	21	21	0.01
4.4.2	280	19	23	0.22
4.4.3	351	23	27	0.18
4.5	196	27	19	0.31
4.6	-	90	8	0.92
4.7	431	79	32	0.60
4.8	-	132	8	0.94
4.9	-	114	8	0.93
5.0	-	280	8	0.97
5.1	-	55	8	0.86
5.2	-	53	8	0.86
The MMRE for the model = 44%				
The median |RE| = 31%				