Duration Estimation Models for Open Source Software Projects

Donatien Koulla Moulla[1,2]
[1] Faculty of Mines and Petroleum Industries, University of Maroua, Maroua, P.O. Box 46, Cameroon
[2] LaRI Lab, University of Maroua, Maroua, P.O. Box 814, Cameroon
E-mail: moulladonatien@gmail.com, donatien-koulla.moulla@univ-maroua.cm

Alain Abran
Department of Software Engineering and Information Technology, École de Technologie Supérieure, 1100, rue Notre-Dame Ouest, Montréal, Québec, Canada H3C 1K3
E-mail: alain.abran@etsmtl.ca

Kolyang
The Higher Teachers' Training College, University of Maroua, Maroua, P.O. Box 46, Cameroon
E-mail: kolyang@cde-saare.de

Abstract: For software organizations that rely on Open Source Software (OSS) to develop customer solutions and products, it is essential to accurately estimate how long it will take to deliver the expected functionalities. While OSS is supported by government policies around the world, most of the research on software project estimation has focused on conventional projects with commercial licenses. OSS effort estimation is challenging since OSS participants do not record effort data in OSS repositories. However, OSS data repositories contain dates of the participants' contributions and these can be used for duration estimation. This study analyses historical data on WordPress and Swift projects to estimate OSS project duration using either commits or lines of code (LOC) as the independent variable. This study proposes first an improved classification of contributors based on the number of active days for each contributor in the development period of a release. For the WordPress and Swift OSS projects environments the results indicate that duration estimation models using the number of commits as the independent variable perform better than those using LOC. The estimation model for full-time contributors gives an estimate of the total duration, while the models with part-time and occasional contributors lead to better estimates of projects duration with both for the commits data and the lines of data.

Index Terms: Data Repositories, Duration Estimation, Estimation Models, Open Source Software Project Estimation, Regression Models.

1. Introduction

Open Source Software (OSS) provides significant technical and economic benefits for a multitude of organizations, large and small, that rely on OSS to develop customer solutions and products [1,2,3]. Estimation, in particular, is challenging for OSS projects due to the collaborative nature of OSS, which is developed and maintained by distributed communities of contributors from all over the world. According to Asundi [4], planning and delivering projects based on an Open Source (OS) distributed community is challenging since resource allocation and budgeting lack a rigorous basis: there is generally no formal project management, budget or schedule. For this reason, he argues that using existing effort estimation models for OSS projects has many disadvantages and therefore, new models are needed. In particular, he notes that information about when functionalities will be available is of major interest to organizations that have adopted OSS as their business strategy.

There is a large number of studies on software effort estimation but very few discuss effort and duration in OSS projects. Effort data is not available in OSS repositories and the categorization of contributors and the size and choice of the datasets also present a challenge for research on OSS projects estimation. To date, in OSS effort estimation studies researchers have had to use substitutes for the missing effort variable, which substitutes had a very weak methodological basis. However, for the duration estimation, this variable can be directly computed from projects calendar dates. Therefore, this research investigates exclusively OSS duration estimation models.

An OSS source code management repository is an online community platform that makes project history data

available to IT practitioners and researchers, enabling developers to host and review code, manage projects, share knowledge and work together to build OSS. There is no direct information in the OSS code repository to identify whether the contributors are working full-time, part-time or occasionally on the OSS project, and therefore no direct information on the total effort of any specific project. Due to the amount and diversity of the raw data (such as commits, contributors, LOC, etc.), extracting and processing the data for either effort or duration estimation is a challenge. These repositories usually contain the following information:

- Delivered software source code with corresponding dates.
- Commits of the contributors who submitted their codes to a project.
- 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 and evaluation of duration estimation models for OSS projects using statistical regression analysis with four measures of activity 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, where an active day correspond to the day 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 of WordPress and Swift projects, extracted from the source code management repository GitHub, were used to design the duration estimation models. A decade after GitHub's official launch, its community has grown to 27 million of developers working all over the world on more than 80 million projects [5]. The rationale behind selecting the WordPress and Swift projects is related to their OSS economic model, data availability and their 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.
- Swift is an Apple Project: it is a programming language created by Apple for building apps for iOS, Mac, Apple TV, and Apple Watch. It has been available in Open Source under Apache license since December 3, 2015. In addition to the data availability, Swift has attracted many OSS contributors.

For software organizations that rely on OSS to develop customer solutions and products, it is important to estimate how long it will take to deliver the expected functionalities. This paper analyzes historical data on WordPress and Swift projects to design duration estimation models of OSS releases using either commits or lines of code (LOC) as the independent variable. This will allow to quantify the impact of 'commits' and 'LOC'as an important variable for estimating the project duration, thereby providing a decision support model for organizations that use OS in software development.

This paper is structured as follows. Section 2 presents related work on OSS effort and duration estimation. Section 3 presents the research approach. Section 4 presents the duration estimation models built using data from the WordPress project and, Section 5, using data from the Swift project. Section 6 concludes with a summary of the key findings, the study limitations and directions for future work.

Supporting documentation is provided in an Electronic Appendix.

2. Related Work

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

Traditional effort estimation models are generally used by organizations in the early stages of a project to estimate total effort, number of developers and time (e.g., duration) needed to develop the software. Software effort estimation is an important step for developing software projects, and is of interest for researchers and software organizations [6,7]. However, most research to date on effort and duration estimation has focused on projects based on proprietary commercial licenses, while few studies have taken into account the characteristics of software built using OS. An example of the latter is the study by Qi et al. [8] who provided a method which can be used to collect sufficient data for solving the problem of lack of training data when building an effort estimation model. In their work, they proposed a method to alleviate the data deficiency problem by 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 definition: in a period of consecutive active days (with a of default 30 days), if a contributor has an average of one committing record per active day, and whose total contributions exceed the average contributions of the project, they call him as active contributor, otherwise they call him an inactive contributor.

Shihab et al. [9] 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 JIRA database. They used correlation tests and linear regression models to investigate the categories of variables as a substitute to 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 were 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 (eg., from issues to files): of course, doing so introduces a number of unknown distortions without being able quantify them and analyze their impact on the findings reported. While the authors report that there are correlations between the indirect effort-measures with actual effort, such results reported are so low, this provides little support on the relevance of the indirect measures of effort in other related works for OSS effort estimation purposes.

Robles et al. [10] present an approach using data from repositories of OS code. In their study, they proceed by identifying contributors as full-time, part-time or occasional, and evaluating their activity using, for each contributor, the following criteria:

- Number of commits merged into the code base during a given period.
- Number of active days during a given period, considered as days in which a contributor performs at least one commit to the code base.

Their model is based on finding a threshold value for the number of commits (or active days) and whether contributors are full-time, part-time or occasional. For the calculation of a threshold, they used a survey and data from over a hundred OpenStack developers. They assume that full-time contributors devote to the project around 1 person-month (around 40h / week) and those contributors who devote less than 40h / week are considered as part-time / occasional contributors. In this study the OSS contributors are categorized as full-time, part-time or occasional, based on the total number of active days by each contributor in the release duration.

Amor et al. [11] proposed approximating the total effort in a project by characterizing the activity of developers by the versioning system, mail, bug tracking system, etc. In their study, effort is an indirect measure of effort derived through the approximation of revision control data, participation in mailing list and bug tracking activity. Kalliamvakou et al. [12] replicated the study and included the developer contribution to the software development process, based on data acquired from software repositories and collaboration infrastructures.

Capiluppi et al. [13] determined for the Linux kernel developers the average hours worked per day by the time of the day when the commits were observed in the repository and when the author carried out the commits most frequently. They divided a working day into traditional office hours (from 9am to 5pm), after office hours (5pm to 1am) and late hours (from 1am to 8am). The authors found that within the Linux kernel community the effort derived from the day when the commits were carried out was relatively the same throughout the week, which suggested the need for different estimation models from the ones traditionally used in industrial settings, where the work schedule is presumed to be 9am-5pm, Monday to Friday. When they proposed their methodology, it could not be applied in a software configuration management (SCM) repository such as the concurrent versioning system (CVS) since, although the time the commit was submitted is stored, the time when the author changed the source code is not. CVS is now seldomly used. They focused their analysis only on the Linux Kernel community and their results cannot be generalized to other OSS projects.

Mockus et al. [14] showed that of the nearly 400 programmers in the Apache project, only 15 of them contributed 88 per cent of the total lines of code (LOC). They compared these 15 Apache developers to programmers using LOC in five other commercial projects. They defined code productivity as the mean number of lines of code per developer per year (KLOC / developer / year). Koch [15,16] reported that in OSS projects, the distribution of lines of code between participants (programmers) is skewed as in Mockus et al. [14]: indeed, the majority of code was written and most of the contributions to the code base done only by a few contributors.

Moulla et al. [17,18] applied the COCOMO model to the TRIADE (version 7a) OSS project using LOC [19] and regression models with COSMIC Function Points as the independent variable [20,21]. They reported that in terms of effort, development of software based on OSS has advantages over development from scratch. Here, in contrast to other studies, 'effort' data was directly collected and available.

Fernandez-Ramil et al. [22] used linear regression models to estimate effort and duration for OSS projects with LOC as the independent variable. Effort was derived based on the number of active committers per month. To

operationalize effort variable, they determined the number of different person ids contributing to its repository over a given month. This number of ids was assumed to be the number of person-months in a given month. They studied the relationship between LOC on the one hand, and derived effort, duration and team size on the other hand for 11 OSS projects. Their study compared OSS projects to proprietary systems and may help understanding and plan better the evolution of OSS communities and their projects.

Yu [23] also used linear regression models with LOC as the independent variable to predict indirectly maintenance effort in free and OSS projects: the lag time to perform maintenance task and source code change was used as a substitute measure of maintenance effort.

Anbalagan et al. [24] used bug reports corrected by developer as a substitute measure of effort in corrective maintenance of free and OSS projects. Their study focused on 72,482 bug reports from nine releases of Ubuntu, a Linux distribution system. They used a linear regression model with 'number of participants' as the independent variable to predict indirectly the effort taken to correct bugs.

In contrast to Kalliamvakou et al. [12], who considered only LOC, in our study the contribution of developers was derived from either LOC or commits. Moreover, we propose in our study reported here to estimate duration while [11] and [12] attempted to indirectly approximate effort using developer activities. We used linear regression models as in Fernandez-Ramil et al. [22] but while they estimated effort and duration using LOC as the independent variable, we also estimated duration using commits as the independent variable.

In summary, some studies are available on software effort estimation but few discuss effort and duration in OSS projects. The categorization of contributors and the size and choice of the datasets also present a challenge for research on OSS projects estimation. The keys findings of these specific studies show that the related works suffer from a lack of data on effort itself and have to approximate it indirectly through other variables and without knowing their variability and the impact of such variability on the estimation models themselves.

To date, studies by researchers for estimating effort based on substitute variable do not rely on reliable methodological bases [9] 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 the duration estimation, this variable is directly calculable from direct data of calendar dates, and our research relates exclusively to duration estimation models.

3. Research Approach

This paper investigates the contribution of the number of commits and LOC in the estimation of the duration of OSS releases using linear regression. The rationale behind the use of linear regression models is that in the software estimation literature, they are widely used [22], [25, 26, 27, 28]. In addition, linear regression models are more suitable for models that use one independent variable. Also, it is easier for management to understand and much easier to represent the findings graphically.

It has been observed that contributions to OSS projects are not equally distributed across contributors: a few are responsible for the majority of the commits, while a large number contribute only a few [14], [16]. In our study, instead of using a threshold [10], we propose an improved approach by 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.

The data were extracted from the GitHub repository using a program written in C including GitHub commands. For the purpose of estimating OSS project duration, we defined the following:

i) Active day for a contributor: it is defined as a day in which the contributor performs at least one commit to the code base.

ii) Release duration is calculated from the time it took 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, if 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 release duration 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 release duration of this category corresponds to the largest number of active days of this category.

To analyze the relationships among the variables in estimation models built from a dataset, Conte et al. [29]

recommended a number of criteria:

- The coefficient of determination (R^2): it describes the percentage of variability explained by the independent variable. An R^2 close to 1 indicates that the variability of response to the independent variable is consistent with the model (e.g., there is a strong relationship between the independent and dependent variables).
- Magnitude of relative error (MRE): gives an indication of the divergence between values estimated by the model and the actual values, expressed as a percentage. MRE is given by the relation *MRE=|RE|=|(Actual-Estimate)/Actual|*. Perfect estimates would give an MRE of 0%. RE is the relative error. This relative error can be either positive or negative, representing either an overestimation or an underestimation by the model.
- MMRE: is the mean magnitude of relative errors across all the data points.
- Prediction level of the model: is given by *PRED(l) = k/n* where *k* is the number of projects in a specific sample size *n* for *MRE ≤ l*. For example, PRED (20%) = 80% means that MRE (Mean Relative Error) is within $\pm$ 20% for 80% of the data points.

The selected research approach is the engineering approach. To build duration estimation models, our research approach is based on:

- Observation of past projects and quantitative data collection;
- Analysis of the impact of individual variables, one at a time;
- Selection of relevant samples, and of samples of sufficient size from a statistical viewpoint.

4. Model of Duration Estimation from the WordPress OSS project Repository

This section presents the duration estimation models built using data from the WordPress project repository.

4.1. Source of data

The WordPress Open Source Content Management System (CMS) has been widely adopted and is used for creating and managing most of the websites in the world [30]. The first version (0.7) was published on May 27, 2003. For the development of our OSS duration estimation models, a statistical analysis of historical data, from February 2015 to May 2019, was carried out on 21 releases of the WordPress project based on data extracted on June 23, 2019 from the GitHub repository (available online from https://github.com/WordPress/WordPress).

Three measures of activity, as presented in the Introduction section, were used for the analysis: number of contributors, total number of commits and LOC per contributor. These data were extracted from the GitHub repository using a program written in C including GitHub commands. These data are presented in the Electronic Appendix.

4.2. Duration estimation models with commits as independent variable

The linear regression models for duration estimation in terms of active days (the estimated dependent variable) were constructed using as the independent variable:

A. The total number of commits per contributor per release.
B. The total number of commits per category of contributors per release.
   A. Twenty one (21) linear regression models corresponding to the 21 releases of the WordPress project were constructed using the total number of commits per contributor per release as the independent variable.
   B. Next, 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 estimation models for full-time, part-time and occasional contributors of the WordPress project.

A. 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. The models built using simple parametric regression techniques require [31]:

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

Values of input data that are significantly far from the average of the dataset population (e.g., the candidate outliers) must be verified [31]. Here, we focus on the quality of the data, not on their 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 in Abran [31]). The presence of outliers in each release was analyzed using the Grubbs test for both the independent variable 'commit' and dependent variable 'duration' and the statistical tool R was used to carry out the Grubbs test. The key parameters of the Grubbs test used to remove outliers were: a significant level (p-value) equal to 0.05 and a removing rate equal to 5%. Outliers for the independent variable 'commit per contributor per release' were identified and removed in 11 releases (4.2.6, 4.2.7, 4.2.8, 4.3.1, 4.3.3, 4.6, 4.8, 4.9, 5.0, 5.1 and 5.2). For the descriptive statistics, we took into account the number of contributors per release, the total number of commits per release, the total duration per release in calendar days of work. The descriptive statistics of the 11 releases that contained an outlier are presented in electronic appendix to this article.

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 also given in the electronic 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.

B. Estimation models without outliers

This section introduces the experiment expectations, environment and process. The proposed duration estimation models are intended to serve as a decision support tool for any organization developing software systems based on Open Source. Our duration estimation models can be applied for projects similar to those in the case study and are intended for both practitioners and industry. To ensure the accuracy and reliability of the results, we used the criteria recommended by Conte et al. [29], mentioned in the "Research Approach" section, paragraph 5 (page 4). The data, the models themselves, and their performance are available. The duration estimation models built on the WordPress datasets can be considered as white-box models, since all the detailed data used to build them are documented and available for independent replication studies. There is no model today that can perfectly replicate the size–duration relationship, and there remains uncertainty in the duration estimation model itself. The MMRE of the duration models are given in the Table 1.

The input variables to the linear regression model for each release are the independent variable (total number of commits per contributor) and the dependent variable (active days per contributor), while the output is the linear equation. Figure 1 presents the graph of the best results with the 'commits' for Release 5.2, and Figure 2, for release 4.2.4 which is not very good.
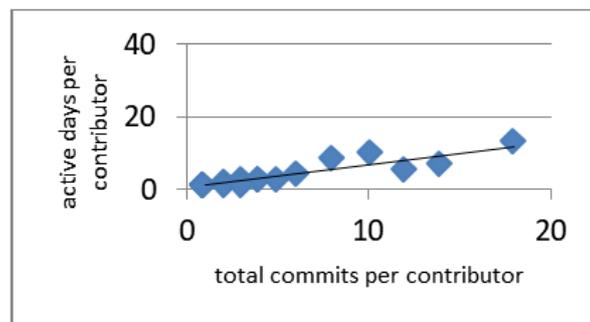

Fig.1. WordPress project: Duration estimation model for release 5.2 (N=20)
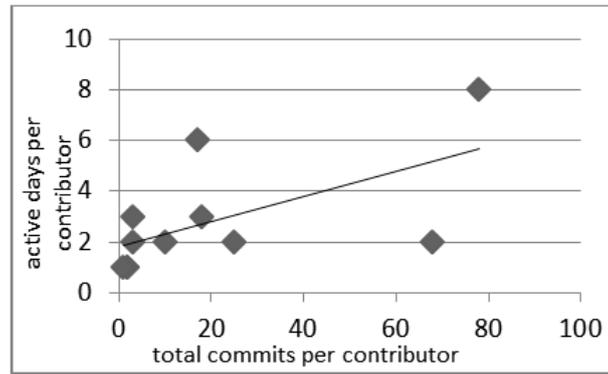
$$y = 0.6339 + 0 \qquad (1)$$

Fig.2. WordPress project: Duration estimation model for release 4.2.4 (N=11)

$$y = 0.0496 + 2 \tag{2}$$

Table 1 gives the equation of the model related to each release and the corresponding criteria recommended by Conte et al. [29] of each model.

Table 1. WordPress project: estimation models based on commits per release (excluding outliers)

Release	Size range [in commits] per release	Total number of contributor per release	Equation of the model	R^2	MMRE	PRED (20%)
4.2.4	[1, 78]	11	$y = 0.0496x + 2$	0.37	2.10	36
4.2.5	[2, 326]	15	$y = 0.0683x + 6$	0.53	0.90	67
4.2.6	[1, 28]	12	$y = 0.2585x + 0$	0.79	0.36	67
4.2.7	[1, 51]	18	$y = 0.18x + 1$	0.71	0.28	61
4.2.8	[1, 171]	15	$y = 0.0839x + 5$	0.61	0.62	73
4.3	[2, 161]	17	$y = 0.0982x + 5$	0.57	0.55	82
4.3.1	[3, 80]	15	$y = 0.1712x + 2$	0.87	0.35	80
4.3.2	[1, 330]	15	$y = 0.046x + 5$	0.77	0.55	67
4.3.3	[7, 206]	16	$y = 0.1154x + 2$	0.85	0.45	69
4.3.4	[1, 83]	29	$y = 0.1412x + 2$	0.60	0.71	72
4.4.1	[1, 63]	21	$y = 0.2156x + 2$	0.82	0.58	86
4.4.2	[1, 94]	20	$y = 0.0794x + 3$	0.34	0.82	70
4.4.3	[1,171]	21	$y = 0.1215x + 2$	0.90	0.44	81
4.5	[1, 76]	24	$y = 0.1442x + 2$	0.71	0.56	75
4.6	[1, 30]	23	$y = 0.6452x + 1$	0.92	0.25	96
4.7	[1, 116]	32	$y = 0.3563x + 3$	0.81	0.63	78
4.8	[1, 82]	30	$y = 0.4532x + 2$	0.92	0.52	83
4.9	[1, 35]	26	$y = 0.593x + 1$	0.90	0.37	96
5.0	[]1, 49]	29	$y = 0.5352x + 1$	0.95	0.33	93
5.1	[1, 29]	20	$y = 0.3442x + 1$	0.62	0.57	75
5.2	[1, 18]	20	$y = 0.6339x + 0$	0.84	0.11	50
The mean MMRE = 57%						
The median MMRE = 55%						

The second column in Table 1 indicates, for each release, the size ranges of the data from which the regression model was built and for which the model is valid: that is, the range of values for the independent variable 'x' in terms of minimum to maximum. Table 1 shows that the estimation models:

- for nine releases have both an R^2 over 0.7 and a PRED (20%) for over 70% of the data points.
- for eight of the releases, an increase in the commits explains over 80% of the increase in duration (e.g., R^2 over 0.80), and
- for 5 of these releases, 90% or more.

I.J. Information Technology and Computer Science, 2021, 1, 1-17

The MMRE:

- for 5 of these is $\leq 57\%$ for 14 releases, and is greater than 57% for seven releases.

4.3. Estimation models with commits by categories of contributors

A. Estimation models

Table 2 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 24%, 30% and 49%, respectively.
- In terms of R^2, MMRE and PRED (20%), the estimation model for full-time contributors was better compared to the models for part-time and occasional contributors. For instance, the PRED value obtained for full-time contributors is excellent (PRED (20%) = 90%): this means that MRE was within $\pm 20\%$ for 90% of the data points.

Table 2. WordPress project: estimation models based on commits per category of contributors

	Full time (N=10)	Part-time (N=15)	Occasional (N=20)
Equation of the models	$y = 0.0151x + 14$	$y = 0.0375x + 5$	$y = 0.0392x + 2$
R^2	0.36	0.33	0.60
MMRE	0.24	0.30	0.49
PRED (20%)	90	93	75

However, it has to be noted that the model with full-time contributors gives an estimate of the total duration, while the other two models give an estimate of their respective dependent variable of either part-time or occasional contributors which, by definitions correspond, to at most 2/3 and 1/3 of the total duration, respectively. For example, for the 4.3.1 release, the actual duration is 20 work days (see Table 3):

- the model for full-time contributors estimates the total duration which is 20 days;
- the models for part-time contributors predict at most 2/3 of the total duration which is 20 x 2/3 days;
- the models for occasional contributors predict at most 1/3 of the total duration which is 20 x 1/3 days.

Therefore, these models (equations $y = 0.0392x + 2$ and $y = 0.375x + 5$ from Table 2) are useful in practice for estimation of total duration of the release provided that an extrapolation is made to represent the full release duration:

- for occasional contributors the extrapolation is determined by multiplying the estimated duration by 3 (e.g., 3 x equation $y = 0.0392x + 2$ - from Table 2),
- for the part-time contributors the extrapolation is determined by multiplying the estimated duration by 3/2 (e.g., 3/2 x equation $y = 0.375x + 5$ - from Table 2).

It is to be noted also that the relationships between the equation of the model column of Table 1 and the equation of the models of full-time, part-time and occasional columns in Table 2 is that the 'x' variable in the equations of Tables 1 and 2 represents the number of commits.

B. Comparison with actual duration – for models with full-time contributors

Table 3 presents a comparison between the actual duration per release and the estimated duration by the model for full-time contributors. The independent variable is the total commits for all full-time contributors per release. The estimated duration for each release is obtained by replacing the variable 'x' in the equation $y = 0.0151x + 14$ (from Table 2) by the total number of commits for all full-time contributors for this release.

Table 3. WordPress project: actual and estimated duration for full-time contributors (N=10 releases)

Release	Total commits for all full-time contributors per release	Actual duration (in work days)	Estimated duration (in work days)	|RE|
4.2.4	95	8	15	0.93
4.2.5	554	33	22	0.32
4.2.6	-	14	14	0.00
4.2.7	35	19	15	0.24
4.2.8	-	24	14	0.42
4.3	251	31	18	0.43
4.3.1	218	20	17	0.14
4.3.2	905	19	28	0.46
4.3.3	500	26	22	0.17
4.3.4	-	24	14	0.42
4.4.1	94	21	15	0.27
4.4.2	51	19	15	0.22
4.4.3	287	23	18	0.20
4.5	-	27	14	0.48
4.6	-	90	14	0.84
4.7	-	79	14	0.82
4.8	-	132	14	0.89
4.9	-	114	14	0.88
5.0	-	280	14	0.95
5.1	-	55	14	0.75
5.2	-	53	14	0.74
The MMRE for the model = 50%				
The median |RE| = 43%				

Since 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 have no commit for full-time contributors, the estimate is the same for each model: here, the estimated duration corresponds to the constant of the estimation equation (e.g., 14 days). Of course, this means that when there is no full-time contributor, such a model does not predict well. The estimation model per category of full-time contributors had an MRE ≤ 50% for 13 releases (4.2.5, 4.2.6, 4.2.7, 4.2.8, 4.3, 4.3.1, 4.3.2, 4.3.3, 4.3.4, 4.4.1, 4.4.2, 4.4.3 and 4.5) where 50% represents the MMRE of the estimation model per category of full time contributors. The median estimation error = 43%. For only the releases with full-time contributors, the estimation per category of full-time contributors had an MRE ≤ 34% for 7 releases (4.2.5, 4.2.7, 4.3.1, 4.3.3, 4.4.1, 4.4.2 and 4.4.3).

C. Estimation models per category of occasional contributors

The estimation models for occasional and part-time contributors predict at most 1/3 and 2/3, respectively, of the total duration since these two categories of contributors are defined as such (see section 3). However, these models (e.g., equation $y = 0.0392x + 2$ and equation $y = 0.375x + 5$ from Table 2) are still useful for the estimation of total duration through the following extrapolation: For occasional contributors: using equation $y = 0.0392x + 2$ from Table 2 and multiplying the estimated project duration by 3.

The comparison between the actual duration per release and the estimated duration in the model for occasional contributors with extrapolation is given in the electronic appendix to this article. The estimated duration for each release is obtained by replacing the 'x' variable in the equation $y = 0.0392x + 2$ (from Table 2) by the total number of commits for all occasional contributors corresponding to this release and multiplying the estimated duration by 3.

The estimation model with extrapolation has an \|RE\| ≤ 42% for 11 releases (4.2.5, 4.2.6, 4.2.7, 4.2.8, 4.3, 4.3.1, 4.3.3, 4.3.4, 4.4.1, 4.4.3 and 4.7) where 42% represents the MMRE of the estimation model for occasional contributors with extrapolation. The median \|RE\| = 38%. The estimation model for occasional contributors with extrapolation gives a better estimate than the model for full-time contributors per release presented in section *A* –from Table 2. It should be noted that our duration estimation model with only one independent variable lead to better estimates than those in Qi et al. [8] - from table 7, with many additional variables. These additional variables in Qi et al. [8] might provide either noise or distortions due to approximations and incorrect assumptions for these variables.

D. Estimation models per category of part-time contributors

The estimation model for part-time contributors was extrapolated by multiplying the estimated duration of the project by 3/2 using equation $y = 0.375x + 5$ from Table 2. The comparison between the actual duration per release and the estimated duration in the model for part-time contributors with extrapolation is given in the electronic appendix to this article. The estimated duration for each release is obtained by replacing the 'x' variable in the equation $y = 0.375x + 5$ (from Table 2), by the number of total commits for all part-time contributors corresponding to this release and multiplying the result by 3/2 (e.g. the required extrapolation for part-time contributors). For these models, the |RE| ≤ 44% for 11 releases (4.2.4, 4.2.5, 4.2.6, 4.3, 4.3.2, 4.3.3, 4.3.4, 4.4.1, 4.4.2, 4.4.3 and 4.5): 44% represents the MMRE of the estimation model for part-time contributors with extrapolation. The median |RE| is 31%. For only the releases with part-time contributors, the estimation per category of part-time contributors had an MRE ≤ 25% for 10 releases (4.2.4, 4.2.5, 4.2.6, 4.3, 4.3.2, 4.3.3, 4.3.4, 4.4.1, 4.4.2 and 4.4.3).

In summary, the single linear regression duration estimation model for occasional contributors (with the required extrapolation) gives the best estimation results compared to the models for full-time contributors and part-time contributors.

4.4. Estimation models with LOC

Twenty one (21) linear regression models were constructed using the total number of LOC added per contributor per release as the independent variable. Next, twenty one linear regression models were built with the total number of |LOC net| added per contributor per release as the independent variable.

Estimation models per release with LOC added per contributor per release are given in the electronic appendix. The performance criteria show that the duration estimation models with LOC added as independent variable do not produce good estimates compared to models with commits as the independent variable (from Table 1).

The independent variable parameter used to build the next set of estimation models was '|LOC net| added per contributor per release' and the dependent variable 'active days per contributor per release'. These estimation models are given in the electronic appendix. The performance criteria show that the duration estimation models with '|LOC net| added per contributor per release' as the independent variable do not produce good estimates compared to models with commits as the independent variable (from Table 1).

Three linear regression models were built next with the total number of |LOC net| added per category of contributors per release as the independent variable for full-time, part-time and occasional contributors – see electronic appendix. The MMRE for the full-time, part-time and occasional contributors are 0.25, 0.42 and 0.69, respectively while the MMRE for full-time contributors is better than for part-time and occasional contributors. The PRED for full-time contributors is excellent (PRED (20%) = 89%) compared to the two other models (e.g., the RE is within ±20% for 89% of the data points). We concluded that the estimation model with |LOC net| added per full-time contributors gave the best estimates compared to the models of the part-time and occasional contributors in terms of R², MMRE and PRED.

The actual and estimated duration per release by the model with |LOC net| added for full-time contributors is presented in the electronic appendix.

The estimation models for occasional was extrapolated using equation $y = 0.005x + 2$ and multiplying the estimated project duration by 3. The comparison between the actual duration per release and the estimated duration in the model for occasional contributors with extrapolation is given in the electronic appendix.

The estimation model for part-time contributors was extrapolated by multiplying the estimated duration of the project by 3/2 using equation $y = 0.0008x + 11$. The comparison between the actual duration per release and the estimated duration in the model for part-time contributors with extrapolation is given in the electronic appendix.

The estimation model for part-time contributors with extrapolation gives a better estimation than the model for full-time contributors and occasional contributors with extrapolations using |LOC net| as independent variable.

In conclusion, the estimation models for occasional contributors with extrapolation and for part-time contributors with extrapolation produced the best estimation results compared to the model for full-time contributors for each release on both commits and |LOC net| data.

4.5. Discussion of the estimation models based on commits and LOC

This section compares:

- The estimation models for each release:

    1) With 'commit',
    2) With 'LOC added', and
    3) With '|LOC net|'.

- The estimation models per category of contributors for each release with:

1) 'commit', and
2) '|LOC net|'.

Table 4 presents the performance criteria of the duration estimation models with independent variables 'commits', 'LOC added' and '|LOC net|' for each release.

In Table 4:

- The highlighted numbers indicate the most accurate results (e.g., good estimate), and
- The bold numbers indicate the least accurate results (e.g., worst estimate) for the same release.

Table 4 shows that:

    A. The duration estimation models for each release with 'commits' as the independent variable give good estimates in terms of the R², MMRE and PRED criteria for 9 releases (4.3.1, 4.4.1, 4.5, 4.6, 4.7, 4.8, 4.9, 5.0 and 5.1). These models also gave good estimates in terms of two criteria (R² and MMRE) for releases 4.2.6, 4.2.7, 4.2.8, 4.3.3, 4.3.4, 4.4.3 and 5.2.

    B. The duration estimation models for each release with |LOC net| as the independent variable give poor estimates in terms of most of the three criteria in 15 releases (4.2.7, 4.2.8, 4.3, 4.3.1, 4.3.3, 4.3.4, 4.4.1, 4.4.2, 4.4.3, 4.5, 4.6, 4.8, 4.9, 5.0 and 5.2).

Therefore, the duration estimation models for each release with 'commits' as the independent variable and the 'duration' as the dependent variable produced the best duration estimation results compared to the models using 'LOC added' and '|LOC net|'.

Table 4. WordPress project: Performance of estimation models based on commits LOC and |LOC net|

Data	Criteria	4.2.4	4.2.5	4.2.6	4.2.7	4.2.8	4.3	4.3.1	4.3.2	4.3.3	4.3.4	4.4.1	4.4.2	4.4.3	4.5	4.6	4.7	4.8	4.9	5.0	5.1	5.2
Commits	R²	0.37	0.53	0.79	0.71	0.61	0.57	0.87	0.77	0.85	0.6	0.82	0.34	0.9	0.71	0.92	0.81	0.92	0.90	0.95	0.62	0.84
	MMRE	2.10	0.90	0.36	0.28	0.62	0.55	0.35	0.55	0.45	0.71	0.58	0.82	0.44	0.56	0.25	0.63	0.52	0.37	0.33	0.57	0.11
	PRED (20%)	36	67	67	61	73	82	80	67	69	72	86	70	81	75	96	78	83	96	93	75	50
LOC added	R²	0.59	0.59	0.14	0.26	0.41	0.59	0.49	0.61	0.11	0.5	0.68	0.53	0.70	0.42	0.33	0.06	0.48	0.40	0.38	0.29	0.78
	MMRE	0.31	1.05	0.87	0.48	0.72	0.62	0.55	0.38	0.59	1.02	0.59	0.77	0.71	0.59	1.96	2.35	1.15	1.21	0.93	1.32	0.74
	PRED (20%)	67	71	75	65	67	82	67	62	80	86	70	74	84	70	79	67	73	60	70	74	72
|LOC net|	R²	0.47	0.58	0.004	0.10	0.40	0.28	0.28	0.57	0.01	0.35	0.30	0.16	0.54	0.22	0.17	0.24	0.39	0.20	0.12	0.33	0.12
	MMRE	0.66	0.94	0.84	0.73	0.74	0.47	0.59	0.54	0.64	1.18	1.21	1.02	0.71	0.67	2.56	2.18	1.50	1.30	1.56	0.78	1.64
	PRED (20%)	83	73	70	81	67	67	64	62	80	81	71	71	78	67	69	67	77	65	76	73	73

Table 5 summarizes the performance criteria resulting from the duration estimation models per category of contributors for each release with commits and |LOC net|.

Table 5. WordPress project: Performance of the estimation models based on category of contributors per release, commits and |LOC net|

Variables	Criteria	Category of contributors		
		Full-time Contributors	Part-time Contributors	Occasional Contributors
Commits	R²	0.36	0.33	0.60
	MMRE	0.24	0.30	0.49
	PRED (20%)	90	93	75
|LOC net|	R²	0.70	0.11	0.23
	MMRE	0.25	0.42	0.69
	PRED (20%)	89	79	72

In Table 5 the highlighted numbers indicate the most accurate estimates among the three categories of contributors by type of data. In Table 5, the estimation models:

- For full-time contributors give a good estimate in terms of the MMRE criterion with commits as the independent variable.

- For full-time contributors give the best estimate in terms of the R?, MMRE and PRED criteria with |LOC net| as the independent variable.

Therefore, the models with 'commits' as the independent variable produce the best estimate compared to models with '| LOC net |' as the independent variable.

5. Models of Duration from the Swift Project

To develop the estimation models, statistical analyses of Swift historical data were carried out on 9 releases from March 2016 to Jun 2018 extracted from the GitHub repository. Three measures of activity were used:

1) The number of changes to source code (commits) per contributor for each release.
2) The number of active days per contributor for each release, considered as days in which the contributor performs at least one commit to the code base.
3) The number of |LOC net| added per contributor for each release (e.g., |LOC net| = |LOC added - LOC deleted|).

Data for the number of contributors, their total commits and |LOC net| were extracted on June 27, 2019 from the GitHub repository (available online from https://github.com/apple/swift) using a program written in C including GitHub commands. These data are provided in the Electronic Appendix.

5.1. Models per release with 'commits'

The linear regression model was used to construct the duration estimation models by considering the number of commits per each contributor for each release as the independent variable and duration (active days) as the dependent variable.

A. Data preparation and descriptive statistics

The presence of outliers in each release was analyzed with the Grubbs test on both the independent variable 'commits per contributor per release' and dependent variable 'duration per contributor per release'.

Since these outliers may unduly influence the models, they were removed from the samples being analyzed. For the descriptive statistics of the releases, we took into account the number of contributors per release, the total number of commits per release, and the total duration per release in calendar days of work.

B. Models per release without outliers

Here, the independent variable was 'commit per contributor per release' and the dependent variable, 'duration per contributor per release (active days)'. The linear regression model is given by: *Duration = a × commit + b* where *a* represents the slope of the linear regression line and *b* represents the point at the origin (that is, when the independent variable is = 0). Figures 3 and 4 present the graph of the best results with the 'commits' and another one that is not very good.
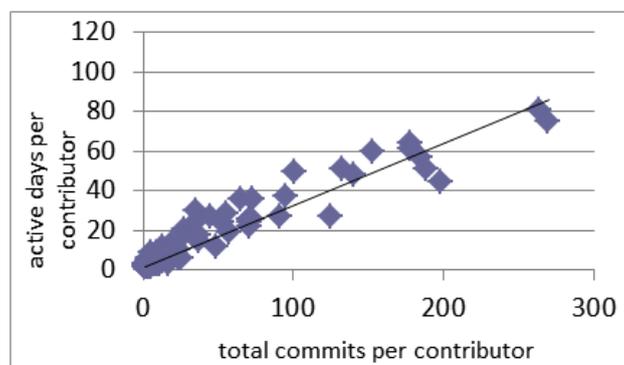

Fig.3. Swift project: Duration estimation models for release 2.2
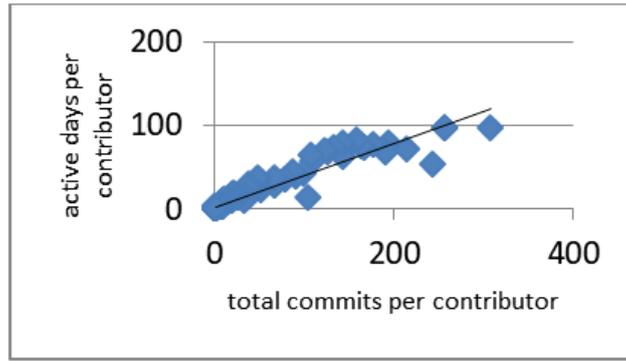
$$y = 0.3125x + 1 \hspace{4cm} (3)$$

Fig.4. Swift project: Duration estimation models for release 4.0

$$y = 0.3827x + 3 \tag{4}$$

Table 6 presents the equation of the model and corresponding R²; MMRE and PRED (0.20).

Table 6. Swift project: Estimation models per release

Release	Size range [in commits] per release	Total duration per release (in work days)	Equation of the model	R²	MMRE	PRED (20%)
2.2	[1, 270]	91	y = 0.3125x + 1	0.93	0.36	81
2.2.1	[1, 97]	31	y = 0.2984x + 2	0.88	0.61	82
3.0	[1, 136]	95	y = 0.4465x + 2	0.91	0.79	88
3.0.1	[1, 42]	33	y = 0.4473x + 1	0.85	0.36	79
3.1	[1, 186]	107	y = 0.4093x + 2	0.93	0.7	88
4.0	[1, 308]	126	y = 0.3827x + 3	0.91	0.98	86
4.0.3	[1, 75]	54	y = 0.4735x + 1	0.92	0.36	85
4.1	[1, 211]	83	y = 0.3848x + 3	0.92	0.97	82
4.2	[1, 135]	48	y = 0.3716x + 2	0.90	0.65	82
Mean MMRE = 64%						
Median MMRE = 65%						

From Table 6 we observe that the estimation models for the nine releases all have very high R² and PRED (20%) over 79%, while releases 2.2; 3.0.1 and 4.0.3 have by far a much better MMRE = 36%.

5.2. Models per release with '|LOC net|'

The next models are constructed using the number of |LOC net| per contributor per release as the independent variable and duration (active days) as the dependent variable.

A. Data preparation and descriptive statistics

The presence of outliers in each release was analyzed with the Grubbs test on both the independent variable '|LOC net|' per contributor and dependent variable 'duration'. Since these outliers may unduly influence the models, they were removed from the samples being analyzed. The descriptive statistics of the releases are presented in the electronic appendix.

B. Models per release (without outliers)

The independent variable is '|LOC net| per contributor per release' and the dependent variable 'duration per contributor per release'. Table 7 gives the equation of the model and the corresponding R², MMRE and PRED (0.20).

Table 7. Swift project: Estimation models per release - N = 9

Release	Size range [in |LOC net|] per release	Total duration per release (in work days)	Equation of the model	R²	MMRE	PRED (20%)
2.2	[1, 178]	91	y = 0.0406x + 2	0.15	0.93	75
2.2.1	[1, 1213]	31	y = 0.0107x + 5	0.24	1.62	71
3.0	[1, 1491]	95	y = 0.0207x + 3	0.47	1.43	71
3.0.1	[1, 476]	33	y = 0.0248x + 3	0.31	1.07	78
3.1	[1, 2201]	107	y = 0.0163x + 4	0.43	1.59	84
4.0	[1, 9349]	126	y = 0.0092x + 7	0.57	2.64	79
4.0.3	[1, 2712]	54	y = 0.0121x + 4	0.41	1.75	76
4.1	[1, 1582]	83	y = 0.021x + 3	0.38	1.65	82
4.2	[1, 1184]	48	y = 0.0184x + 4	0.33	1.32	84
Mean MMRE = 156%						
Median MMRE = 159%						

5.3. Comparison between models with 'commits' and '|LOC net|'

Table 8 presents a comparison of the performance criteria of the duration estimation models using 'commits' or '|LOC net|' for each release.

Table 8. Swift project: Comparison between models using commits or |LOC net|

Data	Criteria	Releases								
		2.2	2.2.1	3.0	3.0.1	3.1	4.0	4.0.3	4.1	4.2
Commits	R²	0.93	0.88	0.91	0.85	0.93	0.91	0.92	0.92	0.90
	MMRE	0.36	0.61	0.79	0.36	0.7	0.98	0.36	0.97	0.65
	PRED (20%)	81	82	88	79	88	86	85	82	82
|LOC net|	R²	0.15	0.24	0.47	0.31	0.43	0.57	0.41	0.38	0.33
	MMRE	0.93	1.62	1.43	1.07	1.59	2.64	1.75	1.65	1.32
	PRED (20%)	75	71	71	78	84	79	76	82	84

In summary, the duration estimation models built from the Swift project data built by release with 'commits' as the independent variable produce the best estimates in terms of R2, MMRE and PRED compared to models using '|LOC net|' as the independent variable.

Even though the number of data points of WordPress project (21 releases) is greater than for the Swift project (9 releases), we can also observe in Table 4 and Table 8 that duration estimation models built from WordPress project data gave the best estimates in terms of R^2, MMRE et PRED compared to models built from Swift project data.

6. Conclusion

6.1. Summary of findings

While a number of issues in software projects estimation have been investigated over the years, most have focused on the construction of software effort and duration estimation models for conventional software projects with commercial licenses, not on Open Source projects. This paper has addressed specifically the estimation of duration of OSS projects and a number of duration estimation models have been constructed using data from the WordPress and Swift OSS projects. Statistical regression techniques were used to construct models using two types of data as variables: the number of commits and the number of LOC per contributors and categories of contributors. We proposed an improved approach by 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.

First, duration estimation models for each release were constructed using the number of commits per contributor. Next, duration estimation models per category of contributor for each release were developed with total commits per

category of contributor per release.

Next, duration estimation models were built with the following independent variables:

- LOC added per contributor per release,
- total |LOC net| per contributor per release, and
- total |LOC net| per category of contributors per release.

For the WordPress OSS project, the duration estimation models with 'commits' gave better estimates than models with 'LOC added' and '|LOC net|'. More specifically in this WordPress project:

- The estimation models for full-time contributors gave better estimates in terms of MMRE and PRED with commits as the independent variable than part-time and occasional contributors.
- The estimation models for full-time contributors gave the best estimates in terms of R^2, MMRE and PRED with |LOC net| as the independent variable.
- While it appears that models with 'commits' as the independent variable produce better estimates than models with '| LOC net |' as the independent variable, 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 both for the commit data and the |LOC net| data. The extrapolation models produce estimates close to the actual duration of the release (MRE close to 0).

The statistical analysis of historical data of the Swift project validated next the estimation results obtained for the WordPress project: indeed, for both OSS projects, the duration models built with commits as the independent variable gave better estimates than the models with LOC added and |LOC net|.

The research objective of this paper is to build duration estimation models for Open Source Software projects. Information about when functionalities will be available is of a major interest to organizations that have adopted OSS as their business strategy: our duration estimation models estimate how long it will take to deliver the expected functionalities. One of the most interesting findings of our study is that, commits variable could be used for the estimation of project duration in Open Source Software projects. This research is also of interest to researchers and the proposed duration estimation models are intended to serve as a decision support tool for any organization developing software systems based on Open Source. All our 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.

6.2. Study limitations

All empirical studies, such as those reported here, are subject to certain threats to validity.

The recommended size of dataset sample required for meaningful statistical regression results is between 20 to 30 data points. The sample size used to construct the estimation models per category of contributors, with commits as the independent variable, was relatively small (10, 15 and 20 data points, respectively). This does not mean, however, that these models are meaningless and not useful to organizations. To the contrary for organization using their own data, even such small samples can provide them with an objective and quantitative view of the performance of their development processes from which they can derive insights into productivity performance about what can be reasonably expected of subsequent projects. In this paper, we used a program written in C and GitHub commands to extract data from the repositories. Other techniques related to extraction and manipulation of data can be also used. The dependent variable is active days (workday's duration unit). The classification of contributors was made for each release into three groups: full-time, part-time and occasional. We proposed criteria for an objective classification to categorize contributors as full-time, part-time or occasional as described in the research approach's section of our paper. However, some assumptions have been made in the design of our 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. For example, a contributor can perform one commit in two minutes but this time is considered as one active day.

6.3. Future work

Further studies may be done using two independent variables (occasional contributors and part-time contributors) using our study as a baseline for comparison purposes. In this paper, we used an approach for classifying contributors. More studies on the classification of contributors should be done. We also 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 independent variable, 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 and code quality, but all these variables together will only help to predict the part of the duration not explained yet by the commits. We also plan to build estimation models per category of contributors with Swift project data.

Electronic Appendix

Acknowledgment

References

[1]   A. Ihara, A. Monden, K. Matsumoto, "Industry Questions About Open Source Software in Business: Research Directions and Potential Answers", In *Proc. the 6th Int. Workshop on Empirical Software Engineering in Practice (IWESEP)*, November 2014, pp.55-59.

[2]   A. Boulanger, "Open-source versus proprietary software: Is one more reliable and secure than the other?", *IBM Systems Journal*, 44(2): 239-248, 2005.

[3]   B. Fitzgerald, "A Critical Look at Open Source", *IEEE Computer*, 37(7): 92-94, 2004.

[4]   J. Asundi, "The Need for Effort Estimation Models for Open Source Software Projects" In *Proc. the 5th Workshop on Open Source Software Engineering*, July 2005, pp.1-3.

[5]   https://github.com/ten, [accessed on April 18, 2018].

[6]   C. Abts, B. Clark, S. Devnani-Chulani, "COCOMO II Model Definition Manual", University of Southern California, 1997.

[7]   G. Mathew, T. Menzies, J. Hihn, "Impacts of bad ESP (early size predictions) on software effort estimation", arXiv preprint arXiv:1612.03240, 2016.

[8]   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.

[9]   E. Shihab, Y. Kamei, B. Adams, et al., "Is lines of code a good measure of effort in effort-aware models?", *Information and Software Technology*, 55(11): 1981-1993, 2013.

[10]   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", In *Proc. the 11th Working Conference on Mining Software Repositories*, May 31 –June 01 2014, pp.222-231.

[11]   J. J. Amor, G. Robles, J. M. González-Barahona, "Effort estimation by characterizing developer activity", In *Proc. the International Workshop on Economics Driven Software Engineering Research*, May 2006, pp.3-6.

[12]   E. Kalliamvakou, G. Gousios, D. Spinellis, N. Pouloudi, "Measuring developer contribution from software repository data", In *Proc. the 4th Mediterranean Conference on Information Systems*, September 2009.

[13]   A. Capiluppi, D. Izquierdo-Cortazar, "Effort estimation of FLOSS projects: a study of the Linux kernel", *Journal of Empirical Software Engineering*, 18(1): pp. 60-88, 2013.

[14]   A. Mockus, R. Fielding, J. Herbsleb, "Two Case Studies of Open Source Software Development: Apache and Mozilla", *ACM Transactions on Software Engineering and Methodology - TOSEM*, 11(3), pp. 309-346, 2002.

[15]   S. Koch, "Profiling an Open Source ecology and its programmers", *Electronic Markets*, 14 (2), pp.416-429, 2004.

[16]   S. Koch, "Effort Modeling and Programmer Participation in Open Source Software Projects", *Information Economics and Policy*, 20(4), pp.345-355, 2008.

[17]   D. K. Moulla, Kolyang, "COCOMO model for software based on Open Source: Application to the adaptation of TRIADE to the university system", *International Journal on Computer Science and Engineering (IJCSE)*, Vol. 5(6), pp. 522-527, 2013.

[18]   D. K. Moulla, I. Damakoa, Kolyang, "Application of Function Points to Software Based on Open Source: A Case Study", In *Proc. the Joint Conference of the International Workshop on Software Measurement and the International Conference on Software Process and Product Measurement (IWSM-MENSURA)*, October 2014, pp.191-195.

[19]   B. W. Boehm, "Software Engineering Economics", Prentice-Hall, New-Jersey, 1981.

[20]   A. J. Albrecht, "Measuring application development productivity", In *Proc. the IBM Application Development Symposium*, October, 1979, 83.

[21]   A. J. Albrecht, J. E. Gaffney, "Software function, source lines of code, and development effort pre-diction: a software science validation", *IEEE Transactions on Software Engineering*, 9 (6), pp. 639–648, 1983.

[22]   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.

[23]   L. Yu, "Indirectly predicting the maintenance effort of Open Source Software", *Journal of Software Maintenance and Evolution: Research and Practice*, 18(5):311-332, 2006.

[24]   P. Anbalagan, M. Vouk, "On predicting the time taken to correct bug reports in Open Source projects", In *Proc. the International Conference on Software Maintenance (ICSM)*, 2009, pp. 523-526.

[25]   S. Chatterjee and A. S. Hadi, "Regression analysis by example", John Wiley & Sons, 2015.

[26]   M. Jorgensen, "Regression models of software development effort estimation accuracy and bias", *Empirical Software Engineering*, vol. 9, pp. 297–314, 2004.

[27]   D. C. Montgomery, E. A. Peck, G. G. Vining, "Introduction to Linear Regression Analysis", 5th ed. Wiley, Hoboken, NJ. 2012.

[28]   R. Silhavy, P. Silhavy, and Z. Prokopova, "Analysis and selection of a regression model for the Use Case Points method using

a stepwise approach", *Journal of Systems and Software*, vol. 125, pp.1-14, 2017.

[29] S. D. Conte, D. E. Dunsmore, V. Y. Shen, "Software Engineering Metrics and Models", Redwood City, California: Benjamin-Cummings Publishing Company, Inc. ISBN:0-8053-2162-4, 1986.

[30] Novatis, Quels sont les CMS les plus utilisés en 2016 ? https://www.novatis.tn/quels-sont-les-cms-les-plus-utilises-en-2016/, 08/2016. [accessed on May 31, 2017].

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

Authors' Profiles

Dr. Donatien K. Moulla is a Lecturer in computer science at the Faculty of Mines and Petroleum Industries, University of Maroua, Cameroon. He received his Ph.D. degree in computer science from the University of Ngaoundéré. He earned his Bachelor's and Master's degrees from the department of Mathematics and Computer Science in the same university. He has more than 8 years of teaching and research experience in information systems development and software engineering. His research interests include software estimation, software measurement, software quality and Open Source Software projects.

Alain Abran, PhD, is a Professor at 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.