On the Relationship Between Story Point and Development Effort in Agile Open-Source Software

Vali Tawosi, Rebecca Moussa, Federica Sarro
{vali.tawosi,rebecca.moussa.18,f.sarro}@ucl.ac.uk
University College London, London, UK

ABSTRACT

Background: Previous work has provided some initial evidence that Story Point (SP) estimated by human-experts may not accurately reflect the effort needed to realise Agile software projects. **Aims:** In this paper, we aim to shed further light on the relationship between SP and Agile software development effort to understand the extent to which human-estimated SP is a good indicator of user story development effort expressed in terms of time needed to realise it. **Method:** To this end, we carry out a thorough empirical study involving a total of 37,440 unique user stories from 37 different open-source projects publicly available in the TAWOS dataset. For these user stories, we investigate the correlation between the issue development time (or its approximation when the actual time is not available) and the SP estimated by human-expert by using three widely-used correlation statistics (i.e., Pearson, Kendall and Spearman). Furthermore, we investigate SP estimations made by the human-experts in order to assess the extent to which they are consistent in their estimations throughout the project, i.e., we assess whether the development time of the issues is proportionate to the SP assigned to them. **Results:** The average results across the three correlation measures reveal that the correlation between the human-expert estimated SP and the approximated development time is strong for only 7% of the projects investigated, and medium (58%) or low (35%) for the remaining ones. Similar results are obtained when the actual development time is considered. Our empirical study also reveals that the estimation made is often not consistent throughout the project and the human estimator tends to misestimate in 78% of the cases. **Conclusions:** Our empirical results suggest that SP might not be an accurate indicator of open-source Agile software development effort expressed in terms of development time. The impact of its use as an indicator of effort should be explored in future work, for example as a cost-driver in automated effort estimation models or as the prediction target.

CCS CONCEPTS

• **Software and its engineering**;

KEYWORDS

Software effort estimation; Story Point; Agile software.

ESEM'22, September 19–23, 2022, Helsinki, Finland
© 2022 Association for Computing Machinery.
ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/XXXXXXX.XXXXXXX

ACM Reference Format:
Vali Tawosi, Rebecca Moussa, Federica Sarro. 2022. On the Relationship Between Story Point and Development Effort in Agile Open-Source Software. In *Proceedings of 16th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM'22).* ACM, New York, NY, USA, 12 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

Software Effort Estimation (SEE) is a crucial activity for managing, planning, and monitoring software projects [53]. Without an accurate estimation of the effort required to develop software, budget and schedule overrun seem inevitable [21, 44]. SEE research has mainly focused on estimating the effort required to develop a whole project (i.e., project-level estimation). To this end, Functional Size Measures (FSM), such as Function Point (FP) [6] or COSMIC Function Point (CFP) [46], have been usually used as a cost driver to estimate traditional software development effort [4, 14, 16, 17, 53].

The advent of Agile Software Development (ASD) methodologies [19] has shifted the focus towards estimating the effort of developing smaller unit of software, like a new feature or change. In these cases, FSM methods are not easy to use [38] and another measure, namely Story Point (SP), has become popular in the context of ASD [56]. SP is a relative unit that represents an intuitive mixture of complexity and required effort of a user story (a.k.a. issue) [8, 53]. [1]

However, previous studies have shown that the accuracy of the SP estimate is sensitive to the practitioners' expertise, and thus, prone to bias. According to Usman et al. [55], who surveyed 60 engineers experienced in Agile Effort Estimation, the estimates of around half of the Agile teams were inaccurate by a factor of 25% or more. Using inaccurate SP could result in iteration mismanagement and wrong prioritization of tasks, which in turn can lead to customer dissatisfaction or even project failure. Moreover, since human-expert estimated SP has been used as a cost driver to train automated estimation models [24, 33, 35, 39, 54, 58] or as a prediction target [2, 10, 20, 29, 34, 40, 43, 49, 50], researchers and practitioners need to be aware if they are using inaccurate SP as this might impact the accuracy of these models.

Previous case studies have provided discordant results on whether SP can accurately capture software size and effort [10, 26, 37, 38], and to date there is not enough empirical evidence on this matter. Our work aims to fill this gap by carrying out a thorough large-scale empirical study investigating the extent to which using Story Point reflects the effort needed to develop a user story (i.e., issue development time). To this end, we analyse 37,440 user stories coming from 37 Agile software projects tracked with Jira [1], which are available in the TAWOS dataset [48]. To the best of our knowledge this is

[1]A user story is a user-valued functionality which is specified in the form of one or two sentences in the everyday language of the user.

the largest empirical study to date to investigate the relationship between SP and effort in Agile open-source projects. In particular, we aim to answer the question *What is the relationship between the SP estimated for a given issue and its actual development time?*. Since developers do not always record the actual time they spent on the development of an issue [48] in the issue tracking system, we compare three different proxies for the development time computed using the issue changelog[2] in order to answer the question: *To what extent can we approximate the actual development time as reported by the developers?*. Furthermore, since one would expect issue development time to be proportionate to the story point assigned to a certain issue, we also aim to answer *How consistent is the assignment of SP throughout a project?*.

The results of our empirical study show that among the three proxies, there is one which more closely reflects the development time as recorded by the developers, namely the InProgress development time. Moreover, we found that the correlation between this issue development time and human-expert estimated SP is medium or low for 93% of the projects we investigated. These results are in line with those we obtained using the recorded development time rather than the proxy and they highlight that SP is not an accurate indicator of the software development effort. Moreover we found that the human-expert estimation is not consistent throughout the projects. Although SP can remain useful for agile teams to organize and plan their iterations, these results raise awareness that the inaccuracy observed in the SP might be carried out into those automated effort estimation models that use SP as a cost driver to predict issue development time. Moreover, recent studies have proposed the use of machine learning approaches to predict SP for issues based on historical human-estimated SP. This means that such approaches learn to imitate human-expert estimations at best which in itself might be misleading of the actual effort needed to realise an issue. Further work is needed to understand the extent to which the use of inaccurate SP impact automated effort estimation models, and whether the use of development time (actual or proxies for it) can provide the engineers and managers with more reliable and accurate models.

The rest of this paper is organised as follows. Section 2 provides some background for those readers who are not familiar with software size and effort measures, and issue tracking systems. Sections 3 and 4 present the design and results of our empirical study, respectively. Section 5 discusses previous work most related to ours. Final remarks and future work are discussed in Section 6.

2 BACKGROUND

In this section we briefly introduce the most common software functional size and effort measures proposed in the literature. We also give some background on Jira, the issue tracking system used by the projects analysed in this study, and describe three proxy measures for issue development time [48].

2.1 Software Size Measures

Albrecht was the first to introduce a disciplined method for measuring software product size, called Function Point Analysis (FPA), based on the functionality the software product is built to deliver to the customer [5]. Soon after, he showed that there is a strong correlation between Function Points and the final effort of a software [6]. Although FPA was designed to measure software from the domain of business applications [47], it is still widely applied in the software production industry [15].

COSMIC[3] Function Point (CFP) method belongs to the second generation of software functional size methods [46]. CFP also takes non-functional requirements into consideration, and is suitable for a broader range of application domains including, but not limited to, business applications, web applications, mobile applications, real-time software, and service-oriented software [4, 14, 18, 47].

In the context of Agile software development, practitioners have introduced and used Story Point (SP) as an Agile specific software size measurement unit [12]. Unlike FPA and CFP, SP do not follow a method of measurement, therefore, developers use them as a relative measure to keep the relative difference of stories in size by assigning a point value to each user story. One common approach to determine the story point value of a user story is to select one of the smallest stories in the Backlog[4] and assign it with one story point. Then the more complex and larger user stories get more points considering their size [12]. So any user story that is assigned two SP is twice as large as a user story that is assigned one SP. SP estimations need to be consistent throughout the project.

2.2 Jira Workflow and Issue Development Time

Jira [1] is a widely-used issue tracking system that supports Agile development [31]. Using Jira, the development teams can record their estimated story point and the time taken for the development of the issue.

Although Jira Software has provided teams with specific fields to record the actual effort (i.e., time) spent on an issue, usually the developers do not use this feature to log their work. Thus, identifying the actual time spent to develop an issue might be challenging. Nonetheless, previous work [10, 48] derived an approximation of the actual development time from the transitions recorded in the change-log of the issues in the Jira repository.

Jira Workflow: In Jira projects, issues transition through stages of work —from creation to completion— following a path. This path is called workflow. Figure 1 shows a generic Jira workflow that could be used to track issue transitions and calculate approximate time spent on issue development.

The life cycle of an issue starts at the time of its creation (the grey circle in the workflow). When the issue is considered to be developed, its status changes into a *To Do* state (1). This is when the issue gets assigned to a developer. When the developer starts working on an issue, he/she changes its status to *In-Progress* (2). The developer is given the option to stop working (3) and restart it again (2) at any time.

When the development is finished, the developer changes the issue's status to *Done* (4), and Jira automatically populates the Resolution field (5). Also, the developer can set the Resolution field any time during the life cycle of the issue. The Resolution field can be populated with one of the several labels predefined

[2]The history of changes in the issue's attributes.

[3]Common Software Measurement International Consortium

[4]Backlog is a breakdown of work containing an ordered list of user stories that an Agile team maintains for a project.

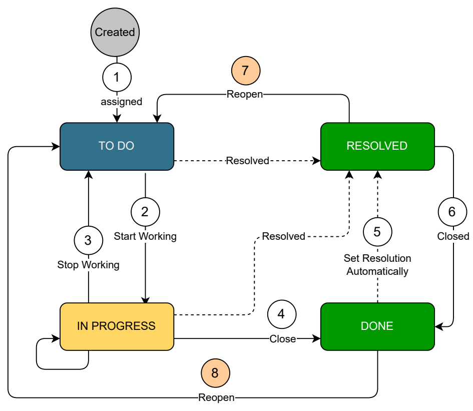


Figure 1: A generic Jira workflow.

in the workflow (usually but not necessarily with statuses defined within the *Done* category), e.g., Fixed, Completed, Closed, Delivered, Invalid, Duplicate, Won't Fix, Rejected, and Cancelled, depending on the project and issue type.

Jira recognises an issue as *Resolved* if the Resolution field is populated. By default *Resolved* means that the issue is in a closed state and no more work is needed to be done. But in many workflows *Resolved* is not an ultimate state. For instance, in a custom workflow, once the issue is resolved there might be an inspection which decides if the solution provided is sufficient and/or correct —the review process— before the issue can be closed (6). If the solution is not accepted, the issue will then be reopened and it would need to be addressed again (7). Ultimately, an issue might be reopened after it has been closed (8), although this is rare.

Issue Development Time: The workflow is typically specific to the work processes within an organization/team. Indeed, Jira provides organizations with the ability to create customised workflows and statuses for each project and issue type. This makes it difficult to create a general method to calculate the time by observing issue transitions. However, a custom status defined in a custom workflow has to belong to one of these three categories: *To Do*, *In-Progress*, and *Done*. Jira mandates the use of these categories and employs them internally to identify the column under which each issue should be listed in the software task board. Therefore, one can base the time calculation upon these three categories. Specifically, using the status categories, one can identify the transitions of the issues between the time they were set to be in progress, stopped progress, or accomplished.

Based on the above workflow, we have defined the following three proxies for issue development time.

In-Progress Time is defined as *the duration in which an issue has been in the "In-Progress" status.* In most projects the "*In-Progress*" status is used by developers to mark the time that they spend on the implementation of an issue. Hence, In-Progress Time might not include any time spent on testing, reviewing or discussion.[5]

Effort Time is defined as *the duration in which an issue has been in any of the statuses categorised as* In-Progress. This definition can be interpreted as a more realistic proxy for the effort since it includes time spent for implementation, testing, reviewing, discussions, etc., as the Effort Time considers all the time that an issue spends under any status from the *In-Progress* category.[6]

Resolution Time is defined as *the duration required for an issue to be resolved.* As we can see in Figure 1, an issue status can be set to *Resolved* at any point in its life cycle. This definition aims at capturing the amount of time it takes for an issue to be resolved. To this end, we consider the duration between the time an issue was created until it is *Resolved* [48]. This is the definition used in the literature to measure the time to fix an issue [23, 28, 41]. This proxy slightly differs from the one used in the work of Choetkiertikul et al. [10], as it also takes into account the time between the creation of the issue and the first time it was set to an In-Progress status. Whereas, the proxy used by Choetkiertikul et al. considers the duration between the time an issue was first set to an In-Progress status and the time that it was resolved.

3 EMPIRICAL STUDY DESIGN

In this section, we describe the research questions posed in our study and the dataset, methods and statistical tests used to answer these questions.

3.1 Research Questions

Story point, as a measure of effort, is expected to have a positive correlation with the actual time needed to realise a software.

In this study we aim at investigating the correlation between estimated story point and the actual effort. As, the actual effort is rarely recorded in an issue report, we analysed three proxies for the development time based on the Jira workflow as described in Section 2.2. Therefore, our first research question assesses which of these proxies are a good approximation of the actual effort.

RQ1. Approximating Issue Development Time: *To what extent can proxies be used to approximate the development time logged by the developers?*

Once we assess whether these proxies provide a satisfactory approximation, we move to investigate the correlation between story point and the actual effort (i.e., each of the three proxies proposed):

RQ2. Correlation: *What is the relationship between an issue's story point and its development time?*

To answer this question we use three widely known correlation statistics to verify the relationship between the SP and each of the three proxies we used for approximating the development time (see Section 2.2).

SP is a relative measure by definition, and the relativeness refers to the amount of the work that one story point represents. It can differ from project to project and from team to team. This rate (i.e., one story point ratio with respect to the amount of the effort in person-hour) might be affected by aspects such as the experience

[5]Note that there is always a status named *In-Progress* within the *In-Progress* category, teams can add other statuses into this category based on their issue ecosystem. For instance, in a project which has *In-Progress*, *Test*, and *Review* statuses defined in the *In-Progress* category, the issue may transition from one status to another until it passes all the required stages before it is closed (i.e., set to *Done*). This is showed by a recursive arrow for the *In-Progress* status in Figure 1.

[6]To identify the category of each status in each project, we queried the metadata of each project's repository by using the REST API provided by Jira [48].

of the team, the programming language and the technology used for development. This rate is used to compute the productivity of the team and to make the story point scale specific to each team. However, within a project, the estimation team should remain consistent throughout the project with respect to the unit of work that a story point represents. In other words, the amount of work considered for a story point should be kept the same until the end of the project and all the issues should be measured with that same unit. Nevertheless, keeping this rate consistent is challenging for any team. This phenomenon justifies the rationale for our third, and last, research question, which emphasises on the variance of the time for each SP:

RQ3. SP Consistency: *How consistent is the assignment of story point throughout a project?*

To answer this question, we rely on a visual representation to identify any deviation between the actual data and an ideal trajectory of consistency in SP estimation, derived from the data itself as further explained in Section 3.2.

3.2 Methodology

To answer RQ1, we compare the *Timespent* value with the time as measured by each of the proxies for each issue, and compute the absolute error one would commit had a proxy been used rather than the actual value. Specifically, to measure the resemblance of the three proxy measures to *Timespent*, we compute the Sum of Absolute Errors (SAE) between each of the proxies and the *Timespent* for the issues contained in each project. SAE is computed as follows: $SAE = \sum_{i=1}^{n} |P_i - TS_i|$, where n is the number of issues in the project with reported *Timespent* values, TS_i is the *Timespent* value for issue i, and P_i is each of the values of the development time proxies for issue i, obtained from the TAWOS dataset [48]. The proxy with the minimum SAE is the most representative of the *Timespent*. Then we apply statistical significance tests on the distribution of each of the proxies against *Timespent* to verify whether the difference between them is statistically significant. Specifically, we used the Wilcoxon Rank-Sum test (a.k.a. Mann–Whitney U test) [30] to check for statistical difference. The confidence limit is initially set to $\alpha = 0.05$ and is corrected for multiple hypotheses using the standard Bonferroni correction (α/K, where K is the number of hypotheses). To answer RQ1, we tested the following null hypothesis: H_0: *The distribution of the Timespent is not different from that of the proxy* P_i. For those cases where the null hypothesis is rejected, the following alternative hypothesis is accepted: H_1: *The distribution of the Timespent is different from that of the proxy* P_i. To measure the effect size of the difference, we use Vargha Delaney's $\hat{A}_{12}$ measure, which is a standardised non-parametric effect size measurement, to assess how meaningful the difference between the two distributions is [7]. According to the Vargha Delaney's effect size, if the two distributions are very similar $\hat{A}_{12} = 0.5$. Respectively, an $\hat{A}_{12}$ closer to 1 means that the two distributions are not similar. The effect size is considered small for $0.6 \le \hat{A}_{12} < 0.7$, medium for $0.7 \le \hat{A}_{12} \le 0.8$, and large for $\hat{A}_{12} \ge 0.8$, although these thresholds are not definitive [52].

To answer RQ2, we apply three correlation statistics to our data: the Pearson r correlation coefficient [32], the Spearman's ρ rank correlation [45], and the Kendall's τ rank correlation [27]. The Pearson correlation test measures the linear correlation between two variables, while the Spearman's and Kendall's correlation tests are statistics used to measure the ordinal association between two samples and assess how well the relationship between two variables can be described using a monotonic function [9]. Unlike the Pearson's r which considers the value of the data points, the Spearman's ρ and Kendall's τ work with ranks of data points which makes them less sensitive to strong outliers that lie in the tails of both samples [13]. All three correlation statistics range from +1 to −1, where +1 indicates a perfect correlation and −1 indicates a perfect inverse correlation. A non-correlation is indicated by a 0. Although both Spearman's ρ and Kendall's τ measure rank correlation, they cannot be compared directly with one another since they have different scales. Gilpin [22] describes the ratio of ρ to τ to be almost 1.5 for most of the range. The two get close to each other as their magnitude increases towards the limits (i.e., both approaching +1 or −1) and when they both approach zero. We use the Cohen's standard [11, 25] for interpreting the correlation coefficients to determine the strength of the relationship. Based on this, correlation coefficients between 0.10 and 0.29 represent a small association, coefficients between 0.30 and 0.49 represent a medium association, and coefficients equal to or greater than 0.50 represent a large association. To perform the correlation we used the `cor.test` method available in `R` version 4.0.1. Both the Spearman's and Kendall's correlation statistic implementations used in this study can handle ties in the data points.

To answer RQ3, we group all the issues that have been assigned a same story point of value <*X*> (i.e., story point <*class X*>) in a same class. Then we analyse the boxplots of the time spent on each issue for the distinct classes to understand if the distribution of time in story point classes is a normal one, which would indicate a normal distribution of the error for story point estimation in each class. To investigate consistency, we observe the median point in the distribution of development time per SP class. We use the median point as it is not affected by the extreme values. For a project with inconsistent SP estimations, the median development time will be affected (misplaced from ideal trajectory) due to many miss-classifications of smaller or bigger tasks in a specific SP class; or from another point of view, issues estimated to have the same SP do not agree on the same (or similar) development time. In an ideal scenario, the median of the distribution of the development time in story point classes should have a linear relationship with the value of the story point. For instance, the median of the development time for issues assigned with story point five should be five times larger than the median of the development time for issues assigned with story point one. Should this linear relationship hold for all of the issues in story point classes, we can assert that story point estimations are consistent throughout a project. To test this we show the trajectory of this linear relation to visualise the degree of the consistency.

3.3 Dataset

We sample data from the TAWOS dataset version 1.0 [48], which is a collection of diverse open-source Agilesoftware projects. These projects have been mined from several different repositories maintained using Jira Issue Tracking System (ITS) [1]. The TAWOS dataset is publicly available in the form of a relational database and contains more than half a million issues from 44 projects. We used SQL queries to sample issues from this database[7]. Below we describe in detail, how we sample the set of projects investigated in this study.

To answer our first research question (i.e., RQ1), we analyse those projects from the TAWOS dataset that have recorded the actual development time in the *Timesent* field of Jira. Hence, we selected all the issues that are resolved (i.e., we filter out all those that are not addressed), and have the *Timespent* field populated. Then, we removed all issues having a *Timespent* value lower than two minutes as done in previous work [57], to reduce noise in the data. After applying such a filtering, we retained all those projects with at least 100 issues each. This resulted in a sample of 9,806 issues from 15 projects. Descriptive statistics of this set, which is used to answer RQ1, are provided in Table 1a.

As most of the issues recorded either one of the *Timespent* or SP, in order to answer RQ2 and RQ3 we also sampled another set of issues.[8] To this end we filtered out from the TAWOS data all those issues that are not addressed, and those that have been assigned with SP less than 1 and greater than 100, as done in previous work [10], to reduce the presence of data that is not relevant to the purpose of our empirical study. Moreover, we noticed that a considerable number of the issues in some of the projects have a proxy time equal to zero. After a careful manual inspection, we found that there are issues which never transitioned to an *In-Progress* status and thus they had been closed immediately after being created or opened. These cases may correspond to issues where developers had already worked on the issue before tracking the corresponding record in Jira created for the mere purpose of recording issues. In order to reduce the bias introduced by these cases, and in accordance with the filter used for RQ1, we removed all the issues with *In-Progress time* less than two minutes, which corresponded to a total of 34.47% of the issues sampled from the TAWOS dataset. Furthermore, we filtered out issues with outlying values of *In-Progress time* to minimise the effect of extreme values in our results.[9] We therefore retained projects with at least 100 issues after filtering out unwanted ones, which left us with 58.33% of the initially sampled data corresponding to a total of $28,608$ issues from 32 projects (equal to 44.11% of all the issues with recorded SP in 32 projects under investigation). To identify the outliers, we used the Interquartile Range (IQR). The IQR, which is equal to the difference between the 75^{th} and 25^{th} percentiles of the distribution of the data points, is multiplied by 1.5 and the resulting value is subtracted from and added to the first and third quartiles, respectively, to get the lower and upper fences (a.k.a. Tukey fences). The data points falling outside the lower and upper fences are considered outliers and, hence, removed from the dataset. The resulting data has been used to answer RQ2 and RQ3. Descriptive statistics of this sampled dataset can be found in Table 1b. Note that a total of 621 issues from the open-source data sampled for RQ1 are also in the sample used for RQ2 and RQ3. Therefore, the total number of unique user stories sampled from the TAWOS dataset is equal to 37,440 extracted from 37 different open-source projects.

3.4 Threats to Validity

To mitigate construct validity threats we use data from real-world projects, which have been carefully curated and used in previous work [48–50]. The story point values are predicted by human-experts and recorded in the Jira issue repository. However, the values we use as the actual time are extracted from the issue change-log, based on the issue transitions recorded in the repository throughout the development process. We are aware that these time-values might not accurately represent the actual effort spent on developing an issue, and we mitigated this threat by considering three different approximations (i.e., In-Progress time, Effort time, and Resolution time), each capturing different aspects of the actual effort. We also used the actual development time recorded for issues to investigate the extent to which these proxies resemble the actual development time. Data points which are likely to be noisy, such as issues that have not been fully resolved or issues with less than 2 minutes of recorded development time, were filtered out from the dataset before any analysis, as described in Section 3.3.

Using proxies of the development time instead of the actual time might also be a threat to the internal validity of this study. In other words, the low correlation between SP and the proxies might be because of an unrepresentative proxy and not the expert misestimation. We mitigate this threat by conducting the same correlation analysis with the actual development time recorded by the developers where this was available.

With regard to the conclusion validity, we used three well-known correlation statistics and reported the corresponding p-value. To investigate the similarity of the proxies to the actual development time, we used the Sum of Absolute Errors and examined the statistical difference between the absolute error distributions by applying the Wilcoxon Rank-Sum test with all required assumptions checked, following best practice for effort estimation studies [42].

To mitigate external validity threats we used a large set of 37 projects which differ in size, application domain, programming language, and development team. Although we used such a diverse dataset, all the projects are open-source and the results might not be generalizable to other contexts.

4 EMPIRICAL STUDY RESULTS

This section presents the results we have obtained in answering the research questions described in Section 3.1.

4.1 RQ1. Approximating Issue Development Time

Table 2 shows the SAE values computed for the three proxies with respect to the *Timespent* value. We can observe that, among the three proxies under study, In-Progress Time has the smallest error for all projects. However, while this indicates that In-Progress Time is the most representative of *Timespent*, the magnitude of the absolute errors shows that all three proxies have large differences with

[7]The queries used to sample the data are publicly available in our online appendix [51].

[8]Since RQ1 aimed at examining the approximation of the three proxies to the recorded *Timespent* values, and there, the analysis is independent from the SP values, therefore, we can use a different sample for RQ2 and RQ3 without loss of generality.

[9]Note that we did not filter out issues with regards to their development time proxy values from the sample used in RQ1, since the aim of RQ1 is to examine the approximation of the proxies to the recorded *Timespent* values.

Table 1: List of projects we analysed for RQ1 (a) and RQs 2-3 (b). The total number of issues in each project is shown in the *Total Issues* column. *Before Filter* shows the original number of issues extracted from the TAWOS dataset [48] and *After Filter* shows the number of issues remaining after the filtering process as explained in Section 3.3. The other columns show summary statistics for SP, Timespent and its proxies.

(a)

Repository	Project	Key	Total Issues	Issues with Timespent		Timespent (minutes)					In-Progress Time (minutes)					Effort Time (minutes)					Resolution Time (minutes)				
				Before Filter	After Filter	Min	Max	Mean	Median	SD	Min	Max	Mean	Median	SD	Min	Max	Mean	Median	SD	Min	Max	Mean	Median	SD
Atlassian	Crowd	CWD	4,311	222	220	2	4,800	468.36	240	729.33	0	1,401,053	13,258.70	44.5	101,509.06	0	1,401,053	17,341.50	2,122.5	102,618.99	7	2,724,297	185,866.38	20,465.5	436,488.49
	Jira Software Cloud	JSWCLOUD	11,702	255	244	30	2,220	383.70	300	368.79	0	219,603	2,873.85	118.5	14,352.33	0	220,762	2,878.60	118.5	14,424.37	0	1,968,722	52,691.43	5,750	253,296.60
	Jira Software Server	JSWSERVER	12,862	262	257	30	3,600	394.01	180	535.28	0	192,687	4,030.39	0	17,679.50	0	192,687	4,030.39	0	17,679.50	0	1,434,770	46,633.84	5,998	127,654.65
	Jira Server	JRASERVER	44,165	990	981	5	24,622	298.01	120	941.64	0	216,201	2,167.55	0	12,139.01	0	248,570	3,345.93	0	18,248.28	0	4,387,661	145,363.86	11,644	416,982.46
	Bamboo	BAM	14,252	524	521	5	8,460	392.57	240	694.73	0	2,278,726	11,284.45	288	103,415.46	0	455,765	12,845.10	4,253	30,649.89	1	2,635,279	78,653.98	20,072	222,346.01
	Clover	CLOV	1,501	106	106	2	4,801	605.87	240	906.33	0	259,569	15,619.25	191.5	42,533.41	0	260,743	21,726.25	7,364.5	43,420.66	0	1,350,621	85,252.67	30,289.5	171,444.39
	FishEye	FE	5,533	634	612	2	8,782	265.36	112	638.94	0	4,771,423	14,706.04	94.5	195,220.17	0	4,771,423	38,411.01	11,178.5	257,383.61	6	4,797,823	167,097.20	34,210	536,889.66
Appcelerator	Titanium Mobile Platform	TIDOC	3,059	714	711	5	11,040	307.01	120	626.55	0	579,203	7,839.01	276	32,121.50	0	579,203	10,971.92	1,399	38,699.95	0	2,463,988	131,868.77	25,699	303,406.05
Lsstcorp	Data management	DM	26,506	191	190	5	24,000	934.08	480	1,847.11	0	1,695,993	29,921.76	1,445	145,880.95	0	1,849,990	54,467.54	8,724.5	222,795.99	0	1,850,092	72,405.15	19,166.5	223,627.89
Sonatype	Nexus	NEXUS	9,912	1,356	1,348	2	4,560	189.26	90	327.95	0	495,244	2,310.59	0	21,501.77	0	495,244	2,370.53	0	21,529.01	0	5,068,853	42,930.46	4,658	202,515.39
Talendforge	Talend Data Quality	TDQ	15,315	2,054	2,053	10	18,960	764.44	480	1,144.12	0	165,637	5,119.09	1,174	11,622.38	0	916,623	42,361.35	11,562	94,348.85	1	4,002,454	179,687.61	42,085	438,753.36
	Talend Data Preparation	TDP	5,670	219	193	10	6,660	723.85	300	1,027.93	0	137,132	7,632.41	1,604	15,791.27	0	162,678	25,879.64	16,936	29,052.18	1	903,339	106,585.19	60,215	141,153.18
	Talend Data Management	TMDM	9,137	1,650	1,648	3	5,760	541.33	360	625.69	0	1,031,320	5,019.47	1,459.5	27,701.47	0	1,231,065	43,492.23	19,125	87,717.66	0	2,902,016	98,187.55	27,414	228,291.61
	Talend Big Data	TBD	4,624	193	191	5	5,700	712.36	360	943.00	0	152,267	4,539.72	1,131	12,736.24	0	1,751,027	103,679.79	56,685	192,330.37	263	1,751,054	119,209.97	53,074	227,411.31
	Talend Enterprise Service Bus	TESB	15,985	436	178	60	2,760	268.66	180	289.30	0	2,391,610	16,848.47	1,472	179,097.87	0	2,391,610	31,680.99	1,490	241,500.85	0	2,480,801	25,164.78	5,380.5	186,888.34
Total			184,534	9,806	**9,453**																				

(b)

Repository	Project	Key	Total Issues	# Issues with SP		Story Point					In-Progress Time (minutes)				
				Before Filter	After Filter	Min	Max	Mean	Median	StD	Min	Max	Mean	Median	StD
Atlassian	Jira Software Cloud	JSWCLOUD	11,702	318	185	1	20	4.19	3	3.58	2	21,931	5,212.30	2,945	5,267.85
	Confluence Server	CONFSERVER	42,324	662	362	1	13	3.03	3	1.73	2	24,847	4,511.35	1,559.5	5,922.60
	Jira Software Server	JSWSERVER	12,862	351	208	1	20	4.19	3	3.51	2	18,864	4,696.80	2,831	4,749.70
	Bamboo	BAM	14,252	528	302	1	20	2.47	2	2.18	2	20,524	4,104.67	1,455.5	5,048.14
	Clover	CLOV	1,501	387	146	1	20	3.48	2	4.09	2	30,293	5,700.54	2,848	6,655.53
Apache	Mesos	MESOS	10,157	3,272	1,157	1	13	3.32	3	2.08	2	39,861	6,311.93	1,616	8,922.99
	Usergrid	USERGRID	1,339	487	162	1	8	2.62	3	1.41	2	21,657	4,905.17	2,950.5	4,898.76
Appcelerator	Titanium Mobile Platform	TIDOC	3,059	1,297	628	1	40	4.28	3	4.14	2	46,257	6,771.88	1,553	10,171.09
	Aptana Studio	APSTUD	8,135	890	302	1	40	7.92	8	5.13	3	6,915	1,222.96	340	1,618.00
	Appcelerator Studio	TISTUD	5,979	3,406	1,918	1	34	5.69	5	4.14	2	5,243	817.76	182	1,159.13
	The Titanium SDK	TIMOB	22,059	4,665	1,753	1	21	5.56	5	2.70	2	14,255	1,927.80	240.5	3,137.66
	Appcelerator Daemon	DAEMON	313	242	131	1	99	9.57	8	11.03	2	26,184	3,403.00	379	6,141.23
DNN Tracker	DotNetNuke Platform	DNN	10,060	2,594	1,122	1	14	2.18	2	1.46	2	9,553	1,371.70	199	2,268.66
Hyperledger	Blockchain Explorer	BE	802	373	239	1	13	3.01	3	1.77	2	33,120	8,387.12	5,754	8,061.10
	Fabric	FAB	13,682	636	235	1	24	2.85	2	2.71	2	64,394	11,464.59	7,021	13,734.58
	Indy Node	INDY	2,321	681	438	1	13	3.21	3	1.73	2	38,453	9,357.24	7,190.5	8,693.04
	Sawtooth	STL	1,663	966	646	1	8	2.38	2	1.30	2	40,392	11,375.65	8,799	9,660.83
	Indy SDK	IS	1,531	720	418	1	13	3.91	3	2.13	2	22,384	4,986.39	2,917	5,339.56
Lsstcorp	Lsstcorp Data management	DM	26,506	20,664	9,019	1	100	6.16	3.2	9.58	2	105,126	18,290.56	8,083	24,287.98
Lyrasis	Lyrasis Dura Cloud	DURACLOUD	1,125	666	243	1	13	2.05	2	1.55	2	24,555	4,253.40	1,363	5,881.14
MongoDB	Compass	COMPASS	1,791	499	275	1	8	3.43	3	1.73	3	50,363	10,583.59	4,351	13,403.21
	C++ driver	CXX	2,032	224	105	1	4	1.35	1	0.65	2	14,917	2,458.21	1,098	3,587.64
	MongoDB Core Server	SERVER	48,663	784	418	1	42	2.53	2	2.76	2	18,577	3,081.26	1,056	4,419.22
	Evergreen	EVG	10,299	5,402	1,674	1	8	1.94	2	1.10	2	10,949	2,116.95	1,155	2,742.38
Mulesoft	Mule	MULE	11,816	4,170	2,105	1	21	4.95	4	3.52	2	22,982	5,195.96	3,179	5,394.67
	Mule APIkit	APIKIT	886	473	284	1	13	3.14	3	2.30	2	17,755	3,522.79	1,640	4,242.36
Sonatype	Nexus	NEXUS	9,912	1,845	421	1	15	1.56	1	1.22	2	11,467	2,054.11	518	2,811.87
Spring	XD	XD	3,707	3,705	1,602	1	24	3.37	3	2.48	2	20,942	4,160.57	1,680.5	5,038.44
Talendforge	Talend Data Quality	TDQ	15,315	1,843	1,151	1	40	5.20	5	4.28	2	27,457	6,161.18	3,945	6,663.57
	Talend Data Preparation	TDP	5,670	813	473	1	18	2.24	2	1.72	2	48,922	10,373.10	7,181	11,139.64
	Talend Data Management	TMDM	9,137	297	177	1	8	2.42	2	1.60	3	20,046	4,795.11	2,996	4,781.31
	Talend Enterprise Service Bus	TESB	15,985	1,000	309	1	13	2.28	2	1.51	2	47,937	11,166.68	8,370	11,427.15
Total			326,585	64,860	**28,608**										

Timespent. This is due to the fact that the *Timespent* field, which stores the aggregated amount of time spent on the development of the issue, is computed as the sum of the work hours logged by the developers on the issue. The proxies obtained from the TAWOS dataset are the aggregation of the duration between points in the timeline for an issue's status change, so they take into account the idle time that a developer might pause working but not change the status. For example, if an issue's status remains unchanged for a week but a developer works five hours a day on the task, they may log 25 hours for *Timespent*, while the proxies would take into account the number of days (in progress, or to resolution) in order to measure development time. Hence, the proxy may be multiple times greater than the actual *Timespent*. However, this difference in magnitude does not affect the correlation results.

Overall, considering all limitations of getting a close approximation of the actual effort in open source projects, these proxies are the most representative we could obtain from the data.

In order to verify whether the differences between *Timespent* and each of the three proxies are statistically significant, we revert to the Wilcoxon test. The results of this test are presented in Table 2 (last three columns). Since we are interested in any difference between each pair of distribution sets, we use a two-sided alternative hypothesis, therefore a comparison of P_i vs. P_j results in the same p-value as the comparison of P_j vs. P_i. As revealed by the results, the difference in the distributions of *Timespent* and In-Progress is

Table 2: RQ1. Difference between *Timespent* and the three proxy measures for development time (i.e., In-Progress Time, Effort Time, and Resolution Time) in terms of Sum Absolute Error (SAE) and significance statistical tests (effect size shown in brackets).

Project	SAE with Timespent			Timespent vs.		
	In-Progress Time	Effort Time	Resolution Time	In-Progress Time	Effort Time	Resolution Time
CWD	2,909,699	3,796,762	40,790,437	<0.001 (0.40)	<0.001 (0.60)	<0.001 (0.95)
JSWCLOUD	666,291	667,450	12,776,147	**0.285** (0.47)	**0.285** (0.47)	<0.001 (0.80)
JSWSERVER	1,024,283	1,024,283	11,903,893	<0.001 (0.23)	<0.001 (0.23)	<0.001 (0.72)
JRASERVER	2,230,661	3,389,772	142,334,494	<0.001 (0.26)	<0.001 (0.25)	<0.001 (0.92)
BAM	5,784,351	6,565,644	40,777,437	**0.361** (0.52)	<0.001 (0.68)	<0.001 (0.96)
CLOV	1,641,549	2,261,458	8,973,869	**0.872** (0.49)	<0.001 (0.67)	<0.001 (0.93)
FE	8,961,125	23,375,131	102,102,251	**0.968** (0.50)	<0.001 (0.86)	<0.001 (0.99)
TIDOC	5,501,120	7,703,434	93,548,470	**0.085** (0.53)	<0.001 (0.61)	<0.001 (0.96)
DM	5,578,740	10,194,085	13,588,070	**0.043** (0.56)	<0.001 (0.77)	<0.001 (0.91)
NEXUS	3,114,794	3,193,031	57,646,533	<0.001 (0.31)	<0.001 (0.32)	<0.001 (0.85)
TDQ	9,568,980	85,743,427	367,355,182	<0.001 (0.54)	<0.001 (0.82)	<0.001 (0.96)
TDP	1,358,593	4,863,758	20,432,471	<0.001 (0.67)	<0.001 (0.91)	<0.001 (0.98)
TMDM	7,521,187	70,814,422	160,926,609	<0.001 (0.65)	<0.001 (0.92)	<0.001 (0.96)
TBD	784,328	19,667,618	22,633,044	**0.020** (0.57)	<0.001 (0.99)	<0.001 (0.99)
TESB	2,963,686	5,602,799	4,438,478	<0.001 (0.70)	<0.001 (0.72)	<0.001 (0.80)

significant in eight out of 15 projects, all with small or negligible effect size, except for the TESB project for which the effect size is medium. However, the difference between *Timespent* and the Effort time proxy is significant in 14 out of 15 cases, with JSWCLOUD being the only exception given that the recorded Effort time is very close to In-Progress time in most of the issues belonging to this project. Moreover, the difference is significant for all the cases when comparing *Timespent* with the Resolution time. Out of 30 cases of statistical tests on Effort Time and Resolution Time, 19 cases show a large effect size, three cases a medium effect size, and 7 cases a small or negligible effect size.

As a result, we only consider In-Progress in our subsequent research questions given that it is the most representative of *Timespent* compared to the other two proxies.

4.2 RQ2. Correlation

The results of three correlation statistics (RQ2) are shown in Table 3. We also reported the p-value for each correlation coefficient.

For all projects, Kendall's τ is consistent with the Spearman's ρ in the scale and confidence level. However, if we consider Gilpin's τ to ρ conversion table [22], we would expect a higher ρ. For example, in the case of the CONFSERVER project (see Table 3), Gilpin's table maps a $\tau = 0.26$ to $\rho = 0.38$, while our data lead to a $\rho = 0.34$. The rationale behind this is the fact that Kendall's τ is the proportion of the concordant to discordant pairs while the Spearman's ρ considers the variance in the ranks. Hence, as we obtain a ρ smaller than expected (indicated by ρ to τ rate) it shows the high variance in the ranks of the data, to which Spearman is sensitive but Kendall is not. This high variance in the ranks is a sign of misclassification of many issues by human-estimators in wrong SP classes, thus an error in the estimation.

The Pearson correlation coefficient is lower than the Spearman's ρ for 24 projects (75% of the cases), which indicates that the relation between the story point and development time is not usually linear.

As we can observe, the correlation denoted by Spearman's ρ for In-Progress time is low in six out of 32 cases, medium in 21 cases and strong for only five cases. The strongest positive correlation appears to be in projects DAEMON, INDY, JSWCLOUD, JSWSERVER, and DURACLOUD. Looking at the p-value of the Spearman's ρ, we find

Table 3: RQ2. Correlation results between SP and In-Progress Time (p-value in brackets). Medium and strong correlations are highlighted in orange and red, respectively.

Project	In-Progress Time Correlation with Story Point		
	Spearman's ρ	Kendall's τ	Pearson r
JSWCLOUD	0.54 (<0.001)	0.41 (<0.001)	0.47 (<0.001)
CONFSERVER	0.34 (<0.001)	0.26 (<0.001)	0.26 (<0.001)
JSWSERVER	0.53 (<0.001)	0.40 (<0.001)	0.49 (<0.001)
BAM	0.35 (<0.001)	0.28 (<0.001)	0.35 (<0.001)
CLOV	0.45 (<0.001)	0.34 (<0.001)	0.44 (<0.001)
MESOS	0.40 (<0.001)	0.30 (<0.001)	0.35 (<0.001)
USERGRID	0.20 (0.013)	0.16 (0.008)	0.12 (0.139)
TIDOC	0.48 (<0.001)	0.36 (<0.001)	0.27 (<0.001)
APSTUD	0.38 (<0.001)	0.30 (<0.001)	0.40 (<0.001)
TISTUD	0.42 (<0.001)	0.33 (<0.001)	0.35 (<0.001)
TIMOB	0.28 (<0.001)	0.22 (<0.001)	0.23 (<0.001)
DAEMON	0.62 (<0.001)	0.48 (<0.001)	0.66 (<0.001)
DNN	0.32 (<0.001)	0.25 (<0.001)	0.27 (<0.001)
BE	0.19 (0.003)	0.15 (0.002)	0.21 (0.001)
FAB	0.49 (<0.001)	0.38 (<0.001)	0.36 (<0.001)
INDY	0.56 (<0.001)	0.44 (<0.001)	0.53 (<0.001)
STL	0.41 (<0.001)	0.32 (<0.001)	0.39 (<0.001)
IS	0.49 (<0.001)	0.38 (<0.001)	0.43 (<0.001)
DM	0.49 (<0.001)	0.36 (<0.001)	0.42 (<0.001)
DURACLOUD	0.52 (<0.001)	0.41 (<0.001)	0.47 (<0.001)
COMPASS	0.30 (<0.001)	0.23 (<0.001)	0.25 (<0.001)
CXX	0.27 (0.005)	0.22 (0.006)	0.36 (<0.001)
SERVER	0.49 (<0.001)	0.37 (<0.001)	0.29 (<0.001)
EVG	0.36 (<0.001)	0.28 (<0.001)	0.28 (<0.001)
MULE	0.48 (<0.001)	0.36 (<0.001)	0.49 (<0.001)
APIKIT	0.37 (<0.001)	0.28 (<0.001)	0.30 (<0.001)
NEXUS	0.25 (<0.001)	0.19 (<0.001)	0.25 (<0.001)
XD	0.41 (<0.001)	0.31 (<0.001)	0.38 (<0.001)
TDQ	0.44 (<0.001)	0.32 (<0.001)	0.36 (<0.001)
TDP	0.36 (<0.001)	0.27 (<0.001)	0.23 (<0.001)
TMDM	0.18 (0.016)	0.14 (0.015)	0.23 (0.002)
TESB	0.33 (<0.001)	0.26 (<0.001)	0.24 (<0.001)
Min	0.18	0.14	0.12
Max	0.62	0.48	0.66
Mean	0.40	0.31	0.35
SD	0.11	0.08	0.11

that the confidence level is above 99% for 30 out of the 32 projects under study (94% of the cases).

As a subsequent analysis we computed the three correlation statistics on all the issues from the TAWOS dataset that have reported both the SP and *Timespent* values. This results in a total

Table 4: RQ2. Correlation results between SP and *Timespent* (p-value in brackets). Medium and strong correlations are highlighted in orange and red, respectively.

Project	*Timespent* Correlation with Story Point		
	Spearman's ρ	Kendall's τ	Pearson r
DM	0.33 (<0.001)	0.28 (<0.001)	0.35 (<0.001)
MDL	0.34 (<0.001)	0.27 (<0.001)	0.11 (0.179)
TDQ	0.64 (<0.001)	0.52 (<0.001)	0.61 (<0.001)
TMDM	0.03 (0.729)	0.03 (0.688)	0.28 (0.004)

of 697 issues from four projects (specifically, 128 issues from DM, 303 issues from TDQ, 104 issues from TMDM, and 162 issues from MDL). Although the number of such issues is not prevalent, it gives us an indication of how much results of RQ2 can be resembled by actual development time values (i.e., *Timespent*).

The result of this correlation analysis is shown in Table 4. As we can see, only one out of four projects showed a strong correlation with respect to all three statistics. From the other three projects, two show a medium range Spearman's ρ and one a medium range Pearson's r coefficient. For the rest of the cases (i.e., 50%) a low correlation is obtained between SP values and actual *Timespent*.

4.3 RQ3. Consistency

Figure 2 shows the boxplots of the development time distribution for each story point value for six sample projects.[10]. As previously explained (Section 3.2), for an acceptable story point estimation error, each of these boxplots should resemble a normal distribution, and the relation between the median of each box should be proportional to the value of the story point class. The ideal projection of median development time per each SP class is depicted by a line connecting the diamonds in Figure 2. This projection is computed by multiplying the SP value with the median development time for all issues estimated to have one story point.

From the boxplots, we can observe that this proportion does not hold for most of the classes. Besides, the distribution of each class tends to be heavily tailed. This observation is confirmed by the Shapiro-Wilk test [36], which has revealed that the data is not normally distributed for any of the projects [51]. Specifically, only for seven projects out of the 32 under study, the median development time per SP class (depicted by median line inside each boxplot) falls in the vicinity of the ideal projected value (for example see projects CLOV and JSWCLOUD in Figure 2). For four other projects, the projection line falls well above the actual median development time (e.g., the BE project in Figure 2), indicating an over-estimation. While for the remaining 21 projects, the projection line falls well below the actual median development time (e.g., the APIKIT, XD and MULE projects in Figure 2). This high number shows the tendency for human-experts to generally underestimate the time required to complete a certain task.

We further analyse this phenomenon by fitting a regression line to the median development time of each class against the value of the story point. We then fit another regression line considering only the classes of SP with values less than five. As the angle between these two lines widens, the consistency between story point classes

[10]Due to space, the boxplots for all projects can be found in our on-line appendix [51].

Table 5: RQ3. Angles created between the X axis and the linear regression fit for SP classes against Median In-Progress time when only SP classes ≤ 5 are considered (Angle (SP≤ 5)) and when all the classes are considered (Angle (SP≤ 100)), and the angle between the two (Difference).

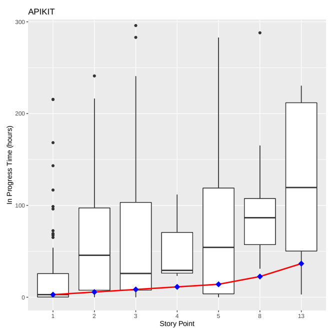
Project	Angle (SP$\leq$ 5)	Angle (SP$\leq$ 100)	Difference
JSWCLOUD	40.41°	24.22°	**16.19°**
CONFSERVER	48.35°	20.77°	**27.58°**
JSWSERVER	41.12°	24.48°	**16.64°**
BAM	57.40°	21.57°	**35.83°**
CLOV	57.51°	22.65°	**34.86°**
MESOS	54.04°	39.22°	**14.82°**
USERGRID	26.17°	38.78°	**12.61°**
TIDOC	23.57°	23.73°	0.16°
APSTUD	-4.67°	5.71°	10.38°
TISTUD	0.72°	7.78°	7.06°
TIMOB	1.45°	6.79°	5.34°
DAEMON	1.40°	11.14°	9.74°
DNN	26.08°	7.92°	**18.16°**
BE	48.00°	45.96°	2.04°
FAB	78.33°	19.39°	**58.94°**
INDY	66.94°	63.87°	3.07°
STL	65.47°	54.82°	10.65°
IS	46.96°	25.49°	**21.47°**
DM	75.48°	26.95°	**48.53°**
DURACLOUD	56.49°	46.08°	10.41°
COMPASS	47.02°	59.73°	**12.71°**
CXX	64.53°	64.53°	0.00°
SERVER	30.39°	7.43°	**22.96°**
EVG	36.15°	24.12°	12.03°
MULE	29.20°	31.42°	2.22°
APIKIT	19.81°	20.73°	0.92°
NEXUS	17.80°	23.40°	5.60°
XD	31.91°	17.83°	**14.08°**
TDQ	50.19°	11.11°	**39.08°**
TDP	63.45°	-0.90°	**64.35°**
TMDM	32.77°	37.86°	5.09°
TESB	69.70°	7.97°	**61.73°**

becomes lower. In contrast, if these two lines are aligned together for a project, we can say that the consistency of estimation in lower SP classes is maintained for higher SP classes. This is based on the premise that human experts are better at estimating smaller tasks than the bigger ones. Plots of the regression fits for all projects can be found in our online appendix [51].

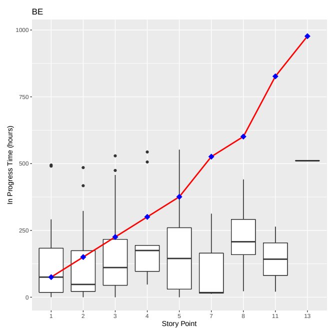
We also report, in Table 5, the angles each of these lines creates with the x-axis as well as the deviation in the trajectory (i.e., difference between the two angles).

We can observe that the angle between the two lines is wider than 12.5° for more than half the projects (53% of the cases). This signifies that there is a notable shift in the trajectory of the regression fit of those projects taking into account the median development time for the higher SP classes. Thus, the scale in which the issues in the higher SP classes are estimated is not consistent with the issues estimated to be in the smaller SP classes. Therefore, based on these observations, we recommend that development teams consider breaking down bigger issues into smaller ones before they attempt to estimate the story point.

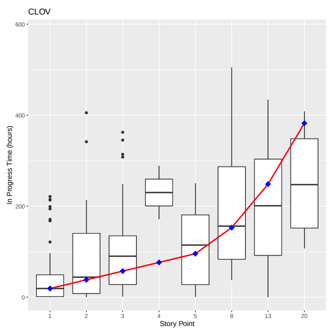
It is also worth noting that using data consisting of estimated SP to train a predictive model, would result in that model imitating human expert misestimates and therefore possibly achieving biased results.



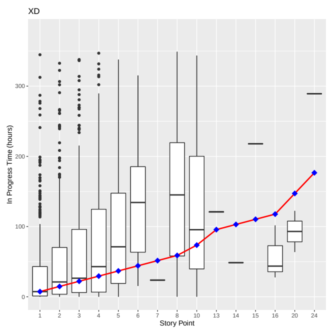
(a) APIKIT



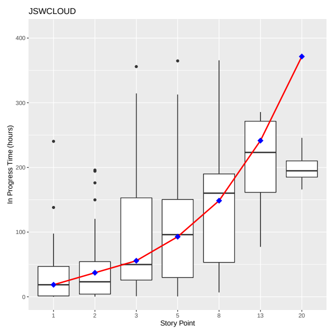
(b) BE



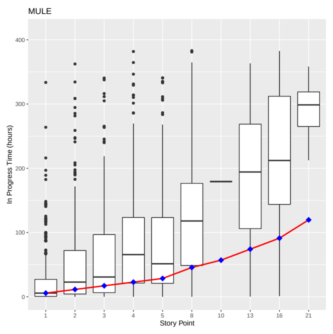
(c) CLOV



(d) XD



(e) JSWCLOUD



(f) MULE

Figure 2: Boxplots of the distribution of development time per SP class for (a) APIKIT, (b) BE, (c) CLOV, (d) XD, (e) JSWCLOUD, (f) MULE. The red line depicts a project-specific baseline, drawn based on the median development time for one SP.

5 RELATED WORK

In this section, we discuss previous studies relevant to ours that (1) analyse the relationship between human-expert estimated SP, functional size measure (such as FP and COSMIC) and development time/effort; (2) use human-expert estimated SP as a cost driver for automated Agile software effort estimation models; (3) use machine learning models to predict the SP of user stories.

5.1 Software Size Measures and Agile Development Effort

The first study assessing the relationship between SP and functional size measures (FSM) was carried out in 2011 by Santana et al. [38]. The authors of this study quantitatively analysed the relationship between FP and SP in a case study involving 2,191 user stories from 18 iterations of an Agile software projects developed by a private company. They found a strong positive correlation between SP and FP (Spearman's $\rho = 0.71$). Subsequently, Huijgens and Solingen [26] replicated Santana et al.'s work on a different case study and found a contrasting result. They gathered data from 14 iterations performed by two teams (*A* and *B*) in a Dutch banking organization that recorded estimations in SP, and computed the size in FP for all iterations. The results of the Spearman's rank correlation revealed a medium (−0.36) and strong (−0.60) negative correlation between SP and FP for Teams *A* and *B*, respectively.

SP has been also compared to COSMIC, and actual effort. Salmanoglu et al. [37] compared the correlation of SP and CFP with the actual effort spent for Agile software development. They carried out three case studies from three large Turkish companies producing software solutions for security, financial, and telecommunication industries, reporting CFP, SP and the actual effort measured in person-hours. They plotted SP and CFP values against the actual effort to measure the linearity of functional size and actual effort, and observed a stronger correlation between CFP and actual effort, in comparison with SP.

Choetkiertikul et al. [10] carried out a preliminary analysis on the relationship between SP and development time on a set of 16 open-source projects mined form Jira repositories for a total of 23,313 issues [10]. The main aim of their study, however, was to build a machine learning model to estimate SP. They computed development time from the issue changelog by considering the duration between the time the issue's status was set to *In-Progress* and the time it was set to *Resolved*. This was regarded by the authors as the most representative proxy for the actual effort they were able to extract from the data with respect to the completion time of an issue. However, this definition also includes the waiting time between development stages as part of the development time. In this paper, we adopt a similar proxy for the issue resolution time, and use two additional proxies for development time to take into account the waiting time (which is usually considered part of the development time) as proposed by Tawosi et al. [48]. Choetkiertikul et al. [10] found a positive correlation between the SP and their proxy for development time with a mean of 0.47 and 0.51 and a standard deviation of 0.19 and 0.18, according to the Spearman's rank and Pearson correlation, respectively. Our investigation on a larger dataset showed a positive but weaker correlation than the one previously found [10], specially based on the Pearson correlation coefficient. Across the 32 projects investigated herein, we obtained a mean value of 0.40 for the Spearman's ρ and 0.35 for the Pearson's r coefficient, with a standard deviation of 0.11 for both.

5.2 Automated Story Point Prediction

Several studies in the literature have used automated estimation techniques to predict SP of issues in Agile software projects.

Abrahamsson et al. [3] used several features extracted from user stories, including their SP, to train a model which estimates SP for new stories. Porru et al. [34] built classification models to classify user stories into SP classes. Scott and Pfahl [40] used developer-related features alongside the features extracted from user stories to estimate SP using machine learning. Soares [43] used Autoencoder Neural Networks to classify user stories based on the semantic differences of their titles in order to estimate their SP size. Choetkiertikul et al. [10] combined two deep learning architectures, to build an end-to-end prediction system for SP size of user stories, called Deep-SE. Abadeer and Sabetzadeh [2] used Deep-SE [10] for SP prediction of a closed-source project with 4,727 user stories. Tawosi et al. [50] replicated Choetkiertikul et al.'s study [10] by evaluating Deep-SE on a larger dataset of open-source projects. In another study, Tawosi et al. [49], used a clustering-based method to estimate SP for issue reports aiming at improving the performance of Deep-SE. Marapelli et al. [29] built a model based on a tree-structured RNN with Convolutional Neural Network (CNN) to predict SP for user stories. This model adopts a Bi-directional LSTM (BiLSTM) which improves Deep-SE's prediction performance. More recently, Fu and Tantithamthavorn [20] proposed GPT2SP, a Transformer-based deep learning model for SP estimation of user stories and found that this model outperforms Deep-SE.

5.3 Story Point as a Cost Driver for Agile Effort Estimation

Zia et al. [58] considered human-expert estimated story size and story complexity to compute SP for user stories and they used it to estimate the actual effort and cost for software projects. They introduced a regression-based model considering characteristics of agile development. The model was applied to 21 previously developed small software projects and produced estimations with a mean absolute error of four days. Later, Popli and Chauhan [33] proposed a similar approach but evaluated the model on one small project.

Ungan et al. [54], investigated the accuracy of multiple linear estimators and a simple Artificial Neural Network estimator on 10 industrial projects as a case study. All the projects had their actual effort and SP recorded by developers and their CFP were automatically approximated using a tool named CUBIT. Results showed that when the estimator uses CFP or SP, as independent variables, the accuracy of effort estimation is low or at most acceptable, and none of the two models is superior to the other.

Raslan et al. [35] proposed a fuzzy logic technique-based effort estimation framework for user stories. The approach feeds expert estimated SP alongside other parameters as input to a trapezoidal membership function to estimate the actual effort. The model is not evaluated on any real data; however, the authors designed a framework based on the proposed model on MATLAB, to make it ready for evaluation and possible adoption.

Satapathy et al. [39] used a dataset of 21 projects from the work of Zia et al. [58] and evaluated the effort estimation accuracy of different machine learning techniques, namely, Decision Trees (DT), Stochastic Gradient Boosting (SGB), and Random Forest (RF). Based on the results, the DT model underperformed the technique previously proposed by Zia et al. [58]. Whereas the SGB and RF models performed better than it.

6 CONCLUSIONS

We have studied the relationship between human-expert estimated Story Point (SP) and the time required by the developers to realise a given issue (i.e., *development time*) on a large sample of open-source user stories sampled from the TAWOS public dataset [48], which consists of 37 software projects for different application domains, diversified in size and characteristics, resulting in a total of 37,440 unique issues.

The results of our empirical study showed that, among the three proxies for development time we studied herein, In-Progress time is the most representative of the actual development time recorded by the developers. When considering its correlation with human-expert estimated SP we found that for the majority of the projects such a correlation is low (35%) or medium (58%). Analysing the correlation between SP and the actual development time unveiled a similar outcome: SP showed a low (50%) or medium (25%) correlation with *Timespent*. We also found that majority of the investigated projects (25 out of 32) lack consistent human-expert estimations for SP. The consistency starts to wear when the issues are estimated to be bigger than five points, thereby suggesting that human estimators are not accurate at assessing the size of the issues that need five times or more effort than an issue worth a single story point. To overcome this issue Agile teams can try to break-down all tasks/issues estimated to be bigger than five SP into smaller ones.

The above results provide empirical evidence that human-expert estimated SP might not be a good indicator for the issue development effort of Agile open-source projects. This might render any machine-learnt effort estimation model, which learns from human-expert estimated SP, vulnerable to the same bias and its impact should be taken into account in future work. It would be interesting, for example, to assess if more accurate effort estimation models can be obtained by using the development time instead of SP as a cost driver. Moreover, future work could replicate our study by considering industrial projects to expand the understanding beyond the open-source realm investigated herein. Also future work could involve expert certified FSM measurers to compute the FP and CFP of the user stories available in the TAWOS dataset, so that one could carry out a large scale empirical study analysing the correlation between SP, FP, CFP, and actual development time.

In order to allow for replication and extension of our work, we make our data and scripts publicly available [51].

ACKNOWLEDGMENTS

Vali Tawosi, Rebecca Moussa and Federica Sarro are supported by the ERC grant no. 741278 (EPIC).

REFERENCES

[1] [n.d.]. Jira | Issue & Project Tracking Software | Atlassian. https://www.atlassian.com/software/jira

[2] Macarious Abadeer and Mehrdad Sabetzadeh. 2021. Machine Learning-based Estimation of Story Points in Agile Development: Industrial Experience and Lessons Learned. In *2021 IEEE 29th International Requirements Engineering Conference Workshops (REW)*. IEEE, 106–115.

[3] Pekka Abrahamsson, Ilenia Fronza, Raimund Moser, Jelena Vlasenko, and Witold Pedrycz. 2011. Predicting development effort from user stories. In *2011 International Symposium on Empirical Software Engineering and Measurement*. IEEE, 400–403.

[4] Silvia Abrahão, Lucia De Marco, Filomena Ferrucci, Jaime Gómez, Carmine Gravino, and Federica Sarro. 2018. Definition and evaluation of a COSMIC measurement procedure for sizing Web applications in a model-driven development environment. *Information and Software Technology* 104 (2018), 144–161.

[5] Allan J Albrecht. 1979. Measuring application development productivity. In *Proc. Joint Share, Guide, and IBM Application Development Symposium, 1979*.

[6] Allan J. Albrecht and John E Gaffney. 1983. Software function, source lines of code, and development effort prediction: a software science validation. *IEEE transactions on software engineering* 6 (1983), 639–648.

[7] Andrea Arcuri and Lionel Briand. 2014. A hitchhiker's guide to statistical tests for assessing randomized algorithms in software engineering. *Software Testing, Verification and Reliability* 24, 3 (2014), 219–250.

[8] Kent Beck and Martin Fowler. 2001. *Planning extreme programming*. Addison-Wesley Professional.

[9] Sarah Boslaugh. 2012. *Statistics in a nutshell: A desktop quick reference*. " O'Reilly Media, Inc.".

[10] Morakot Choetkiertikul, Hoa Khanh Dam, Truyen Tran, Trang Pham, Aditya Ghose, and Tim Menzies. 2018. A deep learning model for estimating story points. *IEEE Transactions on Software Engineering* 45, 7 (2018), 637–656.

[11] Jacob Cohen. 2013. *Statistical power analysis for the behavioral sciences*. Academic press.

[12] Mike Cohn. 2005. *Agile estimating and planning*. Pearson Education.

[13] Christophe Croux and Catherine Dehon. 2010. Influence functions of the Spearman and Kendall correlation measures. *Statistical methods & applications* 19, 4 (2010), 497–515.

[14] S. Di Martino, Filomena Ferrucci, Carmine Gravino, and Federica Sarro. 2016. Web Effort Estimation: Function Point Analysis vs. COSMIC. *Information and Software Technology* 72 (2016), 90–109.

[15] Sergio Di Martino, Filomena Ferrucci, Carmine Gravino, and Federica Sarro. 2016. Web effort estimation: function point analysis vs. COSMIC. *Information and Software Technology* 72 (2016), 90–109.

[16] Sergio Di Martino, Filomena Ferrucci, Carmine Gravino, and Federica Sarro. 2020. Assessing the Effectiveness of Approximate Functional Sizing Approaches for Effort Estimation. *Information and Software Technology* 123 (2020).

[17] F. Ferrucci, C. Gravino, P. Salza, and F. Sarro. 2015. Investigating Functional and Code Size Measures for Mobile Applications. In *Proceedings of Euromicro Conference on Software Engineering and Advanced Applications*. 365–368.

[18] Filomena Ferrucci, Carmine Gravino, Pasquale Salza, and Federica Sarro. 2015. Investigating Functional and Code Size Measures for Mobile Applications: A Replicated Study. In *Product-Focused Software Process Improvement*, Pekka Abrahamsson, Luis Corral, Markku Oivo, and Barbara Russo (Eds.). Springer International Publishing, Cham, 271–287.

[19] Martin Fowler, Jim Highsmith, et al. 2001. The agile manifesto. *Software Development* 9, 8 (2001), 28–35.

[20] Michael Fu and Chakkrit Tantithamthavorn. 2022. GPT2SP: A Transformer-Based Agile Story Point Estimation Approach. *IEEE Transactions on Software Engineering* (2022).

[21] Michael Gammage. 2011. Why Your IT Project May Be Riskier Than You Think. *HARVARD BUSINESS REVIEW* 89, 11 (2011), 22–22.

[22] Andrew R Gilpin. 1993. Table for conversion of Kendall's Tau to Spearman's Rho within the context of measures of magnitude of effect for meta-analysis. *Educational and psychological measurement* 53, 1 (1993), 87–92.

[23] Mayy Habayeb, Syed Shariyar Murtaza, Andriy Miranskyy, and Ayse Basar Bener. 2017. On the use of hidden markov model to predict the time to fix bugs. *IEEE Transactions on Software Engineering* 44, 12 (2017), 1224–1244.

[24] Alaa El Deen Hamouda. 2014. Using agile story points as an estimation technique in cmmi organizations. In *2014 agile conference*. IEEE, 16–23.

[25] James F Hemphill. 2003. Interpreting the magnitudes of correlation coefficients. (2003).

[26] Hennie Huijgens and Rini van Solingen. 2014. A replicated study on correlating agile team velocity measured in function and story points. In *Proceedings of the 5th International Workshop on Emerging Trends in Software Metrics*. 30–36.

[27] Maurice G Kendall. 1938. A new measure of rank correlation. *Biometrika* 30, 1/2 (1938), 81–93.

[28] Youngseok Lee, Suin Lee, Chan-Gun Lee, Ikjun Yeom, and Honguk Woo. 2020. Continual prediction of bug-fix time using deep learning-based activity stream embedding. *IEEE Access* 8 (2020), 10503–10515.

[29] Bhaskar Marapelli, Anil Carie, and Sardar MN Islam. 2020. RNN-CNN MODEL: A Bi-directional Long Short-Term Memory Deep Learning Network For Story Point Estimation. In *2020 5th International Conference on Innovative Technologies in Intelligent Systems and Industrial Applications (CITISIA)*. IEEE, 1–7.

[30] Patrick E McKnight and Julius Najab. 2010. Mann-Whitney U Test. *The Corsini encyclopedia of psychology* (2010), 1–1.

[31] Marco Ortu, Giuseppe Destefanis, Bram Adams, Alessandro Murgia, Michele Marchesi, and Roberto Tonelli. 2015. The jira repository dataset: Understanding social aspects of software development. In *Proceedings of the 11th international conference on predictive models and data analytics in software engineering*. 1–4.

[32] K Pearson. 1895. Notes on Regression and Inheritance in the Case of Two Parents Proceedings of the Royal Society of London, 58, 240-242.

[33] Rashmi Popli and Naresh Chauhan. 2014. Cost and effort estimation in agile software development. In *2014 international conference on reliability optimization and information technology (ICROIT)*. IEEE, 57–61.

[34] Simone Porru, Alessandro Murgia, Serge Demeyer, Michele Marchesi, and Roberto Tonelli. 2016. Estimating story points from issue reports. In *Proceedings of the The 12th International Conference on Predictive Models and Data Analytics in Software Engineering*. 1–10.

[35] Atef Tayh Raslan, Nagy Ramadan Darwish, and Hesham Ahmed Hefny. 2015. Towards a fuzzy based framework for effort estimation in agile software development. *International Journal of Computer Science and Information Security* 13, 1 (2015), 37.

[36] J Patrick Royston. 1982. An extension of Shapiro and Wilk's W test for normality to large samples. *Journal of the Royal Statistical Society: Series C (Applied Statistics)* 31, 2 (1982), 115–124.

[37] Murat Salmanoglu, Tuna Hacaloglu, and Onur Demirors. 2017. Effort estimation for agile software development: Comparative case studies using COSMIC functional size measurement and story points. In *Proceedings of the 27th International Workshop on Software Measurement and 12th International Conference on Software Process and Product Measurement*. 41–49.

[38] Célio Santana, Fabiana Leoneo, Alexandre Vasconcelos, and Cristine Gusmão. 2011. Using function points in agile projects. In *International Conference on Agile Software Development*. Springer, 176–191.

[39] Shashank Mouli Satapathy and Santanu Kumar Rath. 2017. Empirical assessment of machine learning models for agile software development effort estimation using story points. *Innovations in Systems and Software Engineering* 13, 2 (2017), 191–200.

[40] Ezequiel Scott and Dietmar Pfahl. 2018. Using developers' features to estimate story points. In *Proceedings of the 2018 International Conference on Software and System Process*. 106–110.

[41] Reza Sepahvand, Reza Akbari, and Sattar Hashemi. 2020. Predicting the bug fixing time using word embedding and deep long short term memories. *IET Software* 14, 3 (2020), 203–212.

[42] Martin Shepperd and Steve MacDonell. 2012. Evaluating prediction systems in software project estimation. *Information and Software Technology* 54, 8 (2012), 820–827.

[43] Rodrigo GF Soares. 2018. Effort Estimation via Text Classification And Autoencoders. In *2018 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 01–08.

[44] Ian Sommerville. 2016. *Software Engineering GE*. Pearson Australia Pty Limited.

[45] Charles Spearman. 1961. The proof and measurement of association between two things. (1961).

[46] Charles Symons. 2019. The COSMIC Method for Measuring the Work-Output Component of Productivity. In *Rethinking Productivity in Software Engineering*. Springer, 191–204.

[47] Charles Symons, Alain Abran, Christof Ebert, and Frank Vogelezang. 2016. Measurement of software size: advances made by the COSMIC community. In *2016 Joint Conference of the International Workshop on Software Measurement and the International Conference on Software Process and Product Measurement (IWSM-MENSURA)*. IEEE, 75–86.

[48] Vali Tawosi, Afnan Al-Subaihin, Rebecca Moussa, and Federica Sarro. 2022. A Versatile Dataset of Agile Open Source Software Projects. In *Proceedings of the 19th International Conference on Mining Software Repositories (MSR '22)*. IEEE.

[49] Vali Tawosi, Afnan Al-Subaihin, and Federica Sarro. 2022. Investigating the Effectiveness of Clustering for Story Point Estimation. In *Proceedings of the 29th IEEE International Conference on Software Analysis, Evolution and Reengineering*. IEEE, 816–827.

[50] Vali Tawosi, Rebecca Moussa, and Federica Sarro. 2022. Deep Learning for Agile Effort Estimation Have We Solved the Problem Yet? arXiv:2201.05401 [cs.SE]

[51] Vali Tawosi, Rebecca Moussa, and Federica Sarro. 2022. Online Appendix containing Data and R Scripts for this study. https://github.com/SOLAR-group/SPvsDevelopmentEffort.git

[52] Vali Tawosi, Federica Sarro, Alessio Petrozziello, and Mark Harman. 2021. Multi-objective software effort estimation: a replication study. *IEEE Transactions on Software Engineering* (2021).

[53] Adam Trendowicz and Ross Jeffery. 2014. Software project effort estimation. *Foundations and Best Practice Guidelines for Success, Constructive Cost Model–COCOMO pags* (2014), 277–293.

[54] Erdir Ungan, Numan Çizmeli, and Onur Demirörs. 2014. Comparison of functional size based estimation and story points, based on effort estimation effectiveness in SCRUM projects. In *2014 40th EUROMICRO Conference on Software Engineering and Advanced Applications*. IEEE, 77–80.

[55] Muhammad Usman, Emilia Mendes, and Jürgen Börstler. 2015. Effort estimation in agile software development: a survey on the state of the practice. In *Proceedings of the 19th international conference on Evaluation and Assessment in Software Engineering*. 1–10.

[56] Muhammad Usman, Emilia Mendes, Francila Weidt, and Ricardo Britto. 2014. Effort estimation in agile software development: a systematic literature review. In *Proceedings of the 10th international conference on predictive models in software engineering*. 82–91.

[57] Jie M Zhang, Feng Li, Dan Hao, Meng Wang, Hao Tang, Lu Zhang, and Mark Harman. 2019. A study of bug resolution characteristics in popular programming languages. *IEEE Transactions on Software Engineering* 47, 12 (2019), 2684–2697.

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