Using Developers' Features to Estimate Story Points

Ezequiel Scott
University of Tartu
Tartu, Estonia
ezequiel.scott@ut.ee

Dietmar Pfahl
University of Tartu
Tartu, Estonia
dietmar.pfahl@ut.ee

ABSTRACT

Effort estimation is important to correctly plan the use of resources in a software project. In agile projects, a correct effort estimation helps decide which issues have to be fixed or finished during the next iteration. However, estimating issues can be a complex task and developers may make inaccurate estimates. Therefore, the use of automatic approaches that aim to support developers in the estimation process is worth to be studied. We explore the use of a predictive model that use developers' features to assign story points to issue reports. The performance of the model is compared with the performance of models based on features extracted from the text of issues. We assessed the models with different performance metrics including Accuracy, Mean Absolute Error, and Standardized Accuracy. The preliminary results show that the model that uses developers' features sightly outperforms the models based on text features, indicating a promising research direction.

KEYWORDS

Effort Estimation, Agile Software Development, Machine Learning

ACM Reference Format:
Ezequiel Scott and Dietmar Pfahl. 2018. Using Developers' Features to Estimate Story Points. In *ICSSP '18: International Conference on the Software and Systems Process 2018 (ICSSP '18), May 26–27, 2018, Gothenburg, Sweden.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3202710.3203160

1 INTRODUCTION

Effort estimation is crucial in any development process to correctly plan the use of resources. In particular, effort estimation plays an important role in agile projects since they usually organize the development in iterations which are defined by the teams according to the stakeholders' goals and the effort estimation. In agile, effort estimation is often done by teams applying techniques such as planning poker and made in terms of story points [5, 19].

Story points are a unit of measure for expressing an estimate of the overall effort that will be required to fully implement a piece of work [5, 17]. In practice, agile teams assign story points to not only user stories but also other individual pieces of work that they must complete. Project management tools usually name these pieces of work as issue reports. Depending on how teams use these tools, an issue could represent a user story, a bug report, a project task, or a helpdesk ticket, among others.

ICSSP '18, May 26–27, 2018, Gothenburg, Sweden

The effort estimation process has been considered to be a process in which developers' expertise and previous knowledge have a strong influence on estimation accuracy. Therefore, estimating issues can be problematic for novice developers since they do not have enough experience. As a result, novice developers –and sometimes even senior ones– guess the number of story points when they have to estimate issues. In this context, the use of automatic approaches that aim to support developers in the estimation process is worth to be studied.

In order to aid developers during the estimation process, we propose to use a predictive model whose output can be a suggestion for novice developers or be considered as an extra input during a planning poker session. To build the prediction model, we use a supervised approach that takes as input features derived from the issue reports of eight Open Source projects. We explore the use of different sets of features and evaluate their performance to determine the best one, including not only features from the textual descriptions of the issue reports but also about the developers who are involved. Using developer features such as reputation and workload seems to be reasonable since we assume that the estimation process is affected by the individual characteristics of the developer.

To determine whether developer features have a positive effect on prediction performance or not, we compare the results of several models that are trained with a different set of features: a set of developer features, set of text features, and a set given by the combination of both. The comparison is made by using different performance metrics such as Accuracy, Mean Absolute Error (MAE) and Standardized Accuracy (SA). After analyzing the results, we conclude that the models which use developer features to predict story points outperform the models which use features extracted from the text. These preliminary results indicate that further research on the individual characteristics of the developer to predict story points is worth to be done.

2 BACKGROUND

The traditional way of estimating effort is to give a date when a task will be finished. This, however, does not account for the fact that during the work a team member will attend meetings, read e-mails and do other work-related activities. Therefore, story points are often used as a proxy measure for both effort and complexity. Typically, story points are assigned according to a Fibonacci number sequence, where each number is the sum of the previous two [5].

There are techniques to estimate story points, such as planning poker [5, 10]. In planning poker, a user story (or issue) is chosen to be discussed. Then the developers individually choose the number of story points. Once all team members have chosen their estimates, the choices are disclosed. The developers, who chose the lowest and the highest story points, must justify their choices. This process

repeats until a consensus is achieved and the agreed number of story points is assigned to the user story (or issue).

Nowadays, the story points and all other relevant information about issues is managed using software tools. Among a wide range of popular tools, JIRA[1] is frequently used for tracking issues in open source projects. An issue according to JIRA could represent a software bug, a user story, or a custom type of issue. JIRA supports many fields to describe issues such as summary, description, type, status, priority, assignee, creator, among many others.

3 RELATED WORK

A considerable amount of literature has been published on effort estimation. Effort estimation is still attractive to researchers since a reliable estimation process is crucial for a correct project planning and a good management of the resources [15, 18].

Many studies have investigated the use of machine learning to build predictive models in software engineering, aiming to predict the time required for bug-fixing [1, 3, 11, 21] or the effort involved in solving issues [4, 16, 20].

Other works have focused on agile contexts, using machine learning to estimate story points [4, 12]. These studies have compared the performance of the classification task of different techniques such as Neural Networks [2, 4] as well as traditional machine learning algorithms such as Naive Bayes, Decision Trees, K-NN, and SVMs [12].

Porru et al. [12] studied the performance of a machine learning classifier to estimate story points, using attributes from issue reports. The authors extracted textual features from the description of the issue reports as well as their components and issue type fields. The main difference of our study consists in the introduction of developers' features to understand whether these features affect the story point prediction or not.

To analyze the performance of the different models, several performance measures have been proposed. Accuracy has been used to analyze the quality of predictive models [7] as well as the Mean Magnitude of Relative Error (MMRE) [2, 6, 12, 13]. However, several studies [9, 13] have criticized the use of MMRE because of its bias towards underestimation and its instability when comparing different models. For this reason, the use of Mean Absolute Error (MAE) [4, 12] and Standardized Accuracy (SA) [4] has recently been recommended. In our study, we use Accuracy, MAE, and SA to evaluate our predictive models.

4 RESEARCH QUESTIONS

In this study, we aim to answer the following research questions:

RQ1. Is the use of developer features suitable for estimating story points of issue reports?

RQ2. What is the Accuracy of predicting story points using developer features compared to using textual features?

RQ3. Does the use of developer features provide more accurate estimates in terms of MAE and SA than using textual features?

To answer **RQ1**, we conducted a sanity check that consists in comparing the performances of the prediction models with three baseline benchmarks commonly used in the context of effort estimation: Random Guessing, Mean Effort, and Median Effort [14, 15]. Random guessing chooses randomly a story point value from the set of possible values and uses it as the estimate of the target issue. Since Random guessing does not use any information associated with the target issue, it would be expected that any useful estimation model outperforms random guessing. The Mean and Median Effort approaches use the mean and median story points of the past issues as the estimate of the target issue. To answer **RQ2**, we trained predictive models using Support Vector Machines (SVMs) and different sets of features. SVM was selected since it has shown the best performance for predicting story points in comparison with other algorithms such as Decision Trees, k-NN, and Naive Bayes [12]. Then, we compared the Accuracy of the models. To answer **RQ3**, we followed a similar procedure than for RQ2. We first trained the predictive models using different sets of features. Then, we compared the performance of the effort estimation models calculating the MAE and SA.

5 EXPERIMENTAL SETUP

This section describes the steps performed in order to build the prediction models. The steps are based on the standard data mining workflow [7]. Firstly, we describe the dataset used. Secondly, we show the cleaning process that we apply to the dataset. Thirdly, we describe the sets of features used to train the predictive models. Fourthly, we train the models using Support Vector Machines (SVM) with different sets of features. These sets include developer and textual features, as well as a combination of both. Finally, we evaluate the results using different performance measures.

5.1 Dataset

The dataset consists of issue reports of eight open source projects. This dataset has been used in several studies [4, 12] and is publicly available. In addition, the dataset provides a wide range of possible scenarios in terms of project domain, number of issues, and developer experience. The open source projects included in the dataset are: Aptana Studio (APSTUD)[2], a web development IDE; Dnn Platform (DNN)[3], a web content management system; Apache Mesos (MESOS)[4], a cluster manager; Mule (MULE)[5], a lightweight Java-based enterprise service bus and integration platform; Sonatype's Nexus (NEXUS)[6], a repository manager for software artifacts required for development; Titanium SDK/CLI (TIMOB)[7], an SDK and a Node.js based command-line tool for managing, building, and deploying Titanium projects; Appcelerator Studio (TISTUD)[8], an Integrated Development Environment (IDE); and Spring XD (XD)[9], a unified, distributed, and extensible system for data ingestion, real-time analytics, batch processing, and data export. The total number of issues in the dataset is 15155, ranging from 886 (APSTUD) to 3691 (XD). Regarding the type of issue reports, they are mainly Bugs (6593) and User Stories (4062).

[1] Atlassian website – https://www.atlassian.com/software/jira

[2] Aptana Studio website – http://www.aptana.com/

[3] Dnn website – http://www.dnnsoftware.com/platform

[4] Apache Mesos website – http://mesos.apache.org/

[5] Mulesoft website – https://www.mulesoft.com/

[6] Nexus website – http://www.sonatype.org/nexus/

[7] Titanium website – https://jira.appcelerator.org/browse/TIMOB

[8] Appcelerator website –http://www.appcelerator.com/

[9] Spring XD website – http://projects.spring.io/spring-xd/

Table 1: Distribution of issue types and story points of the cleaned project dataset.

	Issue types						Story Point values				
	Story	Improvement	Bug	Task	Others	Total	Max	Mean	Median	Min	Std
APSTUD	37 (18.23%)	38 (18.72%)	110 (54.19%)	0 (0.0%)	18 (8.87%)	203	20	7.064	5	1	4.325
DNN	13 (5.42%)	48 (20.0%)	159 (66.25%)	7 (2.92%)	13 (5.42%)	240	8	2.117	2	1	1.137
MESOS	6 (1.63%)	96 (26.09%)	160 (43.48%)	81 (22.01%)	25 (6.79%)	368	13	2.546	2	1	1.712
MULE	7 (1.13%)	130 (21.04%)	313 (50.65%)	107 (17.31%)	61 (9.87%)	618	13	4.516	5	1	3.199
NEXUS	1 (0.3%)	56 (16.82%)	268 (80.48%)	8 (2.4%)	0 (0.0%)	333	3	0.964	1	0.5	0.571
TIMOB	59 (6.41%)	121 (13.15%)	657 (71.41%)	0 (0.0%)	83 (9.02%)	920	13	5.027	5	0.5	3.144
TISTUD	203 (17.31%)	198 (16.88%)	698 (59.51%)	0 (0.0%)	74 (6.31%)	1173	13	5.111	5	1	2.523
XD	164 (57.14%)	29 (10.1%)	86 (29.97%)	0 (0.0%)	8 (2.79%)	287	8	2.672	2	1	1.833
Total	490 (11.83%)	716 (17.29%)	2451 (59.17%)	203 (4.9%)	282 (6.81%)	4142	—	—	—	—	—

5.2 Cleaning Process

In real-world datasets, the data tend to be incomplete, noisy, and inconsistent [7]. Since our dataset consists of data extracted from real projects, we aim to correct their inconsistencies. To do that, we keep only the issue reports that meet the following conditions:

- The issue report has been assigned to a developer.
- The story points of the issue report have been assigned only once and never updated afterward. Those issues whose story points get updated are considered as unstable and might confuse the classifier [12].
- The issue has been addressed. An issue is addressed if its status is set to *closed* and its resolution field is set to *fixed*. Those issues that have not been addressed are likely to be unstable and they might confuse the classifier.
- The fields *summary* and *description* have been set and their values have not been changed after the initial set up.
- The story points have been assigned according to the Fibonacci sequence (i.e. values 0.5, 1, 2, 3, 5, 8, 13, 20, 40) and the number of instances with those values is greater than one.

The original dataset contained 15155 issue reports from eight projects. After applying the cleaning process, the dataset size decreased to 4142 issues. Table 1 describes the type of issues and the story points of the cleaned dataset.

5.3 Features

We computed features based on the original set of attributes in the dataset to help the classification task. These features are grouped into two sets: developer features and textual features.

5.3.1 Developer Features. This set of features aims to describe the individual characteristics of the developer.

- Reputation: Developer reputation has been used in several studies [8, 22] as a way to characterize the role played by developers in software projects. The reputation of a developer D is calculated as the ratio of the number of issue reports in the dataset that have been both opened and fixed by the developer to the number of issue reports opened by the developer plus one (Equation 1).

$$Reputation(D) = \frac{|opened(D) \cap fixed(D)|}{|opened(D) + 1|} \quad (1)$$

- Current developer workload: This feature is determined by the number of open issues that have been assigned to the developer at a time.
- Total work capacity (number of issues): The total number of issues that have been completed by the developer during the project.
- Total work capacity (story points): The total number of story points that have been completed by the developer during the project.
- Number of developer comments: The total number of comments that the developer has written in the project.

5.3.2 Text features. The textual features consist of several features extracted from the *summary* and *description* fields of the issues, written by the developers in natural language. The feature extraction procedure is the same as used by Porru et. al [12].

- Context: The *summary* and *description* fields were joined together in a new feature named *context*. Since developers often include not only a description of the issue in natural language but also code snippets to describe particular situations, the description in natural language and the code snippet are analyzed separately. The reason for this is that the language used in the blocks of code may have different meanings from those found in the natural language descriptions [12].
- Number of characters: Once we get the context of the issue, we calculate the number of characters used in both corpora (code and description).
- N-grams: We extracted features from both corpora using Term Frequency - Inverse Document Frequency (TF-IDF). Uni-grams and bi-grams are used as inputs to calculate TF-IDF.

5.4 Predictive Models

The research goal of this study is to compare different predictive models that produce a story point estimate for a given issue. Each model takes as input a combination of the features defined in Section 5.3: developer and textual features.

The models are built using Support Vector Machines (SVMs), since SVM have shown the best in result in the same domain [12]. When SVM is used for classification tasks, each training example is marked to belong to one of two categories. The SVM model assigns every new training example to one or the other category, so when mapped, the two categories are separated by a gap. Since we have more than two categories, many models must be used. To address this issue, we use the `scikit-learn`[10] python package that builds many different models with two categories and combines them into one model.

5.5 Validation and Evaluation

To validate the outcomes of the classifier, we use 10-fold cross-validation. Cross-validation is a well-known technique to prevent the classifier from over-fitting. In addition, we set the number of folds to ten since it is more computationally feasible than using a higher number of folds.

To evaluate the performance of the different models, we use three metrics: Accuracy, Mean Absolute Error (MAE), and Standardized Accuracy (SA). The Accuracy of a predictive model is simply defined as the ratio between the number of correct estimates and the total number of estimates. This measure has been used by many studies and it is particularly useful to make straightforward comparisons between the models and with related work.

Mean Absolute Error (MAE) has been recommended as a performance measure by recent research [4, 14, 17]. The MAE is defined by Equation 2, where y are the real story points assigned to an issue, $\hat{y}$ is the outcome given by an estimation model, and N the total number of issues in the dataset. Since this measure evaluates the error of the predictions, we can improve the performance of the model by decreasing the MAE.

$$MAE = \frac{1}{N}\sum_{i}^{N}|y_i - \hat{y}_i| \tag{2}$$

In addition, we evaluate the performance of the estimation models by using Standardized Accuracy (SA). The SA is based on the MAE and the MAE of Random Guessing. The SA is defined by Equation 3. Since this metric compares the predictions against the random approach, we can improve the performance of the model by increasing the SA.

$$SA = \left(1 - \frac{MAE}{MAE_{RandomGuess}}\right) * 100 \tag{3}$$

6 RESULTS

In this Section, we describe and discuss the results for each of the research questions.

RQ1. *Is the use of developer features suitable for estimating story points of issue reports* The results obtained from the prediction models are shown in Table 2. The columns show the values for the three performance measures used: Accuracy, MAE, and SA. The combinations of features are the following: a set of only developer features (Dev), a set of features only extracted from the text (Text), and a set consisting of both of the aforementioned sets (Text+Dev). In addition, Table 2 shows the results of using the three benchmark baselines: Mean, Median, and Random approaches. The analysis of the average values of the performance measures (last row of Table 2) suggests that the estimations obtained using only developer features are better than those achieved by using the standard baselines.

> Using only developer features is suitable for estimating story points since the model outperforms the baseline benchmarks.

RQ2. *What is the Accuracy of predicting story points using developer features compared to using textual features?* The results of the predictions models that use different sets of features are shown in Table 2. In particular, the first three data columns describe the accuracy of the models for all the projects in the dataset. The accuracy of the model using only developer features (Dev) outperforms the others in 5 of 8 projects. Averaging across all the project, their accuracy is also higher than the model that uses only textual features (Text) and a combination of both (Text+Dev).

> A predictive model based on SVM and developers' features can achieve an accuracy of 0.384 on average, outperforming the models based on textual features.

RQ3. *Does the use of developer features provide more accurate estimates in terms of MAE and SA than using textual features?* The values of MAE and SA for the predictive models using only the set of developer features (Dev), textual features (Text), and both sets combined (Text+Dev) are shown in Table 2. When comparing the MAE values, using only developer features (Dev) gets the lowest value on average across all the projects, although this value for the Dev set is the lowest in 3 out of 8 projects. Similarly, the same set of features (Dev) allows for obtaining the highest value of SA on average across all the projects, although the SA value for the Dev set is the highest in only in 3 out of 8 projects.

> Using a SVM classifier and only developer features to predict story points outperforms the values of MAE and SA achieved by using textual features or a combination of textual and developers features.

7 THREATS TO VALIDITY

There are threats to validity that should be carefully considered in this research. We tried to mitigate threats to construct validity by using a real-world dataset of issue reports from several open source projects that have been used in related studies [4, 12]. To deal with threats to conclusion validity, we compare the outcomes of the predictive models using performance measures that are commonly used according to the state of the art.

The used dataset was selected because it contains heterogeneous data about a wide range of projects of different sizes, complexities, teams of developers and communities. The characteristics of the dataset mitigate the threats to external validity. However, the dataset consists of open source projects, which is not representative for all kinds of software projects, in particular for projects conducted in commercial settings. Thus, further research is needed to improve external validity.

In addition, there might be several confounding factors which can influence the estimation process, such as company pressures, previous background, education or expertise of the developers.

[10] Scikit-learn website – http://scikit-learn.org/stable/

Table 2: Evaluation results.

	Accuracy					MAE						SA				
	Dev	Text	Text+Dev	Median	Random	Dev	Text	Text+Dev	Mean	Median	Random	Dev	Text	Text+Dev	Mean	Median
APSTUD	0.288	**0.297**	0.290	0.276	0.089	**3.074**	3.483	3.374	3.484	3.453	6.495	**52.673**	46.378	48.047	46.364	46.834
DNN	0.437	0.410	0.421	**0.438**	0.142	0.704	0.771	0.754	0.753	**0.700**	5.760	87.776	86.618	86.908	86.920	**87.848**
MESOS	0.338	**0.362**	0.359	0.255	0.130	1.375	1.234	1.215	1.254	**1.177**	4.823	71.493	74.423	74.817	74.006	**75.606**
MULE	**0.336**	0.256	0.306	0.241	0.120	**2.540**	2.979	2.814	2.600	2.594	5.551	**54.234**	46.334	49.308	53.159	53.272
NEXUS	**0.591**	0.553	0.581	0.462	0.120	0.495	0.526	0.511	0.373	**0.366**	5.350	90.738	90.177	90.457	93.020	**93.152**
TIMOB	**0.344**	0.283	0.305	0.325	0.116	**2.154**	2.672	2.550	2.278	2.264	5.759	**62.590**	53.605	55.719	60.445	60.683
TISTUD	**0.412**	0.350	0.344	**0.412**	0.139	1.753	2.035	2.064	1.800	**1.744**	5.262	66.691	61.328	60.778	65.792	**66.853**
XD	0.329	**0.356**	0.355	0.226	0.115	1.502	1.509	1.533	1.412	**1.334**	5.118	70.660	70.524	70.048	72.417	**73.928**
Average	**0.384**	0.358	0.370	0.329	0.121	**1.700**	1.901	1.852	1.744	1.704	5.515	**69.179**	65.531	66.421	68.371	69.100

8 CONCLUSIONS AND FUTURE WORK

We explored the use of developers' features to build predictive models that estimate story points in open source projects. We compared the performance of several models using different sets of features. The preliminary results show an improvement in Accuracy, MAE and SA of the predictive models that use developer features over the models that use features extracted from text.

Although the values of the three metrics are slightly better on average across all the projects, there is no improvement when we analyze the results of some projects individually. Using textual features have shown a better accuracy in 3 out of 8 projects whereas using the mean benchmark baseline gets better MAE and SA values in 5 out of 8 projects. Therefore, the first thing to further investigate is why these values are better in some projects than others.

When we analyze the results for each project, we can observe that these values can range from 0.366 to 3.453 for MAE and from 46.334 to 93.152 for SA. Further research is needed to understand these differences. Future work could assess the performance of the same models through a cross-project evaluation.

As for the features used in the models, this preliminary study shows that the model that uses developer features outperforms the one that uses text features. It suggests that predictions might be improved if new features related to the individual characteristics of the developers are taken into account.

ACKNOWLEDGMENTS

The authors would like to thank Annika Laumets-Tättar for their collaboration in this study. The work is supported by the institutional research grant IUT20-55 of the Estonian Research Council as well as the Estonian IT Center of Excellence (EXCITE).

REFERENCES

[1] W AbdelMoez, Mohamed Kholief, and Fayrouz M Elsalmy. 2013. Improving bug fix-time prediction model by filtering out outliers. In *Technological Advances in Electrical, Electronics and Computer Engineering (TAEECE), 2013 International Conference on*. IEEE, 359–364.

[2] Pekka Abrahamsson, Raimund Moser, Witold Pedrycz, Alberto Sillitti, and Giancarlo Succi. 2007. Effort prediction in iterative software development processes–Incremental versus global prediction models. In *Empirical Software Engineering and Measurement, 2007. ESEM 2007.* IEEE, 344–353.

[3] Saïd Assar, Markus Borg, and Dietmar Pfahl. 2016. Using text clustering to predict defect resolution time: a conceptual replication and an evaluation of prediction accuracy. *Empirical Software Engineering* 21, 4 (2016), 1437–1475.

[4] Morakot Choetkiertikul, Hoa Khanh Dam, Truyen Tran, Trang Thi Minh Pham, Aditya Ghose, and Tim Menzies. 2016. A deep learning model for estimating story points. *IEEE Transactions on Software Engineering* (2016).

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

[6] Tron Foss, Erik Stensrud, Barbara Kitchenham, and Ingunn Myrtveit. 2003. A Simulation Study of the Model Evaluation Criterion MMRE. *IEEE Trans. Softw. Eng.* 29, 11 (Nov. 2003), 985–995.

[7] Jiawei Han, Jian Pei, and Micheline Kamber. 2011. *Data mining: concepts and techniques.* Elsevier.

[8] Pieter Hooimeijer and Westley Weimer. 2007. Modeling bug report quality. In *Proceedings of the twenty-second IEEE/ACM international conference on Automated software engineering.* ACM, 34–43.

[9] Barbara A Kitchenham, Lesley M Pickard, Stephen G. MacDonell, and Martin J. Shepperd. 2001. What accuracy statistics really measure. *IEEE Proceedings-Software* 148, 3 (2001), 81–85.

[10] Viljan Mahnič and Toma Hovelja. 2012. On Using Planning Poker for Estimating User Stories. *J. Syst. Softw.* 85, 9 (Sept. 2012), 2086–2095.

[11] Dietmar Pfahl, Siim Karus, and Myroslava Stavnycha. 2016. Improving Expert Prediction of Issue Resolution Time. In *Proceedings of the 20th International Conference on Evaluation and Assessment in Software Engineering (EASE '16).* ACM, New York, NY, USA, Article 42, 6 pages.

[12] Simone Porru, Alessandro Murgia, Serge Demeyer, Michele Marchesi, and Roberto Tonelli. 2016. Estimating Story Points from Issue Reports. In *Proceedings of the The 12th International Conference on Predictive Models and Data Analytics in Software Engineering (PROMISE 2016).* ACM, New York, NY, USA, 2:1–2:10.

[13] Dan Port and Marcel Korte. 2008. Comparative Studies of the Model Evaluation Criterions MMRE and Pred in Software Cost Estimation Research. In *Proceedings of the Second ACM-IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM '08).* ACM, New York, NY, USA, 51–60.

[14] Federica Sarro, Alessio Petrozziello, and Mark Harman. 2016. Multi-objective Software Effort Estimation. In *Proceedings of the 38th International Conference on Software Engineering (ICSE '16).* ACM, New York, NY, USA, 619–630.

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

[16] Qinbao Song, Martin Shepperd, Michelle Cartwright, and Carolyn Mair. 2006. Software defect association mining and defect correction effort prediction. *IEEE Transactions on Software Engineering* 32, 2 (2006), 69–82.

[17] 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.

[18] 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 (PROMISE '14).* ACM, New York, NY, USA, 82–91.

[19] VersionOne. 2017. 11th Annual State of Agile Survey. (2017). https://explore.versionone.com/state-of-agile.

[20] Hui Zeng and David Rine. 2004. Estimation of software defects fix effort using neural networks. In *Computer Software and Applications Conference, 2004. COMPSAC 2004. Proceedings of the 28th Annual International*, Vol. 2. IEEE, 20–21.

[21] Hongyu Zhang, Liang Gong, and Steve Versteeg. 2013. Predicting bug-fixing time: an empirical study of commercial software projects. In *Proceedings of the 2013 international conference on software engineering.* IEEE Press, 1042–1051.

[22] Thomas Zimmermann, Nachiappan Nagappan, Philip J Guo, and Brendan Murphy. 2012. Characterizing and predicting which bugs get reopened. In *Proceedings of the 34th International Conference on Software Engineering.* IEEE Press, 1074–1083.