Code Reusability Estimation based on Static Analysis Metrics

Ioanna Apostolina, Konstantinos Samaras-Tsakiris
Electrical and Computer Engineering Dept., Aristotle University Of Thessaloniki
Aristotle University Of Thessaloniki
Thessaloniki, Greece
Email: {igaposto, kisamara}@ece.auth.gr

***Abstract*—The idea of reusing software components has been present in software engineering for several decades. This idea becomes more and more popular nowadays that open source software repositories are a common fact. So far, several tools have been developed to assess the software quality of these repositories which depends on fixed metric thresholds for defining the ground truth. In this work we present an organized approach that relates quality of software components with static analysis metrics and it estimates the reusability of software components in popular GitHub repositories. Our methodology includes two models: a one-class classifier, used to exclude all low quality code find inside classes of our 137 popular repositories and a neural network that computes a reusability score for each class of each repository. Evaluations shows(?)**

Keywords—source code quality, reusability, static analysis, user-perceived quality, neural network, one-class classification

I. Introduction

The numerous open source projects exist in online repositories such as GitHub have been used as a tool to reduce the development cost and time of the software. Nowadays, majority of software systems are being developed to a certain degree from an assembly of already existing reusable components. In order to assess these open source software components effectively, we should measure the *reusability* of these components which is closely related to the *quality* of the component.

Quality is a complex concept because it means different things to different people, it is highly context-dependent [2]. There was a common need for standardization of *quality* so international standard ISO/IEC 25010:2011 created. This international standard defines that the quality of a software component can be measured with eight quality characteristics to ensure *Functional Suitability*, *Usability*, *Performance Efficiency*, *Portability*, *Operability*, *Security*, *Compatibility*, and *Maintainability* [3]. However from a user's perspective quality is referred to the reusability to a software component and it is related with only four of the above characteristics: *Functional Suitability*, *Usability*, *Maintainability and Portability* [4].

Researchers during their attempt to measure software quality in an appropriate manner has led to the development of various metrics that we will discuss later. However most of them have a problem with determining the thresholds of these metrics. Defining a threshold for these metrics is not an easy task and it is usually performed by an expert [5]. This method is not flexible enough to describe in an effective way major software components. Nevertheless, a method that uses user-perceived quality as a measure of the reusability of a software component was introduced a year ago and had good evaluation results as we will describe in the next chapter [6].(?)Correction

In this paper, we appraise the code quality in terms of reusability of a software component through the use of static analysis metrics. We do the hypothesis that popularity from a user's point of view is a measure of reusability of a software component. In other words, we use the popularity of these projects as the ground truth. In the next section we will provide a proof for this hypothesis and examine the relationship between stars and forks for a sufficient dataset of repositories residing in GitHub. Based on this hypothesis , we design and create a system based on the four software quality characteristics: *Functional Suitability*, *Maintainability* , *Usability and Portability* with the simultaneous use of a set of static analysis metrics. Estimation of the reusability of a software component becomes possible. Our system consists of two main models: a one-class classifier trained with a support vector machine (SVM) that shows if a class component overcomes a quality threshold, and an artificial neural network (ANN) that estimates the reusability of the classes given the static analysis metrics.

The rest of this paper is organized as follows. (?)

II. Research Overview

Previous years various software metrics have been proposed from research for measuring the quality of software components [7, 8] but this problem remains not an easy one. This task requires an exact specification of software metrics and it is usually accomplished by an expert. Nevertheless, the help of an expert is not always available. For this reason, various methods have been proposed.

(?)Maybe later analysis

Estimating quality characteristics has been a popular research topic for many years. Many methods have been developed about software metrics thresholds definition. Nevertheless, most of them do not be effective when it comes to real world scenarios. A common practice is the adaptable quality estimation with the design of models [9]. One other practice is by analyzing the results of numerous software metrics that are computed. Ferreira et al. have proposed a method that uses the value of the computed metrics with probability distribution of them and indicate the bounds of metrics [10]. Another common approach is the design of quality evaluation systems which target to a single quality characteristic. Kumar proposes an SVM -based classifier and in this ways builds a reusability estimation system for software components[11]. One other method works efficient is this of Papamichail et al. which given static analysis metrics and using the popularity of software repositories create a system wich combines two models: a one- class classifier and an artificial neural network that estimates the quality of the code [6].

Although most of these proposals do not work properly in real-world scenarios as we referred earlier. Firstly, they are limited within certain quality thresholds; something that shows a limitation to the extent of being objective for all classes [9]. Automated system from the other side need the knowledge of an expert for training the model so we cannot use them every time that we need to evaluate the quality of a software component. Furthermore, these systems do not offer a single output measurement the user-perceived quality.

In this work, we build a system that estimates the code quality and provides a single quality metric based on a set of static analysis metrics, related to user-perceived quality. Given that popularity of components has a strong relationship with reuse [12], we consider that popularity means high quality of components. In other words, a repository residing in GitHub which is rated with many stars and forks possibly is a high quality project and reusable one. Based on this idea, we build a system that provides a reusability score and information about the values of metrics

III. CORRELATION BETWEEN STATIC ANALYSIS METRICS AND POPULARITY

In this section, we discuss the relationship between the stars and forks of a software repository in GitHub and form a dataset that will be used us our quality model. The dataset consists of 60 static analysis metrics computed for the most 137 GitHub repositories .

A. *Popularity and Reusability of Source Code (?)out*

---->τελικα δεν εχει σημασια προς το παρον ==σβήσιμο

B. *Scoring Classes*

IV. SYSTEM DESIGN

A.

B. *System Overview*

First, provide an overview of the system. This could be in the form of a figure. For the formatting of a figure see Fig. 1.

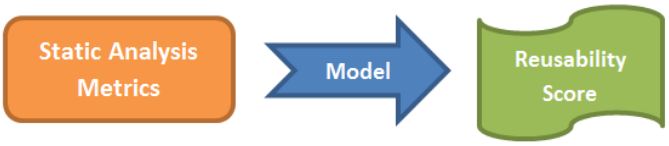

Fig. 1. Example of a figure caption.

Fig. 1 depicts …

C. *Data Preprocessing*

Write about the steps of data preprocessing.

D. *Model Construction*

Write about the models that you have created. You can add figures, tables, equations, references, etc.

If you want to add an equation, an example formatted equation is (1):

$$a + b = \gamma \tag{1}$$

Equation (1) concerns the …

E. ...

…

V. EVALUATION

A. *Evaluation Methodology*

Write what are the metrics that you have selected, what are the axes on which you assess your system, what are the experiments performed, etc.

B. *Evaluation Results*

Present the results of the experiments, possibly one subsection per experiment. Create tables, figures and any other supporting material. For an example of a table see Table 1.

TABLE I. TABLE STYLES

Table Head	**Table Column Head**		
	Table column subhead	***Subhead***	***Subhead***

Table Head	Table Column Head		
	Table column subhead	***Subhead***	***Subhead***
copy	More table copy		

Table 1 presents …

VI. CONCLUSIONS

Write here the findings of your analysis.

References

[1] IEEE Manuscript Templates for Conference Proceedings, online: http://www.ieee.org/conferences_events/conferences/publishing/templates.

[2] Kitchenham, Barbara, and Shari Lawrence Pfleeger. "Software quality: the elusive target [special issues section]." *IEEE software* 13.1 (1996): 12-21.

[3] "CISQ Code Quality Standarts." [Retrieved March 2017]. [Online]. Available: http://it-cisq.org/standards/

[4] Diamantopoulos, Themistoklis, Klearchos Thomopoulos, and Andreas Symeonidis. "QualBoa: reusability-aware recommendations of source code components." *Proceedings of the 13th International Conference on Mining Software Repositories*. ACM, 2016.

[5] Zhong, Shi, Taghi M. Khoshgoftaar, and Naeem Seliya. "Unsupervised Learning for Expert-Based Software Quality Estimation." *HASE*. 2004.

[6] M. Papamichail, T. Diamantopoulos and A. Symeonidis, "User-Perceived Source Code Quality Estimation Based on Static Analysis Metrics," *2016 IEEE International Conference on Software Quality, Reliability and Security (QRS)*, Vienna, 2016, pp. 100-107.

[7] Padhy, Neelamadhab, Suresh Satapathy, and R. P. Singh. "Utility of an Object Oriented Reusability Metrics and Estimation Complexity." *Indian Journal of Science and Technology* 8.1 (2017).

[8] Ferreira, Kecia AM, et al. "Identifying thresholds for object-oriented software metrics." *Journal of Systems and Software* 85.2 (2012): 244-257.

[9] Cai T, Lyu MR, Wong KF, Wong M. ComPARE: A generic quality assessment environment for component-based software systems. InProceedings of the 2001 *International Symposium on Information Systems and Engineering 2001*.

[10] Ferreira, Kecia AM, et al. "Identifying thresholds for object-oriented software metrics." *Journal of Systems and Software* 85.2 (2012): 244-257.

[11] Kumar, Ajay. "Measuring Software reusability using SVM based classifier approach." *International Journal of Information Technology and Knowledge Management* 5.1 (2012): 205-209.

[12] Borges, Hudson, Andre Hora, and Marco Tulio Valente. "Predicting the popularity of github repositories." *Proceedings of the The 12th International Conference on Predictive Models and Data Analytics in Software Engineering*. ACM, 2016.

[13]