Information and Software Technology 72 (2016) 110–124

Contents lists available at ScienceDirect

Information and Software Technology

journal homepage: www.elsevier.com/locate/infsof

Software metrics fluctuation: a property for assisting the metric selection process

Elvira-Maria Arvanitou[a], Apostolos Ampatzoglou[a,*], Alexander Chatzigeorgiou[b], Paris Avgeriou[a]

[a] Department of Mathematics and Computer Science, University of Groningen, Zernike Campus, Groningen, The Netherlands
[b] Department of Applied Informatics, University of Macedonia, Thessaloniki, Greece

ARTICLE INFO

Article history:
Received 24 May 2015
Revised 22 December 2015
Accepted 22 December 2015
Available online 30 December 2015

Keywords:
Object-oriented metrics
Fluctuation
Case study
Software evolution

ABSTRACT

Context: Software quality attributes are assessed by employing appropriate metrics. However, the choice of such metrics is not always obvious and is further complicated by the multitude of available metrics. To assist metrics selection, several properties have been proposed. However, although metrics are often used to assess successive software versions, there is no property that assesses their ability to capture structural changes along evolution.

Objective: We introduce a property, *Software Metric Fluctuation* (SMF), which quantifies the degree to which a metric score varies, due to changes occurring between successive system's versions. Regarding SMF, metrics can be characterized as *sensitive* (changes induce high variation on the metric score) or *stable* (changes induce low variation on the metric score).

Method: SMF property has been evaluated by: (a) a case study on 20 OSS projects to assess the ability of SMF to differently characterize different metrics, and (b) a case study on 10 software engineers to assess SMF's usefulness in the metric selection process.

Results: The results of the first case study suggest that different metrics that quantify the same quality attributes present differences in their fluctuation. We also provide evidence that an additional factor that is related to metrics' fluctuation is the function that is used for aggregating metric from the micro to the macro level. In addition, the outcome of the second case study suggested that SMF is capable of helping practitioners in metric selection, since: (a) different practitioners have different perception of metric fluctuation, and (b) this perception is less accurate than the systematic approach that SMF offers.

Conclusions: SMF is a useful metric property that can improve the accuracy of metrics selection. Based on SMF, we can differentiate metrics, based on their degree of fluctuation. Such results can provide input to researchers and practitioners in their metric selection processes.

1. Introduction

Software measurement is one of the most prevalent ways of monitoring software quality [21]. In practice, software quality measurement activities are governed by a measurement plan (e.g., developed based on the IEEE/ISO/IEC-15939 Std. [1]), which, among others focuses in defining the measurement goals, and the metrics selection process. According to Fenton and Pfleeger [16], building a measurement plan involves answering three main questions, two on defining the measurement goals, and a third one, on selecting appropriate metrics:

- *What to measure*? This question has two levels: (a) what quality attributes to measure?—this is related to the identification of the most important concerns of the stakeholders, and (b) what parts of the system should be assessed?—this is related to whether quality measurement should be performed on the complete system (*measure-in-large*) or on a specific design "hot-spot" (*measure-in-small*) [16].
- *When to measure*? This question has two levels as well. The first level concerns the measurement frequency, where one can choose between two major options: (i) perform the measurement tasks once during the software lifecycle (*measure once*), or (ii) perform measurement tasks many times during the software lifecycle (*measure repeatedly*) [16]. The second level concerns the development phase(s) when measurement is to be performed. This decision sets some additional constraints to the metric selection process, in the sense that if one selects to

* Corresponding author. Tel.: +30 2310611090.
E-mail addresses: e.m.arvanitou@rug.nl (E.-M. Arvanitou), apostolos.ampatzoglou@gmail.com, a.ampatzoglou@rug.nl (A. Ampatzoglou), achat@uom.gr (A. Chatzigeorgiou), paris@cs.rug.nl (P. Avgeriou).

http://dx.doi.org/10.1016/j.infsof.2015.12.010
0950-5849/

perform measurement activities in an early development phase the available metric suites are different from those that are available at the implementation level. Usually, design-level metrics are less accurate than code-level metrics; however, they are considered equally important, because they provide early indications on parts of the system that are not well-structured. A detailed discussion on high-level metrics (design-level) and low-level metrics (code-level), can be found in [3].

- *How to measure*? While answering this question, one should select the most fitting measure from the vast collection of available software quality metrics.

All aforementioned questions are inter-dependent, so the order of answering them varies; for example one could start from metric selection (i.e. 'how') and then move on to answer 'when' and 'what' (i.e., measurement goal), or the other way around. When answering one of the questions, the available options for answering the subsequent questions are getting more limited (an elaborate discussion on the inter-connection among the answers to these questions is presented in Section 7). For example, if someone selects to measure cohesion at the design phase, the set of available metrics is limited to the high-level cohesion metrics, as discussed by Al Dallal and Briand [3]; whereas if one selects to measure cohesion at implementation level, the set of available metrics is broadened to the union of high- and low-level cohesion metrics [3]. Due to the high number of available metrics, the selection of metrics that quantify the target quality attributes is far from trivial. For example, concerning cohesion and coupling, recent studies describe more than 20 metrics for each one of them [3] and [19]. This selection process becomes even more complex by the option to choose among multiple *aggregation functions*. Such functions are used to aggregate metric scores from the micro-level of individual artifacts (e.g. classes), to the macro-level of entire systems [13] and [33], whose industrial relevance is discussed in detail by Mordal et al. [29]. In order to assist this metric selection process, researchers and practitioners have proposed several metric properties that can be used for metrics validation and characterization [1,9] and [10].

Metrics selection becomes very interesting in the context of software evolution. Along evolution metric scores change over time reflecting the changes of different characteristics of the underlying systems. For example a metric that concerns coupling, changes from one version of the system to the other, reflecting the changes in the dependencies among its classes. Therefore, a quality assurance team needs to decide on the accuracy with which they wish to capture small-scale changes from one version of the system to the other. This decision is influenced by the goals of the measurement (i.e., the answers to the first two aforementioned questions—"*what and when to measure?*"). In particular, both available options (i.e., capture small changes or neglect them) may be relevant in different contexts. For example, when trying to assess the overall software architecture, the quality team might not be interested in changes that are limited inside a specific component; on the contrary, when trying to assess the effect of applying a source code refactoring, e.g., extract a superclass [18], which is a local change, a small fluctuation of the metric score should be captured. Thus, a property that characterizes a metric's ability to capture such fluctuations would be useful in the metric selection processes. Nevertheless, to the best of our knowledge, there is no such property in the current state of the art for research or practice.

Therefore, in this paper, we define a new metric property, namely ***Software Metrics Fluctuation (SMF)***, as *the degree to which a metric score changes from one version of the system to the other* (for more details see Section 3). While assessing a metric with respect to its fluctuation, it can be characterized as **stable** (*low fluctuation*: the metric changes insignificantly over successive versions) or as **sensitive** (*high fluctuation*: the metric changes substantially over successive versions). Of course, the property is not binary but continuous: there is a wide range between metrics that are highly stable and those that are highly sensitive. Although the observed metric fluctuations depend strongly on the underlying changes in a system, the metrics calculation process also plays a significant role for the assessment of fluctuation. For example, "*What structural characteristics does it measure?*", "*How frequently/easily do these characteristics change?*" or "*What is the value range for a metric?*". In order for the fluctuation property to be useful in practice it should be able to distinguish between different metrics that quantify the same quality attribute (e.g., cohesion, coupling, complexity, etc.). This would support the metric selection process by guiding practitioners to select a metric that is either stable or sensitive according to their needs for a particular quality attribute. Additionally, several metrics work at the micro-level (e.g., method- or class-level), whereas practitioners might be interested in working at a different level (e.g., component- or system-level). The most frequent way of aggregating metrics from the micro- to the macro-level is the use of an aggregation function (e.g., average, maximum, sum, etc.). Therefore, we need to investigate if SMF is able to distinguish between different aggregation functions when used for the same metric. Such an ability would enable SMF to provide guidance to practitioners for choosing the appropriate combination of metric and aggregation function.

In this paper we empirically validate SMF by assessing: (a) the fluctuation of 19 existing object-oriented (OO) metrics, through a case study on open-source software (OSS) projects—see Section 5, and (b) its usefulness by conducting a second case study with 10 software engineers as subject—see Section 6. The *contribution* of the paper is comprised of both the ***introduction and validation of SMF as a property*** and the ***empirical evidence*** derived from both case studies. The organization of the rest of the paper is as follows: *Section 2* presents related work and Section 3 presents background information on the object-oriented metrics that are used in the case studies; *Section 4* discusses fluctuation and introduces the definition of a software fluctuation metric; *Section 5* describes the design and results of the case study performed so as to assess the fluctuation of different object-oriented metrics; *Section 6* presents the design and outcome of the case study conducted for empirically validating the usefulness of SMF; Section 7 discusses the main findings of this paper; *Section 8* presents potential threats to the validity; and *Section 9* concludes the paper.

2. Related work

Since the proposed Software Metrics Fluctuation property allows the evaluation of existing metrics, past research efforts related to desired metric properties will be presented in this section. Moreover, since the proposed property is of interest when someone aims at performing software evolution analysis, other metrics that have been used in order to quantify aspects of software evolution will be described as well.

2.1. Metric properties

According to Briand et al. [9,10] metrics should conform to various theoretical/mathematical properties. Specifically, Briand et al., have proposed several properties for cohesion and coupling metrics [9,10], namely: *Normalization* and *Non-Negativity, Null Value* and *Maximum Value, Monotonicity,* and *Merging of Unconnected Classes* [9,10]. The aforementioned metric properties are widely used in the literature to mathematically validate existing metrics of these categories (e.g., by Al Dallal et al. [2]). Additionally, in a similar context, IEEE introduced six criteria that can be used for assessing the validity of a metric in an empirical manner.