Information and Software Technology 75 (2016) 90–104

Contents lists available at ScienceDirect

Information and Software Technology

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

An empirical investigation into the effect of slice types on slice-based cohesion metrics

Yibiao Yang, Yangyang Zhao, Changsong Liu, Hongmin Lu, Yuming Zhou*, Baowen Xu

State Key Laboratory for Novel Software Technology, Nanjing University, China

ARTICLE INFO

Article history:
Received 2 June 2015
Revised 28 February 2016
Accepted 1 April 2016
Available online 12 April 2016

Keywords:
Cohesion
End slice
Metric slice
Metrics

ABSTRACT

Context: There is a debate about whether end slice or metric slice is preferable for computing slice-based cohesion metrics. However, up till now, there is no consensus about this issue.
Objective: We aim to investigate the relationship between end-slice-based and metric-slice-based cohesion metrics and then determine which type of slice is preferable for computing slice-based cohesion metrics.
Method: We used forty widely used open-source software systems to conduct the study. First, we compute the baseline values for end-slice-based and metric-slice-based cohesion metrics. Then, we investigate their relationships with module size. Finally, we employ correlation analysis and principal component analysis to analyze the relationships between end-slice-based and metric-slice-based cohesion metrics.
Results: End-slice-based and metric-slice-based cohesion metrics have similar baseline metric values. Furthermore, they exhibit a similar negative correlation with module size. In particular, the results from correlation analysis and principal component analysis reveal that they essentially measure the same cohesion dimensions.
Conclusion: From the viewpoint of metric values, there is little difference between end-slice-based and metric-slice-based cohesion metrics. We hence recommend choosing end slice for computing slice-based cohesion metrics in practice, as extra cost involved in data collection could be avoided.

1. Introduction

Cohesion is an important software quality attribute which refers to the tightness of elements within a module [1,2]. Highly cohesive modules are desirable in a system as they are easier to develop, maintain, and reuse, and hence are less fault-prone [1,2]. In the last three decades, researchers developed many slice-based cohesion metrics to quantify the cohesion of a module at the function level of granularity [3–10]. In [11], Meyers and Binkley found that slice-based cohesion metrics could be used to quantify the deterioration that accompanies software evolution. More recently, our study shows that slice-based cohesion metrics are not redundant with respect to the most commonly used code/process metrics and are of practically important value in the context of fault-proneness prediction [12]. Therefore, slice-based cohesion metrics can be used as an important quality indicator for practitioners. In other words, practitioners could use these metrics to identify potentially faulty modules for quality enhancement.

For a given function, the computation of a slice-based cohesion metric consists of the following three steps. The first step is to identify the output variables of the function, including function return values, modified global variables, printed variables, and modified reference parameters [13]. The second step is to employ program slicing techniques to obtain one slice with respect to each output variable of the function [9,14]. The third step is to use the resulting slices to compute the slice-based cohesion metric. Of these three steps, the second step is the most crucial, as it is the basis for the computation of slice-based cohesion metrics. In the literature, there is a debate about whether end slice or metric slice should be used for computing slice-based cohesion metrics [11,12,15]. An end slice with respect to variable v is the *backward slice* with respect to v at the end point of the module [9,16]. A metric slice with respect to variable v is the union of the *backward slice* with respect to v at the end point of the module and the *forward slice* computed from the top of the backward slice [7]. Consequently, a metric slice is more time-consuming to compute compared with an end slice. In most previous studies, end slice was used to compute slice-based cohesion metrics [5–9,11,15–18]. However, Ott and her colleagues argued that metric slice can result in more accurate cohesion metrics [10]. Up till now, there is

* Corresponding author. Tel.: +86 25 89682450.
E-mail address: cs.zhou.yuming@gmail.com (Y. Zhou).

http://dx.doi.org/10.1016/j.infsof.2016.04.001
0950-5849/© 2016 Elsevier B.V. All rights reserved.

no consensus about which type of slice is preferable for slice-based cohesion metrics.

In this study, we aim to attack this issue. To this end, we conduct a comparative study to investigate the relationship between end-slice-based and metric-slice-based cohesion metrics. The subject systems in our study consist of forty open-source software systems. We use a source code analysis tool called Frama-C to collect slice-based cohesion metrics. Based on the data collected from these forty systems, we first compute the baseline metric values for end-slice-based and metric-slice-based cohesion metrics. As stated by Meyers and Binkley [11], baseline values for slice-based metrics "are useful for identifying degraded modules" and can "aid in the transfer of technology from academia to industry". Similar to [11], we use the average metric values and the 95% confidence interval to compute the baseline metric values. Then, we investigate their relationships with module size. This will help us determine "whether slice-based cohesion metrics are nothing more than a proxy for module size" [11]. If they are, it will mean that they do not provide new information. Consequently, there would be no need to use them in practice, especially considering the relatively high computation cost incurred. Finally, we employ correlation analysis and principal component analysis (PCA) to analyze the relationships between end-slice-based cohesion metrics and metric-slice-based cohesion metrics. The purpose of this aims to determine whether end-slice-based and metric-slice-based cohesion metrics indeed measure the same cohesion dimensions, although different types of slices are used. The experimental results show that end-slice-based and metric-slice-based cohesion metrics have similar baseline metric values. Furthermore, they exhibit a similar negative correlation with module size. In other words, modules with larger size tend to be less cohesive. In particular, the results from correlation analysis and PCA reveal that end-slice-based and metric-slice-based cohesion metrics essentially measure the same software cohesion dimensions. Therefore, from the viewpoint of metric values, there is little difference between end-slice-based and metric-slice-based cohesion metrics. We hence recommend choosing end slice rather than metric slice, as extra cost evolved in data collection could be avoided.

The rest of this paper is organized as follows. Section 2 introduces slice-based cohesion metrics that we will investigate. Section 3 introduces the experimental methodology used for this study, including the data sets and the data analysis method. Section 4 reports in detail the experimental results. Section 5 discusses the findings. Section 6 presents the related work. Section 7 examines the threats to validity of our study. Section 8 concludes the paper and outlines directions for future work.

2. Slice-based cohesion metrics

In this section, we first introduce the concept of end slice and metric slice. Then, we describe these slice-based cohesion metrics that will be investigated in this study. Finally, we use an example function to illustrate the computations of end-slice-based and metric-slice-based cohesion metrics.

2.1. End slice and metric slice

As aforementioned, for a given module, the most crucial step for the computations of slice-based cohesion metrics is to obtain either the end slice or the metric slice for each output variable of a module. An end slice with respect to variable v is the *backward slice* with respect to v at the end point of the module [9,16]. A metric slice with respect to variable v is the union of the *backward slice* with respect to v at the end point of the module and the *forward slice* computed from the top of the backward slice [7]. More specifically, a *backward slice* of a module at statement n with respect to variable v is the sequence of all statements and predicates that might affect the value of v at n. A *forward slice* of a module at statement n with respect to variable v is the sequence of all statements and predicates that might be affected by the value of v at n. It is easy to know that end slice only takes into account the *uses* data relationship. However, metric slice also takes into account the *used by* data relationship [3].

We next use an example function *fun* shown in Table 1, which aims to determine the smallest, the largest, and the range of an array, to illustrate the concepts of end slice and metric slice. In Table 1, the first column lists the statement number (excluding non-executable statements such as blank statements, "{", and "}"). The second column lists the code of the example function. As can be observed, the output variables of function *fun* are *smallest*, *largest*, and *range*. The former two variables are the modified reference parameters and the latter one is the function return value. The third to fifth columns list the end slice for each output variable. The sixth to eighth columns list the forward slice computed from the top of the end slice with respect to each output variable. The ninth to eleventh columns list the metric slice for each output variable. Here, a vertical bar "|" in the last nine columns denotes that the indicated statement is part of the corresponding slice for the named output variable. As can be seen, statement is the basic unit for these slices. In other words, end slice and metric slice shown in Table 1 are indeed *statement-level end slice* and *statement-level metric slice*. We next use the output variable *largest* as an example to explain the computation of end slice and metric slice. In Table 1, for the output variable *largest* of the function *fun*, we can find that: 1) the 7th, 9th, 10th, 11th, 14th, 15th, and 16th statements belong to its end slice since these statements will have an direct or indirect impact on the final value of *largest* at the end of the function *fun*; 2) the 14th, 15th, 17th, and 18th statements belong to its forward slice of the variable largest since these four statements are directly or indirectly impacted by the first definition statement of variable *largest* (i.e. the 10th statement "$*$largest $= *$smallest;").

In order to obtain the dependencies at a finer granularity, researchers further propose the concepts of data-token-level end slice and metric slice, in which data token is the basic unit [3]. As shown in the second column in Table 1, one statement might consist of a number of data tokens (i.e. the definitions of and references to variables and constants). For example, the ninth statement "$*$smallest $=$ A[0];" consists of the following three data tokens: "smallest", "A", and "0″. In this sense, data token is a finer granularity than statement. Table 2 shows the data-token-level end slice and metric slice for each output variable of the function fun. In this table, T_i in the second column indicates the i-th data token for T in the function, "1″ and "0″ in the last six columns respectively denotes that the indicated data-token is or is not part of the corresponding slice for the named output variable.

2.2. Slice-based cohesion metrics

After obtaining the end slices or metric slices for individual output variables of a given function, we can use them to calculate slice-based cohesion metrics. In this study, a function is regarded as a module and the output variables of a function consist of the function return value, modified global variables, modified reference parameters, and standard outputs by the function. In the last three decades, researchers proposed the following ten slice-based cohesion metrics: *Coverage, MaxCoverage, MinCoverage, Tightness, Overlap, SFC, WFC, A, NHD*, and *SBFC* [3–10].

Of these ten metrics, *Coverage, Tightness*, and *Overlap* can be originally traced back to Weiser's work [16]. For a given module, Weiser first sliced on every output statement occurred in the mod-