Information and Software Technology 75 (2016) 56–70

Contents lists available at ScienceDirect

Information and Software Technology

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

The effect of automatic concern mapping strategies on conceptual cohesion measurement

Bruno Silva [a,*], Claudio Sant'Anna [b], Neylor Rocha [b], Christina Chavez [b]

[a] Universidade Salvador - UNIFACS Salvador, Bahia, Brazil
[b] Department of Computer Science, Federal University of Bahia Salvador, Bahia, Brazil

ARTICLE INFO

Article history:
Received 19 June 2015
Revised 20 March 2016
Accepted 23 March 2016
Available online 1 April 2016

Keywords:
Module cohesion
Cohesion metrics
Concern mapping
Comparative empirical study

ABSTRACT

Context: Cohesion has been recognized as an important quality attribute of software design across decades. It can be defined as the degree to which a module is focused on a single concern of software. A concern is any concept, feature, requirement or property of the problem or solution domain. Conceptual cohesion is an alternative way of cohesion measurement based on what concerns each module addresses. Therefore, adopting a strategy to map concerns to source code elements is challenging but necessary.

Objective: We aim at providing empirical evidence about whether automatic concern mapping strategies are already ready to be used effectively for conceptual cohesion measurement.

Method: We carried out an empirical study to assess the ability of conceptual cohesion measurement using different automatic concern mapping strategies in selecting the least cohesive modules.

Results: Conceptual cohesion measurements over the two analyzed mapping strategies performed weakly in the ability of selecting the least cohesive modules. We then provide a discussion to explain the reasons.

Conclusion: Concern mapping strategies should be carefully chosen for conceptual cohesion measurement, specially if automatic mapping is under consideration. Manual mapping is still the most reliable way for computing conceptual cohesion. We pointed out limitations in automatic mapping strategies that go beyond conceptual cohesion measurement purposes and which should be considered in future research or applications in industry.

1. Introduction

Cohesion is one of the key properties of software design. Amongst several definitions, cohesion can be defined as the degree to which a module[1] is focused on a single concern of software [1–3]. A concern is any concept, feature, requirement or property of the problem or solution domain of software [4].

However, measuring cohesion is not straightforward as it is difficult to capture what concerns a module realizes. Indeed, several researchers have attempted to provide an objective and effective way to measure cohesion [5]. Most of them rely on structural information extracted from the source code. For example, the most traditional and well-known cohesion metrics quantify cohesion by counting pairs of methods of a class that access the same attributes [6,7]. This notion of cohesion is dependent on the source code structure and does not consider any abstract information regarding the concerns implemented by the classes. We classify this group of metrics as *Structural Cohesion*.

In contrast, there is an alternative group of recently proposed cohesion metrics which attempts to measure module cohesion by considering the concerns each module addresses [3,8,9]. For example, the MWE (Maximal Weighted Entropy) metric [8] executes a text mining method to identify concerns in source code and then computes cohesion based on how concerns are distributed over each software module. In this direction, we have been investigating the LCbC (Lack of Concern-based Cohesion) metric [9–11], which measures cohesion by counting the number of concerns a module addresses. MWE and LCbC are classified in the group of *Conceptual Cohesion* metrics.

* Corresponding author. Tel.: +5571991669687.
E-mail addresses: bruno.carreiro@pro.unifacs.br (B. Silva), santanna@dcc.ufba.br (C. Sant'Anna), neylorbr@dcc.ufba.br (N. Rocha), flach@dcc.ufba.br (C. Chavez).

[1] A module can be a class, an interface, or whatever abstraction representing a module as a unit of implementation.

http://dx.doi.org/10.1016/j.infsof.2016.03.006
0950-5849/© 2016 Published by Elsevier B.V.

In theory, conceptual cohesion metrics represent better the real notion of cohesion than structural cohesion metrics as they are supposed to measure this property by considering the abstract information of concerns besides the source code structure. In fact, researchers have found evidence in this direction and recent results have demonstrated the potential of conceptual cohesion metrics to be further investigated and used in practice. For instance, in a previous study we found that conceptual cohesion is closer to developers' reasoning even for unexperienced ones [11]. Also, researchers have demonstrated an association between conceptual cohesion and external quality attributes such as bug-proneness [8,12] and change-proneness [9].

However, although researchers have pointed out advantages of conceptual cohesion compared to structural cohesion, it is more difficult to measure conceptual cohesion because it relies on the so-called *Concern Mapping* [4]. A concern mapping is the assignment of software concerns to source code elements (e.g., attributes, methods, classes). In Section 3, we give an overview of concern mapping techniques summarized in five categories. Their basic intent is to heuristically provide information about what concerns each module addresses in a software system. They also vary on the level of automation (manual, semi or fully automatic). In general, manual concern mapping is more accurate as developers can apply their multiple knowledge, experience and may also consider auxiliary artifacts besides source code for supporting them. Several works have used manual mappings in their datasets [13–15]. However, manual mapping is too time-consuming, thus becoming difficult for developers to cost-effectively build and keep it consistent over time.

Therefore, the high cost of manual concern mapping may hinder the adoption of conceptual cohesion in practice. Some concern mapping strategies, such as the one used for MWE metric computation, employ a fully automatic heuristic. However, current automatic strategies for concern mapping have limitations which may cause failures in identifying concerns. Ultimately, such failures may affect, to some unknown extent, conceptual cohesion measurement [9].

In this context, the main goal of this paper is to investigate the effect of applying automatic concern mapping strategies for conceptual cohesion measurement. We aim at providing scientific evidence that could help: (i) researchers on further works in the field of software measurement; (ii) tool builders looking for incorporating innovative ways of cohesion measurement in their products; and (iii) practitioners in applying conceptual cohesion in their software measurement tasks. Therefore, our overarching research question is: *Are automatic concern mapping strategies already ready to be used effectively for conceptual cohesion measurement in place of manual mappings?*

We carried out an empirical study where we investigated the ability of two different conceptual cohesion measurements, using two automatic mapping strategies, in selecting the least cohesive modules. The manual mapping strategy is our baseline for generating a reference conceptual cohesion measurement. In summary, we found that the two automatic strategies we applied have limitations yet, so they negatively affect an effective use of conceptual cohesion measurement. Also, we discussed possible reasons we observed from this finding and how they could be exploited in further works for researchers and practitioners.

The remainder of the paper is organized as follows: Section 2 describes conceptual cohesion metrics and explains our metric choice for this study; Section 3 presents the concept of concern mapping and an overview of available techniques; In Section 4 we explain the study scope and corresponding methodology; Section 5 details the study settings; Section 6 presents results and findings, whereas Section 7 discusses their implications; Section 8 explains threats to validity; Section 9 describes related work; and Section 10 presents the conclusion and future work.

```
public class RelationSpouse extends Relation {
    public Person husband;
    public Person wife;

    public RelationSpouse (Person husband, Person wife){

    public String getRelationType(Person person){
        if (person.equals(husband))
            return WIFE;
        else if (person.equals(wife))
            return HUSBAND;
        else{
            JOptionPane.showMessageDialog(null,
                    "Wrong person type for spouse relationship...");
            return INVALID_RELATIONSHIP;
        }
    }
    public Person getPartner(Person person){
        if (person.equals(husband))
            return wife;
        else if (person.equals(wife))
            return husband;
        else{
            JOptionPane.showMessageDialog(null,
                    "Wrong person type for spouse relationship...");
            return null;
        }
    }

    public Person getHusband() {
    public Person getWife() {
    public Person getPerson1(){
    public Person getPerson2(){
    public boolean equals(Object obj){
    public void printToFile(File f)throws IOException{
}
```

Fig. 1. A sample class and its concerns mapped. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

2. Conceptual Cohesion Measurement

C3 (Conceptual Cohesion of Classes) [12], MWE (Maximal Weighted Entropy) [8] and LCbC (Lack of Concern-based Cohesion) [9] are the most recent conceptual cohesion metrics available in literature. On the one hand, C3 and MWE directly depend upon text mining techniques for identifying concerns in source code and then computing cohesion. On the other hand, for the same purpose, LCbC is not dependent upon a specific technique. LCbC quantifies cohesion of a given module in terms of the number of concerns addressed by it. It just counts the number of concerns identified in each module. Therefore, it is possible to compute LCbC by having any concern mapping strategy applied beforehand.

As C3 and MWE strongly depend upon a specific concern mapping technique, this study uses LCbC as the representative metric for conceptual cohesion. The study purpose is to assess conceptual cohesion measurement on top of different concern mapping strategies, which is only possible with LCbC.

To exemplify LCbC measurement, Fig. 1 illustrates a Java class with two attributes and nine methods, which is part of a system that builds and represents family relationships as trees. Particularly, this class represents the spouse relationship between two people: the *husband* and *wife* attributes. The methods involve: getting access to private attributes, getting the relation type, getting the partner object, objects comparison, and exporting the relationship to a given file. The light pink background, predominant in the figure, highlights the class main concern, which is what the class is supposed to implement (i.e., representing spouse relationship). The blue background highlights the code fragments calling a window box to output a message in the user interface, and the green background highlights a method to export the relationship represented by the class to a given file. Those additional two concerns