Federal University of Bahia
Computer Science Department
Post Graduate Program in Computer Science

CASE STUDY PROTOCOL

"Developers' perception about code anomalies identified by threshold values suggested by different strategies."

Luan Passos
Rebeca Sousa
Thamiris Gomes

SALVADOR – BA
July, 2017

1. Background

1.1 Systematic Review

The erosion caused in a software project is inevitable because of how the software is developed. However, good development methods help to increase the system's longevity. On the other hand, it is important to use mechanisms which help to identify anomalies in the code that need to be revised, so there is no degradation in the system [van Gurp & Bosch 2002].

The strategies based on software metrics and threshold values are frequently used on the detection of these anomalies in the code, although the metrics and definition of threshold values can indicate false positives and false negatives, when there it does not exist precision on the used strategies. [Macia, 2013] [Marinescu, 2004]

One of the biggest difficulties for precision on the software metric results is the definition of threshold values. With that in mind, there are different strategies to calculate these values [Aniche 2016; Dósea & Sant'Anna 2016]. So, the study aims to present accuracy on existent strategies, which are: Assign generic threshold values to classes considering only the application's domain; consider the architectural roles of classes to indicate threshold value; and, indicate different threshold values for the classes, considering the design's role.

1.2 Motivations

- Difficulties in maintaining , understanding and evolutioning the software;
- Difficulties in defining threshold values;
- The use of generic thresholds generates many positive and negative falses;
- Understand what the developers think about techniques and tools for detecting anomalies in the available code.

1.3 Contributions

The evaluation approach of the ContextSmell tool, that recommends codes with anomalies in which applies different strategies indicated by threshold values, provides to the researcher the accuracy of different strategies to indicate the threshold values to the classes, based on metrics of software quality.

The extracted information by the tool indicates to the developer candidates to poorly written methods according to long method's metrics, cyclic complexity, attachment and method by parameter. Thereby, it enables that the developer team have more attention and do code revising of these methods, to avoid future problems in the software development, such as, understanding, maintenance, and inclusion of new features.

1.4 Research Questions

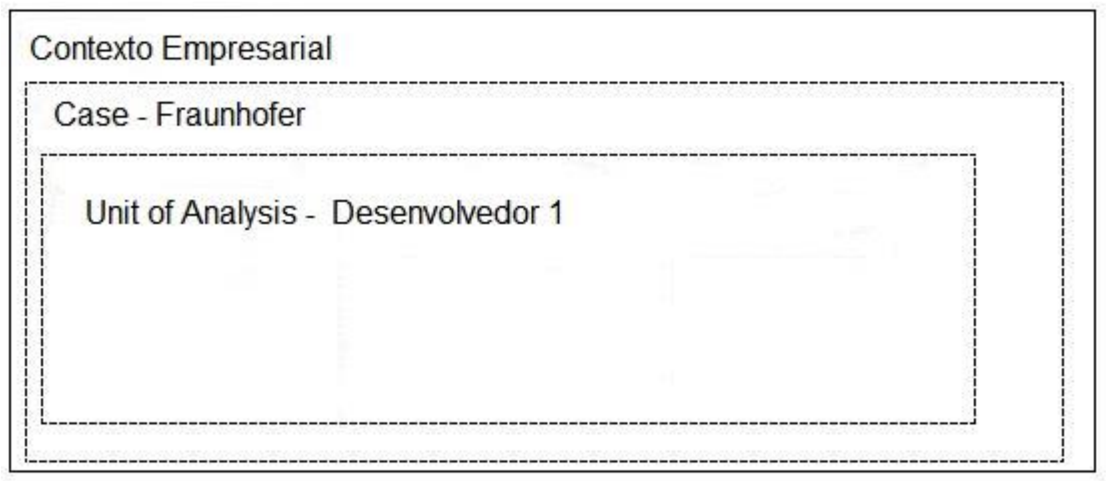
RQ1: Are there any differences in the accuracy of the techniques on the calculus of threshold values for the metrics in the metrics level?

RQ2: Which design decisions influenced the developers to point methods as complex?

2. Design

The goal of the study is to evaluate the accuracy of a recommendation tool of poorly written methods, indicated by threshold values that are involved by different strategies, from the developers' perspectives. The study also will investigate the developers' opinions about the used techniques to extract the methods agreed as problematic, that could cause future problems to the system evolution.

To do so, we will make a case study in which we will apply the tool in a development environment of a company, that will provide a system and expert developers in the chosen project. With the result from the tool and the developers perspectives about those results, we have the scope as in the figure below.



3. Case Selection

The main criterion of selection of the case study was a company with systems developed in the JAVA language and professionals with experience in the chosen system to contribute with the results of the study. The unit of analysis will be Fraunhofer, located in the technology park of Bahia, where it works with the development of innovative software solutions for the market.

In the first step we will define the system that will be applied to the tool under study, known to the developers involved. The results will be displayed and presented to at least three developers, where they will respond to questionnaires applied as a case study procedure.

Company	
Type of company	Private
Number of professionals	6
Number of professionals in the project	2
Type of Software	Management distribution system of organs and tissues
Type of domain	WEB System
Programming language	JAVA

4. Procedures and functions of study case

To reach the proposed goal, the team will select one or a set of implemented applications in the programming language Java, that can be evaluated by specialists who have experience on the development of the selected system, and with the selected application, it will be applied the detection techniques of threshold values.

With these threshold values, all “anomalous methods” identified by the different used techniques will be selected. These anomalous methods will be presented to the specialists that will select the methods that they judge to be poorly written. From the judgments of these specialists, through a questionnaire, we will identify precision and recall of each technique that we used to identify the threshold values. And also obtain the opinion from developers about the best strategy to indicate threshold values that gives a more reliable return in the list of poorly written methods.

The research team is composed by Thamiris Gomes and Rebeca Sousa. The researchers are going to apply the tool on the chosen system, gather the list of returned methods and plan questions for the questionnaire that is going to be applied to the specialists who work in the project.

5. Data Collection

5.1. Data to be collected

The first data collection will be extracted from the chosen system, when applying the anomalous code recommendation tool. The tool will make a checkout on the repository in which the system is stored, extract system design information, calculate the threshold values from different strategies and detect the defective methods. With that, the tool will present a list for poorly written methods, extract from all detection strategies used.

The second collection will be a questionnaire divided into two stages. At the first stage we are going to investigate if the listed methods by the tool are poorly written and propitious to future problems, ether from the system evolution, understanding, among others quality software attributes. And at the second stage from the questionnaire will be investigated the developers' opinions in reference to which strategies to calculate threshold values for the classes is the best option.

5.2. Procedures of data storage

We will make available all documentation related to the design of the study; Data collected and shared consent by the units of analysis; Artifacts generated by the development and refinement of the questionnaires; E-mail templates and consent letters in a GitHub, where the tool will also be made available.

6. Data Analyses

The results returned by the tool will be recommended methods as "poorly written" and will be presented to developers in two different perspectives.

In the first, through a questionnaire, the accuracy and coverage values of the tool will be analyzed and its results presented quantitatively. For this, we will inform the name of each method recommended by the tool, and question the developers if they agree, yes or no, that such selected methods are "poorly written".

The second questionnaire will inform the threshold value of all the methods indicated by the tool, by all the different strategies and ask the opinion of the developers about the threshold values assigned to each class, with this we will analyze the opinion of the developers for this question and their Results will be qualitatively presented.

7. Threat to validations

Some factors may influence the validity of the results to be obtained, among them:

- Level of developer expertise: The evaluation parameters of a method such as "poorly written" depends on the subjective evaluation of the developers, which can compromise the quality of the answers.
- Level of developer involvement in the project: An object that has little understanding of project implementation may tend to give inaccurate or poorly justified answers.
- One of the strategies developed in the tool allows grouping similar design rules to indicate threshold value for groups. However, if there is a design rule in the software that is not well defined, the algorithm will group as an undefined design and may be imprecise in the threshold value of those classes.

8. Results

We aim to report the results, researchers in software engineering looking for threshold definition strategies, to help implement metrics. Our results will be used with the intention of expanding the work done, will be disclosed to the company that made the study available and to the academic community of software engineering with the publication of articles.

9. Limitations

In a research involving qualitative and quantitative evaluation of the tool and the results generated, we find some limitations of this study as the difficulty of generalization of the results in view: the restricted business domain of the companies; The quantitative and variety of contexts (industrial and academic) of the enterprises; Variety of software.

In addition, the process of choosing the questions from the questionnaires and quantifying the values obtained may lead to biased results.

10. Ethical Matters

All information collected will comply with the consent protocol signed by the units and the developers, guaranteeing the confidentiality of the identity of the developers participating in the questionnaires and their freedom of opinion. All results will be reported and will be accessible to companies as well as agreed by the consent term.

REFERÊNCES

Van Gurp, J. & Bosch, J., 2002. *Design erosion: problems and causes. Journal of Systems and Software*, 61(2), pp.105–119.

Macia, I. et al., 2013. *Enhancing the detection of code anomalies with architecture-sensitive strategies*. Proceedings of the European Conference on Software Maintenance and Re engineering, CSMR, pp.177–186.

Marinescu, R., 2004. *Detection strategies: metrics-based rules for detecting design flaws*. In 20th IEEE International Conference on Software Maintenance.

Aniche, M., Gerosa, M. A., & Treude, C. (2016, September). *Developers' Perceptions on Object-Oriented Design and Architectural Roles*. In Proceedings of the 30th Brazilian Symposium on Software Engineering (pp. 63-72). ACM.

Dósea, M., Sant'Anna, C., & Santos, C. (2016). *Towards an Approach to Prevent Long Methods Based on Architecture-Sensitive Recommendations*. Sociedade Brasileira de Computação–SBC, 73.