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

Thamiris Gomes
Federal University of Bahia
Department of Computer Science
Salvador, Brazil
thamigomess@gmail.com

Rebeca Sousa
Federal University of Bahia
Department of Computer Science
Salvador, Brazil
rebecatvs@gmail.com

Luan Passos
Universidade Salvador
Mechatronics Engineering
Salvador, Brazil
luanfcpassos@gmail.com

***Abstract*—The strategies based in metrics and threshold values are mechanisms frequently used for code anomaly detection. There are a large variety of tools that propose the application of those measurements. However, there are still threshold values identification problems causing difficulties to ensure the accuracy of metrics. To obtain a metric precision, it is necessary to apply and validate strategies to calculate these threshold values by different techniques. Recent approaches used to implement the calculation of thresholds evaluate classes, in metric terms, by different strategies. There is approach that determines a generic threshold value for all the classes, considering the application domain, another aims to evaluate the architectural role and indicates a specific value for the classes of each architectural role, and the last one that differentiates the classes based on the design role. To analyze the accuracy of each strategy is necessary to apply them in a real environment with the perspective of professionals with experience on the studied project. The main goal of the study is to evaluate the accuracy of the calculation threshold techniques, in a method level, for four metrics and understand the decisions that developers made to identify a method as a false positive. For that, we are going perform a case study in which will be applied the ContextSmell tool, that implements different techniques to indicate threshold values, and analyze the recommendation of methods with implementation problems.**

***Keywords*—Metrics, threshold values, code maintenance, accuracy.**

I. Introduction

Due to the growth and complexity of software projects, implementation problems can be inevitable, and can impact on software evolution [1]. As code anomalies appear, more difficult is to identify them by manual revision. Because of that, tools that implement strategies based in software metrics and threshold values are frequently used mechanisms for code anomaly detection [2] [3] [4].

The software metrics are used with the objective of measure the quality of systems, in which they contribute to not lose the control of the inevitable bad structuring of the software, that cause impacts on quality attributes [5] [6]. So that the software metrics have satisfactory results, or useful inside the studied perspective, is important to identify the best manner of indicate the threshold values, in order to provide correct interpretations about the metric results [6].

The definitions of the threshold values imply in a process of measurements of software quality, which allow to elect possible implementation problems, presenting to the developer the necessity to perform code revision practices [7]. The threshold values determine a limit to measure if the class fits in one or more metric problems, nonetheless there are still identification problems due to particularities in each implementation. [4] [3] [8].

The main purpose of this article is to evaluate the accuracy of the threshold values calculation techniques of the *ContextSmell* tool, on the developers perspectives. The tool implements different threshold calculation strategies and recommends a list of anomalous methods, which will be analyzed by developers. The article will also investigate design strategies used by the developers to identify methods as false positive.

To reach these objectives, we will execute a case study, in which the *ContextSmell* tool will be applied on a company development environment, so the developers can evaluate the methods recommended by the tool that have implementation problems. With that in mind, it will be measured the precision and coverage of the tool, comparing the threshold values calculation strategies, understanding how developers identify method design problems.

II. Methodology

A. Background information

The 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 Lines of Code (LOC), Cyclomatic Complexity(CC), Efferent Coupling (AE) and Number of Parameters (NP). Here we present a brief explanation of these metrics:

- Lines of Code (LOC) count the number of uncommented lines of code per class [9]. The value of this metric indicates the size of a class. Lower numbers are desirable.

- Cyclomatic Complexity (CC) indicates the complexity of a program [10]. It quantifies the the linearly independent paths through the source code. Mathematically speaking, the CC of a structured program is defined with reference to the control flow graph of the program, which has E edges, N nodes and P connected components (For a single program or subroutine, P is always equal to 1), and the formula for the complexity M is as follows: M = E - N + 2*P.
  In case the exit node is connected back to the entry node, the graph becomes strongly connected [11], and the complexity M is now defined as follows: M = E - N + P. Like for LOC, a lower number is desirable.

- Number of Parameters (NOP) indicates the number of parameters that build the signature of a method [12]. Methods with a large number of NOP often indicates that classes are missing from the model. As such, lower numbers are desired.

- Efferent Coupling (CE) measures the number of data types a classes knows about [13], which includes inheritance, interface implementations, parameter types, variable types and exceptions. A large efferent coupling can indicate that a package is unfocused and may also indicate that it's unstable since it depends on the stability of all the types to which it is coupled. Efferent coupling can be reduced by extracting classes from the original class so the class is decomposed into smaller classes.

Thereby, these information enable the developers team to have more attention and do code revising of these methods, and to avoid future problems in the software development, such as, understanding, maintenance, and inclusion of new features.

B. *Research Questions*

The research questions are the following:

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

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

C. *Case Selection*

The main criteria for the case study selection was a company with systems developed in JAVA and professionals with experience on the chosen system to contribute with the study results. The analyzed unit was the Franhoufer Project Center, situated on the technological park of Bahia, which develops innovative software solutions for the market.

To execute the activities proposed by the study, we defined, along with the company, the system in which the tool would be applied and the developer involved. The results were presented to the most experiment member of the team, which answered the questionnaires as a study case procedure.

TABLE I. CASE STUDY INFORMATION

Company: Franhoufer Project Center	
Type of company	Private
Number of professionals	6
Number of professionals in the project	2
Software type	Distribution system of organs and tissues.
Domain type	WEB System
Programming language	JAVA

TABLE II. ALVES PROPOSAL

Metric/Risk	High risk	Very low risk
LOC (Lines of Code)	44	74
Cyclomatic Complexity	8	14
Number of Parameters	3	4

D. *Data Collection Routine*

The case study is a research method used to investigate an entity of phenomenon within a determined time and space. A variety of data collection procedures could be applied to execute a case study. Here we are going to describe with detail the steps that must be followed by the researcher to perform this case study:

Preparation

Before the execution of the study the participants should perform the following steps:

1) Create an development environment to execute the study;
2) Configure fixed threshold values, based on Alves [8] proposal (Table II);
3) Define and configure systems that are going to be used as design reference;
4) Define participants and systems that are going to be analyzed by each participant;
5) Execute the .jar file ContextSmellView.jar, of the tool on the available development environment;

Execution Protocol

[1]

For the study execution we must follow these steps:

1) Ask the participant to fill the Consent Letter*;
2) Ask the participant to fill the Project and Company Characterization Form *;
3) Ask the participant to fill the Participant Characterization Form *;
4) Edit the files References.txt* with the reference project path (older version or any other project with similar design) for the threshold values calculation and Analysis.txt* with the path of the project that will be studied;
5) Prepare and execute the *ContextSmell** tool in the company environment; Make a short explanation showing how the tool present possible design and implementation problems;

[1]*All the documentation templates, tool executable and further material is available at https://github.com/thamirisgomess/CaseStudy

6) Each participant should evaluate the code implementation problems listed by the tool during a maximum period of 1 hour.
7) If the output list is too extensive, they should evaluate blocks of 4 recommendations for each metric, randomly chosen, by prioritizing the identifications made individually by the techniques.
8) For each analyzed methods the participant should answer the following questions that should be registered by the researcher on the sheet: Should be refactored because of this problem? If disagree, what are the motivations?

The data collection was performed with one developer of the selected project in a section that took 1 hour and 30 minutes since the environment preparation until finish the questionnaire filling process.

III. DATA ANALYSIS

We made the qualitative data analysis based on the developers questionnaire answers and we used recall and precision to compare the accuracy of the tool. The technique that recommended a method with a design problem and accorded with the developer was marked a true positive and the technique that did not recommended this same method was classified as false positive. On the other hand, the recommended method that was not confirmed by the developers was classified as false positive and the techniques that did not recommended the same method were classified as true negatives.

From that, we calculated for each isolated metric the recall and precision, based on the quantity of classifications: True positive (TP), false negative (FN), false positive (FP) and true negative (TN). The recall value for each technique is performed on the true positive and false negative values (TP/(TP+FN)). On the other hand, the precision is calculated with the true positives and false positives (TP/(TP+FP)). Based on recall and precision, we calculate the accuracy for each technique ((TP+TN)/(TP+TN+FN+FP)).

The qualitative data analysis was performed through the questionnaire answers related to design strategy used by the developer to point a method as a false positive. The recommended method by the tool with some design problem and that the developer disagreed, is considered as a false positive. Based on the strategy described by the developer to reach this conclusion, we create new design annotations so different threshold values could be calculated for the classes for these methods.

IV. RESULTS

In this section, we will present the qualitative and quantitative results of the study, to answer both research questions. On the section A, we will present the results for the quantitative analysis and on the section B, the qualitative results.

A. *Quantitative Analysis*

tool was executed in one of the modules of the system with 286 classes and 54.737 code lines and had its results analyzed by a developer with 5 years of experience on the studied project. The developer analyzed the methods recommended by the tool and answered the questionnaire agreeing or not with the pointed design problem.
the calculation of threshold values we used the techniques:

- A : based on the data analysis of a representative gather of systems. (benchmark) [8];
- R: Thresholds extracted from systems that follow the same design rules;
- D: Thresholds extracted from systems that follow the same design rules and considering design roles;
- X: Thresholds extracted from systems considering architectural roles.

TABLE III. LINES OF CODE

	A	R	D	X
True Positive	35	43	29	67
False Negative	64	56	70	32
False Positive	6	8	2	9
True Positive	4	2	8	1
Recall	0,35	0,43	0,29	0,68
Precision	0,85	0,84	0,94	0,88
Accuracy	0,36	0,41	0,34	0,62
F-measure	1,06	1,30	0,87	2,03

TABLE IV. CYCLOMATIC COMPLEXITY

	A	R	D	X
True Positive	50	29	22	16
False Negative	7	28	36	41
False Positive	41	50	2	12
True Positive	9	0	47	38
Recall	0,87	0,50	0,37	0,28
Precision	0,54	0,36	0,91	0,57
Accuracy	0,55	0,27	0,64	0,50
F-measure	2,61	1,50	1,11	0,84

Table III presents the results calculated by the Lines of Code metric (LOC), in which has the best accuracy for the X technique with the value of 0,62. On the table IV, there is data from the Cyclomatic Complexity (CC), in which the D technique performed the best result with the value of 0,64. For the Efferent Coupling (AE), on table V, shows D as the best technique with 0,65. And finally, with the Number of Parameters (NOP), the best result was also D with the value of 0,71, on the table VI.

The accuracy of the techniques was analyzed through recall and precision values, in which recall represents a instance fraction of the information recovering, contemplated as completeness or quantity. Yet the precision represents the utility and relevance of the extracted data, being a quality measurement. The F-measure function is the harmonic measurement that determines the same weight for recall and precision, giving

TABLE V. EFFERENT COUPLING

	A	R	D	X
True Positive	23	19	42	43
False Negative	42	46	23	22
False Positive	0	1	2	4
True Positive	6	5	4	2
Recall	0,35	0,29	0,65	0,66
Precision	1,00	0,95	0,95	0,91
Accuracy	0,41	0,34	0,65	0,63
F-measure	1,05	0,87	1,93	1,98

TABLE VI. NUMBER OF PARAMETERS

	A	R	D	X
True Positive	36	31	36	15
False Negative	18	23	18	39
False Positive	3	5	0	1
True Positive	6	4	9	8
Recall	0,67	0,57	0,67	0,28
Precision	0,92	0,86	1,00	0,94
Accuracy	0,67	0,56	0,71	0,37
F-measure	2,00	1,72	2,00	0,83

the same importance for both of them.

The comparison between the results from the tables III, IV, V and VI respond the RQ1: Are there any differences in the accuracy of the techniques on the calculation of threshold values for the metrics in the metrics level? It shows that there are differences in the accuracy of techniques A, R, D and X.

B. *Qualitative Analysis*

For a Qualitative Analysis, the developer justified the methods recommended by the tool as false positive. Thus, the developer disagreed with some design problems and justified the reason for no need to refactor the method. For example, the file manipulation methods, which were suggested by the tool to be long methods, were not agreed by the participant and justified by the lack of need to refactor the methods that manipulate files. With this, the tool will be tweaked to not regard long methods as ones that manipulate files.

To answer the second Research Question: Which Design Decisions influence the developers to indicate methods as false positives?. The developer described the following reasons, which contribute with possible modifications in the proposed tool.

- "Necessary Parameters and clustering them in classes does not make sense when it comes to domain logic";
- "26 lines are necessary" - for a file manipulation method;
- "Application Initialization method";
- "Equals method";
- "Validation function";
- "System Parameters Initialization";

V. LIMITATIONS

In a research that involves qualitative and quantitative evaluation of the tool and the generated results, we found a few limitations of this study like the (difficulty) to generalize the obtained results given that: The restricted domain of business of companies; The quantitative and variety of contexts (industrial and academic) of the enterprises; The variety of evaluated software.

Also, the process to choose the questionnaire questions and quantification of the obtained values might lead to biased results.

VI. THREATS TO VALIDITY

Some factors might influence the validity of the obtained results, including:

- Developers expertise level: The evaluation parameters of a method as badly written depends on the subjective evaluation of the developers, which can compromise the answers quality;
- Involvement in the project: An implemented object that is not widely comprehended by the developers might lead to answers that are not very precise or badly justified;
- One of the strategies in the tool makes clustering similar design rules to indicate threshold values for the groups possible. However, if a design rule which is not well defined exists in the software, the algorithm will cluster it as undefined design, which can be imprecise for the threshold values of those classes.

VII. CONCLUSION

This paper evaluated the usage of the *ContextSmell* tool in an industrial environment, and the analysis gave us [favorable] results, although the tool itself still needs some adjustments, according to the developer who filled the questionnaire.

As of the results, even with the difference in the techniques varying depending on the analyzed metric, the study presents D as the highest accuracy in 3 of the 4 studied metrics. Furthermore, we notice the necessity to consider the design roles for threshold value calculation because even if a class belongs to the same architectural role of other classes, they can have differences in their design roles and this can propitiate different threshold values.

REFERENCES

[1] Fowler, M. et al. *Refactoring: Improving the Design of Existing Code* Reading, Massachusetts: Addison Wesley, 1999.

[2] Dhambri, Karim; SAHRAOUI, Houari; POULIN, Pierre. *Visual detection of design anomalies.* In: Software Maintenance and Reengineering, 2008. CSMR 2008. 12th European Conference on. IEEE, 2008. p. 279-283.

[3] Aniche, Maurcio et al. SATT: *Tailoring Code Metric Thresholds for Different Software Architectures.* In: Source Code Analysis and Manipulation (SCAM), 2016 IEEE 16th International Working Conference on. IEEE, 2016. p. 41-50.

[4] Fontana, Francesca Arcelli et al. *Automatic metric thresholds derivation for code smell detection.* In: Proceedings of the Sixth International Workshop on Emerging Trends in Software Metrics. IEEE Press, 2015. p. 44-53.

[5] Marinescu, Radu. *Detection strategies: Metrics-based rules for detecting design flaws.* In: Software Maintenance, 2004. Proceedings. 20th IEEE International Conference on. IEEE, 2004. p. 350-359.

[6] Lanza, Michele; MARINESCU, Radu; DUCASSE, Stphane. *Object-oriented metrics in practice: using software metrics to characterize.* Evaluate, and Improve the Design of Object-Oriented Systems, p. 220, 2006.

[7] Lavazza, Luigi; MORASCA, Sandro. *An Empirical Evaluation of Distribution-based Thresholds for Internal Software Measures.* In: Proceedings of the The 12th International Conference on Predictive Models and Data Analytics in Software Engineering. ACM, 2016. p. 6.

[8] Alves, Tiago L.; YPMA, Christiaan; VISSER, Joost. *Deriving metric thresholds from benchmark data.* In: Software Maintenance (ICSM), 2010 IEEE International Conference on. IEEE, 2010. p. 1-10.

[9] M. Lorenz and J. Kidd, *Object-oriented software metrics.* New York: Prentice Hall, p. 146, 1994.

[10] T.J. McCabe, *A Complexity Measure*, IEEE Transactions on Soft. Engineering, Vol. SE-2, Issue 4, pp. 308320, Dec., 1976.

[11] Sharir, Micha*A strong-connectivity algorithm and its applications in data flow analysis.* Computers Mathematics with Applications 7.1 (1981): 67-72.

[12] A.R. Sharafat and L. Tahvildari, *Change prediction in objectoriented software systems:A probabilistic approach*, Journal of Software, vol.3, no. 5, pp. 26-39, 2008.

[13] Martin, R., 1994. *OO design quality metrics. An analysis of dependencies*, 12, pp.151-170.