***SUMMARY***

***Developer-Driven Code Smell Prioritization***

***Authors:***

*Fabiano Pecorelli,*

*Fabio Palomba,*

*Foutse Khomh,*

*Andrea DeLucia*

***Universities:***

*1SeSaLab-University of Salerno, Italy— 2École Poly technique de Montréal, Canada*

***Date:*** *MSR ’20, October 5–6, 2020, Seoul, Republic of Kore*

***INTRODUCTION:***

*As a software system evolves, continuous modifications are required to a daptit to new requirement sand/ or changing environment so even fixed effects that can preclude its correct functioning [69]. One of the main indicators of technical debt is represented by the presence of code smells [26], which are poor Code smells have been of ten associated to a decrease of program comprehensibility [1], maintainability [38,59], testability [28] as well as an increase of maintenance effort and costs [80]. These findings have motivated researchers to propose automated mechanisms to support developrs in both the identification [4,18,21] and removal [51] of code smells, obtaining promising results. While some researchers provided initial attempts toward the prioritization of code smells using measures of severity derived from software metrics [3,25,49,93], the available solutions either rely on predefined heuristics that have not been empirically assessed or do not address the problem of providing developers with recommendations aligning with their perception of design issues, thus possibly being still in effective in practice The main result of the study high lights that the devised approach can classify the developer’s perceived criticality of code smells with an F-Measure ranging between 72% and 85%. Moreover, we discovered that, depending on the code smell type, specific features are more relevant to classify its criticality.*

***ABSTRACT***

*Code smells are symptoms of poor implementation choices applied during software evolution. While previous research has devoted effort in the definition of automated solutions to detect them, still little is known on how to support developers when prioritizing them. Some works attempted to deliver solutions that can rank smell instances based on their severity, computed on the basis of software metrics. However, this may not be enough since it has been shown that the recommendations provided by current approaches do not take the developer’s perception of design issues into account. In this paper, we perform a first step toward the concept of developer-driven code smell prioritization and propose an approach based on match in earning able to rank code smells according to the perceived criticality that developers assign to them. We evaluate our technique in an empirical study to investigate its accuracy and the features that are more relevant for classifying the developer’s perception. Finally, we compare our approach with a state-of-the-art technique. Key findings show that the our solution has an F-Measure up to 85% and out performs the base line approach*

***CONCLUSION***

*This paper presented a novel code smell prioritization approach based on the developers’ perceived criticality of code smells. We exploited several aspects related to code quality to predict he criticality of code smells, computed by collecting feed back from original developers about their perception of 1,332 code smell instances. Then, we applied several machine learning techniques to classify the code smell criticality in a three-level variable, and compared their results with a state-of-the-art tool. The results reported Random Forest to be the best machine learning algorithm with an F-measure rang between 72% and 85%. More over,we found that our approach is, on average, 20% more accurate than the considered base line when classifying the perceived criticality of code smells. Future work includes (1) further improvement soft the approach, e.g., by considering social network analysis metrics, (2) an experimentation with a large r number of code smells, and (3) an in-vivo assessment of our technique.*