Too trivial to test? An inverse view on defect prediction to identify methods with low fault risk

Rainer Niedermayr[1] Tobias Röhm[2] Stefan Wagner[3]

Abstract: To cope with the scarce resources for testing, teams can apply defect prediction to identify fault-prone code regions. However, defect prediction tends to low precision in cross-project prediction scenarios. We take an inverse view on defect prediction and aim to identify methods that can be deferred when testing because they contain hardly any faults due to their code being "trivial". We compute code metrics and apply association rule mining to create rules for identifying methods with low fault risk (LFR) and assess our approach with six Java open-source projects containing precise fault data at the method level. Our results show that inverse defect prediction can identify approx. 32–44% of the methods of a project to have a LFR; on average, they are about six times less likely to contain a fault than other methods. Our approach identifies methods that can be treated with less priority in testing activities and is well applicable in cross-project prediction scenarios.

Keywords: Testing; Inverse defect prediction; Fault risk; Low-fault-risk methods

1 Introduction

In a perfect world, it would be possible to completely test every new version of a software application before it was deployed into production. In practice, however, software development teams often face a problem of scarce test resources. Hence, development teams need to prioritise and limit their testing scope by restricting the code regions to be tested. Defect prediction identifies code regions that are likely to contain a fault and should therefore be tested.

This paper suggests, implements, and evaluates another view on defect prediction: inverse defect prediction (IDP). The idea is to identify code artefacts that are so trivial that they contain hardly any faults and thus can be deferred or ignored in testing. IDP also uses a set of metrics that characterise artefacts, applies transformations to pre-process metrics, and uses a machine-learning classifier to build a prediction model. The difference rather lies in the predicted classes: IDP identifies methods that exhibit a low fault risk (LFR) with high certainty and does not make an assumption about the remaining methods. As a consequence, the objective of the prediction also differs. Defect prediction aims to achieve a high recall, such that as many faults as possible can be detected, and a high precision, such that only few false positives occur. In contrast, IDP aims to achieve high precision to ensure that

[1] CQSE GmbH, Munich, Germany niedermayr@cqse.eu
[2] CQSE GmbH, Munich, Germany roehm@cqse.eu
[3] University of Stuttgart, Stuttgart, Germany stefan.wagner@iste.uni-stuttgart.de

doi:10.18420/SE2020_41

LFR methods contain indeed hardly any faults, but it does not necessarily seek to predict all non-faulty methods. This is a summary of the full article published in PeerJ Computer Science in 2019 [NRW19].

2 IDP Approach

In the IDP approach, we use 39 source code metrics for Java such as maximum nesting depth or the number of if conditions as well as two derived metrics. Furthermore, we take into account whether the method to be analysed is a constructor, a getter/setter, an empty method, delegation method or a toString method. We use these metrics to create rules that indicate when a method is non-faulty using association rule mining.

3 Empirical Study

Using the Defects4J data [JJE14], we could evaluate the IDP approach on six open source Java systems. We found that our approach on average only 0.3% of the methods classified as low fault risk are actually faulty. Those identified methods are 5.7 times less likely to contain a fault. About 15–20% of the lines of code are identified as low fault risk.

4 Conclusion

We show that IDP using association rule mining on code metrics can successfully identify LFR methods. The identified methods contain considerably fewer faults than the average code and can provide a savings potential for QA activities. Our results furthermore suggest that the IDP approach can be applied in a cross-project prediction scenario at the method level.

Bibliography

[JJE14] Just, René; Jalali, Darioush; Ernst, Michael D.: Defects4J: a database of existing faults to enable controlled testing studies for Java programs. In: International Symposium on Software Testing and Analysis, ISSTA '14. ACM, pp. 437–440, 2014.

[NRW19] Niedermayr, Rainer; Röhm, Tobias; Wagner, Stefan: Too trivial to test? An inverse view on defect prediction to identify methods with low fault risk. PeerJ Computer Science, 5:e187, 2019.