Feature Defect Prediction

Stefan Strüder
University of Koblenz-Landau, Germany

Daniel Strüber
Radboud University Nijmegen, Netherlands

Thorsten Berger
Chalmers | University of Gothenburg, Sweden

ABSTRACT

Software errors are a major nuisance in software development and can lead not only to damage of reputation but also to considerable financial losses for companies. For this reason, numerous techniques for detecting and predicting errors have been developed over the past decade, which are largely based on machine learning methods. The usual approach of these techniques is to predict errors at file level. For some years now, however, the popularity of feature-based software development has been increasing - a paradigm that relies on function increments of a software system (features) and thus ensures a wide variability of the software product. A common implementation technique for features is based on annotations with preprocessor instructions, such as `#IFDEF` and `#IFNDEF`, whose code is spread over several files of the software's source code files ("code scattering"). A bug in such a feature code can have far-reaching consequences for the functionality of the entire software. If a part of the feature code contains errors, the entire function of the feature becomes faulty and may lead to the failure of the entire functionality of the Software (features are "cross-cutting"). This problem is the subject of this thesis. A prediction technique for faulty features is developed, which is based on methods of machine learning. The evaluation of eight classifiers, each based on an individual classification algorithm, shows that the feature-based data set created for this thesis allows an accuracy of up to 92% for the prediction of faulty or error-free features. It is also shown how the feature orientation aspect was incorporated into the creation of the dataset and what results were achieved compared to the traditional file-based methodology.

REFERENCES

[1] Rodrigo Queiroz, Thorsten Berger, and Krzysztof Czarnecki. 2016. Towards predicting feature defects in software product lines. In *Proceedings of the 7th International Workshop on Feature-Oriented Software Development - FOSD 2016*. ACM Press. https://doi.org/10.1145/3001867.3001874

ACM Reference Format:
Stefan Strüder, Daniel Strüber, and Thorsten Berger. 2020. Feature Defect Prediction. In *Proceedings of ACM Conference (Conference'17)*. ACM, New York, NY, USA, 2 pages.

Conference'17, July 2017, Washington, DC, USA
2020. ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00

Table 1: Overview of used metrics

	Metric	Description	Source
Process metrics	Number of commits (COMM)	Number of commits associated with the feature/file in a release.	[1]
	Number of active developers (ADEV)	Number of developers who have edited (changed, deleted or added) the feature / file within a release	[1]
	Number of distinct developers (DDEV)	Cumultative number of developers who have edited (changed, deleted or added) the feature / file within a release	[1]
	Experience of all develepoers (EXP)	geometric mean of the experience of all developers who have edited (changed, deleted or added) the feature / file within a release.~ Experience is defined as the sum of the changed, deleted or added lines in the commits associated with the feature / file.	[1]
	Experience of the most involved developers (OEXP)	Experience of the developer who has edited (changed, deleted or added) the feature / file most often within a release. Experience is defined as the sum of changed, deleted, or added lines in the commits associated with the feature/file.	[1]
	Degree of modifications (MODD)	Number of edits (change, removal, extension) of the feature / file within a release.	*
	Scope of modifications (MODS)	Number of edited features / files within a release (feature or file overlapping value). Idea: The more features / files have been edited in a release, the more error-prone they seem to be.	*
Code metrics	Lines of code (NLOC)	Average number of lines of code of the files associated with the feature / ~file within a release.	*
	Cyclomatic Complexity (CYCO)	Average cyclomatic complexity of the files associated with the feature / file within a release.	*
	Number of added lines (ADDL)	Average number of lines of code added to the files associated with the feature / file within a release.	*
	Number of removed lines (REML)	Average number of lines of code deleted from the files associated with the feature / from the file within a release	*
** These values were calculated based on the metadata obtained with PyDriller. Feature-level metrics were calculated based on the metadata of the underlying files.*			