Paper Eight Summary

Matthew Neal, Joseph Sankar, and Alexander Sobran

November 14, 2015

Reference

Jiang et al. [1] listed below.

Important Keywords

Personalized change classification A defect prediction modeling approach that works by building a change classification model for each specific developer. The methodology promoted by the paper.

Traditional change classification A defect prediction modeling approach that works by building a single change classification model for all developers in aggregate. Used as a baseline, since it is the way that defect prediction is generally carried out.

Multivariate Regression Splines (MARS) A defect prediction modeling approach that works with a global aggregate of developers, but groups data to minimize the fitting error. Used as a baseline in the study, as a sort of middle way between traditional and personalized change classification.

Characteristic vector One of three classes of features the author used to describe the characteristics of source code. Basically a vector of integers, where each integer corresponds to a count of node types, e.g. reserved keywords.

Bag-of-words Another of three classes of features used. Stores words within the code, e.g. function names.

Feature Extraction

Motivational Statements The idea is that generally studies for defect prediction aggregate all developers together in one model, but there are many differences between individual developers. One developer may be stronger in one area than another and will make more errors in a certain area than

another. Given that this is the case, the author believes that it makes sense to create models for individual developers to predict defects since it had not been done prior to this paper.

Informative Visualization Figure two on page 281 is a really telling visual for proving the author's point that there are drastic differences in the skill of different developers. The interesting thing about these developers in particular is the fact that this graph is drawn from Linux kernel developers, where you would expect some of the most skilled developers worldwide, but we find errors on things like for loops and modulo just like any normal developer. But, in terms of the paper, this visual really shows the value in producing models for specific developers.

Sampling Procedures The author chose six open source projects: Linux, PostgreSQL, Xorg, Eclipse, Lucene, and Jackrabbit. The author states that these projects have enough change history data to build good experimental models, and they are commonly found within the literature, which makes it easier to compare their results with those of other researchers.

Future Work The author mentioned improving defect prediction models to giving more of an explanation of its reasoning so management can make more informed decisions. One of the threats to validity was that only open source projects were used, leaving open the possibility to run these models on closed source systems. They mentioned that their bug data contains noise that might have affected their results and suggested reducing the noise levels through advanced techniques. Finally, they noted that their selection of developers and changes might bias the model and hope to experiment with different selection techniques in the future.

Possible Improvements

- Even though the author has shown that the results in terms of both number of bugs found and F1 are statistically significant, it seems that they are overplaying their hand a bit. Even though the results are statistically significant they seem only to improve on CC and Mars by a very small margin in terms of F1 and not at all in several cases. They do not even bother discussing this fact in terms of their Threats to Validity.

- There is a section on future work, but it is incredibly short, and isn't even broken out into its own section given its brevity. A bit more thought put towards future work would have been nice.

- The author goes to the trouble of putting together a second methodology for defect prediction in PCC+, but really does not go to a lot of trouble in proving anything about its efficacy. It seems like a bit of an afterthought and may well have made up another paper entirely if the results had been more conclusive.

Connection to Other Papers

The author used Cost Effectiveness as their most important metric in the paper as discussed in [2]. They make the case that at when looking at only 20% of a particular code base PCC ends up being a better predictor of defects.

References

[1] Tian Jiang, Lin Tan, and Sunghun Kim. Personalized defect prediction. In *Automated Software Engineering (ASE), 2013 IEEE/ACM 28th International Conference on*, pages 279–289, Nov 2013.

[2] Foyzur Rahman, Daryl Posnett, and Premkumar Devanbu. Recalling the "imprecision" of cross-project defect prediction. In *Proceedings of the ACM SIGSOFT 20th International Symposium on the Foundations of Software Engineering*, FSE '12, pages 61:1–61:11, New York, NY, USA, 2012. ACM.