Learning Effective Changes for Software Projects

Rahul Krishna
Department of Computer Science
North Carolina State University
Email: rkrish11@ncsu.edu

***Abstract*—The primary motivation of much of software analytics is decision making. How do you make these decisions? Should one make decisions based on lessons that arise from within a particular project or should one generate these decisions from across multiple projects? If sufficient data is not available within a project, can practitioners learn lessons from other projects? This work is an attempt to answer these questions. Our work was motivated by a realization that much of the current generation software analytics tools focus primarily on prediction algorithms. Indeed prediction is a useful task, but it is usually followed by "planning" about what actions need to be taken. This research seeks to address the planning task by seeking methods that support actionable analytics that offer clear guidance on *what to do* within the context of a specific software project. Specifically, we propose the XTREE algorithm for generating a set of actionable plans. Each of these plans, if followed will improve the quality of the software project.**

Keywords*—Data mining, actionable analytics, bellwethers, defect prediction.

I. Introduction

Over the past decade, advances in AI have enabled a widespread use of data analytics in software engineering. For example, we can now estimate how long it would take to integrate the new code [1], where bugs are most likely to occur [2], [3], or how much effort it will take to develop a software package [4], [5], etc. Despite these successes, there are some operational shortcomings with certain software analytic tools. Business users lament that most software analytics tools, "Tell us what *is*. But they don't tell us *what to do*". A similar concern was raised by several researchers at a recent workshop on "Actionable Analytics" at 2015 IEEE conference on Automated Software Engineering [6].

For example, most software analytics tools in the area of detecting software defects are mostly *prediction* algorithms such as support vector machines [7], naive bayes classifiers [8], logistic regression [8], decision trees [9], etc. These prediction algorithms report what combinations of software project features predict for say the number of defects. But this is different task to *planning*, which answers a more pressing question: what to *change* in order to *improve* quality. Accordingly, in this research, we seek tools that offer clear guidance on what to do in a specific project.

The tool assessed in this paper is the XTREE *planning* tool [10]. XTREE employs a $cluster + contrast$ approach to planning where it (a) *Clusters* different parts of the software project based on a quality measure (e.g. the number of defects); (b) Reports the *contrast sets* between neighboring clusters. Each of these contrast sets represent the difference between these clusters and they can be interpreted as plans, i.e.,

- If your current project falls into cluster C_1,
- Some neighboring cluster C_2 has better quality.
- Then the difference $\Delta = C_2 - C_1$ is a *plan* for changing a project such that it *might* have higher quality.

II. Conclusion

Acknowledgment

¡¡¡¡¡¡¡ HEAD

References

[1] J. Czerwonka, R. Das, N. Nagappan, A. Tarvo, and A. Teterev, "Crane: Failure prediction, change analysis and test prioritization in practice – experiences from windows," in *Software Testing, Verification and Validation (ICST), 2011 IEEE Fourth International Conference on*, march 2011, pp. 357 –366.

[2] T. J. Ostrand, E. J. Weyuker, and R. M. Bell, "Where the bugs are," in *ISSTA '04: Proceedings of the 2004 ACM SIGSOFT international symposium on Software testing and analysis.* New York, NY, USA: ACM, 2004, pp. 86–96.

[3] T. Menzies, A. Dekhtyar, J. Distefano, and J. Greenwald, "Problems with Precision: A Response to "Comments on 'Data Mining Static Code Attributes to Learn Defect Predictors'"," *IEEE Transactions on Software Engineering*, vol. 33, no. 9, pp. 637–640, sep 2007. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4288197

[4] B. Turhan, A. Tosun, and A. Bener, "Empirical evaluation of mixed-project defect prediction models," in *Software Engineering and Advanced Applications (SEAA), 2011 37th EUROMICRO Conference on.* IEEE, 2011, pp. 396–403.

[5] E. Kocaguneli, T. Menzies, A. Bener, and J. Keung, "Exploiting the essential assumptions of analogy-based effort estimation," *IEEE Transactions on Software Engineering*, vol. 28, pp. 425–438, 2012, available from http://menzies.us/pdf/11teak.pdf.

[6] J. Hihn and T. Menzies, "Data mining methods and cost estimation models: Why is it so hard to infuse new ideas?" in *2015 30th IEEE/ACM International Conference on Automated Software Engineering Workshop (ASEW)*, Nov 2015, pp. 5–9.

[7] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[8] S. Lessmann, B. Baesens, C. Mues, and S. Pietsch, "Benchmarking Classification Models for Software Defect Prediction: A Proposed Framework and Novel Findings," *IEEE Trans. Softw. Eng.*, vol. 34, no. 4, pp. 485–496, jul 2008.

[9] J. R. Quinlan, "Induction of decision trees," *Machine learning*, vol. 1, no. 1, pp. 81–106, 1986.

[10] R. Krishna, T. Menzies, and L. Layman, "Less is more: Minimizing code reorganization using XTREE," *Information and Software Technology*, mar 2017. [Online]. Available: http://arxiv.org/abs/1609.03614