Exploring "Bellwether" Effect in Software Engineering Data Sets

Rahul Krishna, Tim Menzies
Computer Science, NC State, USA
{i.m.ralk, tim.menzies}@gmail.com

Lucas Layman
Fraunhofer CESE, College Park, USA
llayman@cese.fraunhofer.org

ABSTRACT

Categories/Subject Descriptors: D.2 [Software Engineering]; I.2 [Artificial Intelligence];
Keywords:

1. INTRODUCTION

Numerous *local learning* results show that we should mistrust general conclusions (made over a wide population of projects) since they may not hold for projects. Posnett et al. [1] discuss *ecological inference* in software engineering, which is the concept that what holds for the entire population also holds for each individual. They learn models at different levels of aggregation (modules, packages, files) and show that models that work at one level of aggregation can be sub-optimal at others. For example, Yang et al. [2], Bettenburg et al. [3], and Menzies et al. [4] all explore the generation of models using *all* data versus *local* samples that more specific to particular test cases. These papers report that better models (sometimes with much lower variance in their predictions) are generated from local information. These results have an unsettling effect on anyone struggling to propose policies for an organization. If all prior conclusions can change for the new project, or some small part of a project, how can any manager ever hope to propose and defend IT policies (e.g., when should some module be inspected, when should it be refactored, where to focus expensive testing procedures, etc.)?

If we cannot *generalize* to all projects and all parts of current projects, perhaps a more achievable goal is to *stabilize* the pace of conclusion change. While it may be a fool's errand and wait for eternal and global SE conclusions, one possible approach is for organizations to declare N prior projects as *reference projects*, from which lessons learned will be transferred to new projects. In practice, using such reference sets requires three processes:

1. Finding the reference sets (this paper shows that finding them may not be a too complex task, at least for defect prediction).

2. Recognizing when to update the reference set. In practice, this could be as simple as noting when predictions start failing for new projects—at which time, we would loop to point #1.

3. Transferring lessons from the reference set to new projects.

In this approach, the policies of the organization will be stable just as long as the reference set is not updated. In this paper, we do not address the pace of change in the reference set (that is left for future work). Rather, we focus on point #3: transferring lessons from the reference set to new projects. To support this third point, we need to resolve the problems that this paper addresses (data expressed in different terminology cannot transfer till there is enough data to match old projects to new).

2. MOTIVATION

There has been a general stride over the past three decades to incorporate lessons from within the company over the past to predict defects in on going projects.

3. REFERENCES

[1] Daryl Posnett, Vladimir Filkov, and Premkumar Devanbu. Ecological inference in empirical software engineering. In *2011 26th IEEE/ACM International Conference on Automated Software Engineering (ASE 2011)*, pages 362–371. IEEE, nov 2011.

[2] Ye Yang, Lang Xie, Zhimin He, Qi Li, Vu Nguyen, Barry Boehm, and Ricardo Valerdi. Local bias and its impacts on the performance of parametric estimation models. In *Proceedings of the 7th International Conference on Predictive Models in Software Engineering - Promise '11*, pages 1–10, New York, New York, USA, 2011. ACM Press.

[3] Nicolas Bettenburg, Meiyappan Nagappan, and Ahmed E. Hassan. Think locally, act globally: Improving defect and effort prediction models. In *2012 9th IEEE Working Conference on Mining Software Repositories (MSR)*, pages 60–69. IEEE, jun 2012.

[4] Tim Menzies, Andrew Butcher, Andrian Marcus, Thomas Zimmermann, and David Cok. Local vs. global models for effort estimation and defect prediction. In *2011 26th IEEE/ACM International Conference on Automated Software Engineering (ASE 2011)*, pages 343–351. IEEE, nov 2011.

ASE '16 Singapore, Singapore.
ACM ISBN 978-1-4503-2138-9.
DOI: 10.1145/1235

Figure 1: Bellwether Effect

	Ant	Camel	Ivy	Jedit	Log4j	Lucene	Poi	Velocity	Xalan	Xerces
Ant						×				
Camel						×				
Ivy						×				
Poi						×				
Velocity						×				
Xalan						×				
Xerces						×				
Jedit	×									
Lucene					×					
Log4j										×