Chapter 6

Genetic Programming for Cross–Release Fault Count Predictions in Large and Complex Software Projects

Wasif Afzal
Blekinge Institute of Technology, Sweden

Richard Torkar
Blekinge Institute of Technology, Sweden

Robert Feldt
Blekinge Institute of Technology, Sweden

Tony Gorschek
Blekinge Institute of Technology, Sweden

ABSTRACT

Software fault prediction can play an important role in ensuring software quality through efficient resource allocation. This could, in turn, reduce the potentially high consequential costs due to faults. Predicting faults might be even more important with the emergence of short-timed and multiple software releases aimed at quick delivery of functionality. Previous research in software fault prediction has indicated that there is a need i) to improve the validity of results by having comparisons among number of data sets from a variety of software, ii) to use appropriate model evaluation measures and iii) to use statistical testing procedures. Moreover, cross-release prediction of faults has not yet achieved sufficient attention in the literature. In an attempt to address these concerns, this paper compares the quantitative and qualitative attributes of 7 traditional and machine-learning techniques for modeling the cross-release prediction of fault count data. The comparison is done using extensive data sets gathered from a total of 7 multi-release open-source and industrial software projects. These software projects together have several years of development and are from diverse application areas, ranging from a web browser to a robotic controller software. Our quantitative analysis suggests that genetic programming (GP) tends to have better consistency in terms of goodness of fit and accuracy across majority of data sets. It also has

DOI: 10.4018/978-1-61520-809-8.ch006
comparatively less model bias. Qualitatively, ease of configuration and complexity are less strong points for GP even though it shows generality and gives transparent models. Artificial neural networks did not perform as well as expected while linear regression gave average predictions in terms of goodness of fit and accuracy. Support vector machine regression and traditional software reliability growth models performed below average on most of the quantitative evaluation criteria while remained on average for most of the qualitative measures.

1. INTRODUCTION

The development of quality software on time and within stipulated cost is a challenging task. One influential factor in software quality is the presence of software faults, which have a potentially considerable impact on timely and cost-effective software development. Thus fault prediction models have attracted considerable interest in research (as shown in Section 2). A fault prediction model uses historic software quality data in the form of metrics (including software fault data) to predict the number of software faults in a module or a release (Taghi and Naeem, 2002). Fault predictions for a software release are fundamental to the efforts of quantifying software quality. A fault prediction model helps a software development team in prioritizing the effort spent on a software project and also for selective architectural improvements. This paper presents both quantitative and qualitative evaluations of using genetic programming (GP) for cross-release predictions of fault count data gathered from open source and industrial software projects. Fault counts denotes the cumulative faults aggregated on a weekly or monthly basis. We quantitatively compare the results from traditional and machine learning approaches to fault count predictions and also assess various qualitative criteria for a better trade-off analysis. The main purpose is to increase empirical knowledge concerning innovative ways of predicting fault count data and to apply the resulting models in a manner that is suited to multi-release software development projects.

Regardless of large number of studies on software fault prediction, there is little convergence of results across them. This non-convergence of results is also highlighted by other authors such as (Stefan, Bart, Christophe & Swantje, 2008; Magnus and Per, 2002). This necessitates further research to increase confidence in the results of fault prediction studies. Stefan et al. (2008) identifies three sources of bias in fault prediction studies: use of small and proprietary data sets, inappropriate accuracy indicators and limited used of statistical testing procedures. Magnus and Per (2002) further highlight the need for the authors to provide information about context, data characteristics, statistical methods and chosen parameters; so as to encourage replications of these studies. These factors, more or less, contribute to a lack of benchmarking in software fault prediction studies. Therefore, benchmarking, although recommended by (Susan, Steve & Richard, 2003) as a way to advance research, is still open for further research in software fault prediction studies.

One challenge in having trustworthy software fault prediction models is the nature of typical software engineering data sets. Software engineering data come with certain characteristics that complicate the task of having accurate software fault prediction models. These characteristics include missing data, large number of variables, strong collinearity between the variables, heteroscedasticity, complex non-linear relationships, outliers and small size (Gray and MacDonell, 1997). Therefore, “it is very difficult to make valid assumptions about the form of the functional relationship between the variables” (Lionel, Victor & William, 1992, pp. 931). Moreover, the acceptability of models has seen little success due to lack of meaningful explanation of the relationship among different