Chapter 7.15
Improving Credibility of Machine Learner Models in Software Engineering

Gary D. Boetticher
University of Houston – Clear Lake, USA

ABSTRACT

Given a choice, software project managers frequently prefer traditional methods of making decisions rather than relying on empirical software engineering (empirical/machine learning-based models). One reason for this choice is the perceived lack of credibility associated with these models. To promote better empirical software engineering, a series of experiments are conducted on various NASA datasets to demonstrate the importance of assessing the ease/difficulty of a modeling situation. Each dataset is divided into three groups, a training set, and "nice/nasty" neighbor test sets. Using a nearest neighbor approach, "nice neighbors" align closest to same class training instances. "Nasty neighbors" align to the opposite class training instances. The "nice", "nasty" experiments average 94% and 20% accuracy, respectively. Another set of experiments show how a ten-fold cross-validation is not sufficient in characterizing a dataset. Finally, a set of metric equations is proposed for improving the credibility assessment of empirical/machine learning models.

INTRODUCTION

Software Project Management: State-of-Practice

Software project management has improved over the years. For example, the Standish Group, a consulting company, which has been studying IT management since 1994 noted in their latest release of the *Chaos Chronicles* (The Standish Group, 2003) that, "2003 Project success rates improved by more than 100 percent over the 16 percent rate from 1994." Furthermore, "Project

failures in 2003 declined to 15 percent of all projects. This is a decrease of more than half of the 31 percent in 1994."

Even with these successes, there are still significant opportunities for improvement in software project management. Table 1 shows several "state-of-practice" surveys collected in 2003 from IT companies in the United States (The Standish Group, 2003); South Africa (Sonnekus & Labuschagne, 2003); and the United Kingdom (Sauer & Cuthbertson, 2003).

According to the *Chaos Chronicles* (The Standish Group, 2003), *successful projects* refers to projects that are completed on time and within budget with all features fully implemented; *project challenged* means that the projects are completed, but exceed budget, go over time, and/or are lacking some/all of the features and functions from the original specifications; and *project failures* are those projects which are abandoned and/or cancelled at some point.

Applying a weighted average to Table 1 results in 34% of the projects identified as successful, 50% are challenged, and 16% end up in failure. Thus, about one-third of the surveyed projects end up as a complete success, half the projects fail to some extent, and one sixth end up as complete failures. Considering the role of computers in various industries, such as the airlines and banking, these are alarming numbers.

From a financial perspective,[1] the lost dollar value for U.S. projects in 2002 is estimated at $38 billion with another $17 billion in cost overruns for a total project waste of $55 billion against $255 billion in project spending (The Standish Group, 2003). Dalcher and Genus (2003) estimate the cost for low success rates at $150 billion per year attributable to wastage arising from IT project failures in the Unites States, with an additional $140 billion in the European Union. Irrespective of which estimate is adopted, it is evident that software project mismanagement results in an annual waste of billions of dollars.

Empirical Software Engineering

One of the keys for improving the chances of project development success is the application of empirical-based software engineering. **Empirical-based software engineering** is the process of collecting software metrics and using these metrics as a basis for constructing a model to help in the decision-making process.

Two common types of software metrics are project and product metrics. **Project metrics** refer to the estimated time, money, or resource effort needed in completing a software project. The Standish Group (2003) perceives software cost estimating as the *most effective way to avoid cost and schedule*. Furthermore, several studies (Jones, 1998; The Standish Group, 2003) have shown that by using software cost-estimation techniques, the probability of completing a project successfully doubles. Thus, estimating the schedule, cost, and resources needed for the project is paramount for project success.

Product metrics are metrics extracted from software code and are frequently used for soft-

Table 1. State-of-practice surveys

Year	Successful	Challenged	Failure	Projects Surveyed
United States (Chaos Chronicles III)	34%	51%	15%	13,522
South Africa	43%	35%	22%	1,633
United Kingdom	16%	75%	9%	421