Chapter 3

Optimizing Software Development Cost Estimates using Multi-Objective Particle Swarm Optimization

Tad Gonsalves
Sophia University, Japan

Kei Yamagishi
Sophia University, Japan

Ryo Kawabata
Sophia University, Japan

Kiyoshi Itoh
Sophia University, Japan

ABSTRACT

Software development projects are notorious for being completed behind schedule and over budget and for often failing to meet user requirements. A myriad of cost estimation models have been proposed to predict development costs early in the lifecycle with the hope of managing the project well within time and budget. However, studies have reported rather high error rates of prediction even in the case of the well-established and widely acknowledged models. This study focuses on the improvement and fine-tuning of the COCOMO 81 model. Although this model is based on software development practices that were prevalent in the 80s, its wide use in industry and academia, the simple form of the parametric equations and the availability of the data in an online repository make it attractive as a test-bed for further research in software development cost estimation. In this study, we show how the recently developed Computational Intelligence techniques can effectively improve the prediction power of existing models. In particular, we focus on the adaptation of the Multi-Objective Particle Swarm Optimization (MOPSO) algorithm in simultaneously minimizing two objective functions – prediction error rate and model complexity. This provides the project manager with an opportunity to choose from a set of optimal solutions represented in a trade-off relationship on the Pareto front. Traditional search algorithms use

DOI: 10.4018/978-1-60566-758-4.ch003
knowledge of the terrain and expert heuristics to guide the search; such uses make them problem-specific and domain-dependent. The MOPSO meta-heuristic approach relies neither on the knowledge of the problem nor on the experts’ heuristics, making its application wide and extensive.

**INTRODUCTION**

Every software development project is expected to meet at least the following three key goals. First, the developed software fulfills customers’ needs. Second, the developed software is well engineered so that it allows smooth integration, maintenance, and further development. Third, the software is delivered on time. However, a lot of software is created in a hurry under unrealistic plans. Projects that commit themselves to unrealistic schedules run late and ship unfinished software with bad quality (Jaaksi, 2003). Software development projects are notorious for being completed behind schedule and over budget and for often failing to meet user requirements (Fox & Spencer, 2005). According to recent statistics, roughly 50 percent of all IT projects fail to meet chief executives’ expectations (Wysocki, 1998). The Standish Group reports that, in the year 2000, only 28 percent of all IT application development projects have succeeded, while 23 percent failed (cancelled before completion or never implemented) and 49 percent were challenged (completed but failed to achieve the project goals like cost, time or specifications) (Johnson, 2001). IT projects that go wildly over budget or drag on long past their scheduled completion have been called ‘runaway projects’ (Glass, 1998; Keil, Rai, Mann, & Zhang, 2003).

Delivering a software product on time, within budget, and with an agreed level of quality is a critical concern for software organizations. Underestimating software costs can have detrimental effects on the quality of the delivered software and thus on a company’s business reputation and competitiveness. Overestimation of software cost, on the other hand, can result in missed opportunities to use funds in other projects (Briand, Langley, & Wieczorek, 2000). In response to the industry demand, software development organizations and project managers need accurate estimates during the early stages in the lifecycle of the project to allocate and manage the project resources properly. Expert judgment and algorithmic models have been the two dominant estimation methods practiced through the years. Although expert judgment is difficult to measure and quantify, it is found to be effective in estimating software development costs (Gray & MacDonell, 1997). Algorithmic (or parametric) models attempt to express the relationship between the project development effort and the project characteristics in the form of equations. Software size (measured as kilo lines of code, KLOC) is the most prominent characteristic in the estimation equations. Function points (Albrecht & Gaffney, 1983), SLIM (Putnam, 1978), PRICE-S (Jensen, 1983), COCOMO 81 (Boehm, 1981) and COCOMO II (Boehm et al., 2001) are some of the well-known cost estimation algorithmic models.

The algorithmic as well as the non-algorithmic (based on expert judgment) cost estimation models, however, are not without errors. In the research conducted by the Standish group in 2001 (Standish Group Report, 2001), it is reported that 53% of the US software projects ran over 189% of the original estimate. COCOMO is a well-established and widely acknowledged software cost estimation model. By plugging in the carefully measured cost drivers, one would expect this handy “off the shelf model” to deliver a fairly accurate cost estimate for a new project. Unfortunately, there is considerable evidence that this off the shelf approach is not always successful. Kemerer (1987),