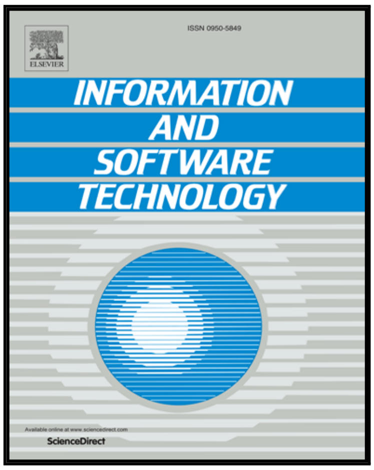
Accepted Manuscript



Duplex Output Software Effort Estimation Model with Self-guided Interpretation

Solomon Mensah , Jacky Keung , Michael Franklin Bosu , Kwabena Ebo Bennin

PII: S0950-5849(16)30402-5
DOI: 10.1016/j.infsof.2017.09.010
Reference: INFSOF 5883

To appear in: *Information and Software Technology*

Received date: 5 December 2016
Revised date: 19 September 2017
Accepted date: 22 September 2017

Please cite this article as: Solomon Mensah , Jacky Keung , Michael Franklin Bosu , Kwabena Ebo Bennin , Duplex Output Software Effort Estimation Model with Self-guided Interpretation, *Information and Software Technology* (2017), doi: 10.1016/j.infsof.2017.09.010

Duplex Output Software Effort Estimation Model with Self-guided Interpretation

Solomon Mensah[a], Jacky Keung[a], Michael Franklin Bosu[b] and Kwabena Ebo Bennin[a]
[a]Department of Computer Science, City University of Hong Kong, Hong Kong, China
[b]Centre for Business, Information Technology and Enterprise Wintec, Hamilton, New Zealand
{smensah2-c, kebennin2-c}@my.cityu.edu.hk, Jacky.Keung@cityu.edu.hk, michael.bosu@wintec.ac.nz

Abstract

Context: Software effort estimation (SEE) plays a key role in predicting the effort needed to complete software development task. However, the *conclusion instability* across learners has affected the implementation of SEE models. This instability can be attributed to the lack of an *effort classification* benchmark that software researchers and practitioners can use to facilitate and interpret prediction results.

Objective: To ameliorate the *conclusion instability* challenge by introducing a classification and self-guided interpretation scheme for SEE.

Method: We first used the density quantile function to discretise the effort recorded in 14 datasets into three classes (*high, low* and *moderate*) and built regression models for these datasets. The results of the regression models were an effort estimate, termed *output 1*, which was then classified into an effort class, termed *output 2.* We refer to the models generated in this study as *duplex output models* as they return two outputs. The introduced *duplex output models* trained with the leave-one-out cross validation and evaluated with MAE, BMMRE and adjusted R^2, can be used to predict both the software effort and the class of software effort estimate. Robust statistical tests (Welch's *t*-test and Kruskal-Wallis *H*-test) were used to examine the statistical significant differences in the models' prediction performances.

Results: We observed the following: (1) the *duplex output models* not only predicted the effort estimates, they also offered a guide to interpreting the effort expended; (2) incorporating the genetic search algorithm into the *duplex output model* allowed the sampling of relevant features for improved prediction accuracy; and (3) ElasticNet, a hybrid regression, provided superior prediction accuracy over the ATLM, the state-of-the-art baseline regression.

Conclusion: The results show that the *duplex output model* provides a self-guided benchmark for interpreting estimated software effort. ElasticNet can also serve as a baseline model for SEE.

Keywords – *Duplex Output; Effort Estimation; Effort Classification; Multiple Regression Models*

1 Introduction

Software effort estimation (SEE) is the process of predicting the effort needed for scheduling, costing and allocating resources to meet project delivery deadlines [1]. According to Menzies et al. [2], accurately predicting the software effort of a *new* project plays a significant role in software engineering, as it minimises overestimation and underestimation [3][4]. Both overestimating and underestimating software effort can result in project cancellation due to budget overruns [2] and can delay funding of project development [5]. In contrast, successful and accurate SEE leads to effective scheduling, monitoring and management of projects [6][7].

Although many prediction models have been developed [5][6][8][9][10][11][12][13][14][15][16][1] to aid in estimating software effort, challenges still exist in relation to the interpretation of the level of effort expended during development [17][7][18]. To the best of our knowledge, there is no benchmark to interpret and validate levels of estimated effort expended in SEE [17][19]. Irrespective of the complexity or simplicity of existing SEE models, the effort estimates mostly contradict the true or actual software effort shown in the validation sets [20]. This phenomenon is the result of the variations within SEE data, the selection criteria for data collection and the training and testing criteria used to set up the SEE models. Currently, there is no optimal SEE model due in part to data uncertainty and to variations within the training and validation sets [9]. The variation in estimation results from different SEE models leads to *conclusion instability* [7][18][21][19]. Thus, a SEE model without a corresponding interpretation guide might not be useful to software engineers. Therefore, there is a need for a more efficient and reliable prediction model that not only estimates the software effort, but also offers a means through which the effort estimates can be effectively applied to specific contexts.

According to Keung et al. [7], the elusiveness of accurate prediction models for software effort has been a challenge for researchers, software engineers, practitioners, project managers and developers, who need an appropriate model and reliable training and validation sets to make relatively effective predictions of SEE [18]. Despite the numerous prediction models [9][6][22], the SEE research community still faces difficulties in making reliable and accurate predictions due to the following factors: (1) variations within SEE datasets, (2) the methodological procedures used for generating the training and validation