TEST Engineering & Management

Jan-Feb 2020
ISSN: 0193-4120 Page No. 17674 - 17683

FGRA BASED SOFTWARE EFFORT AND RELIABILITY ESTIMATION

CHAITHANYA K

Research Scholar, Dept. of Computer Science & Engineering,

Sri Satya Sai University of Technology & Medical Sciences,

Sehore, Bhopal-Indore Road, MadhyaPradesh, India

Dr. Satendra Kurariya

Research Guide, Dept. of Computer Science & Engineering,

Sri Satya Sai University of Technology & Medical Sciences,

Sehore, Bhopal Indore Road,Madhya Pradesh, India

Abstract

Software engineering, estimation assumes an indispensable role in software development. Hence, affecting its expense and required effort and consequently influencing the overall success of software development. Software develops constantly in every single social field. It assumes an increasingly significant role in systems securing, engineering, and development, especially for large and complex systems. For such systems accurate estimates are imperative and considered an essential piece of effective software project management. From a broad perspective, software effort estimation is the process of predicting the required effort to develop or keep up software based on incomplete, uncertain as well as loud data input.

Introduction

Effort estimation required for a software development project is extremely significant for the success of the overall arrangement delivery. Despite this reality, studies show that a huge progress in improving the performance estimation techniques has not been reported, which represents one of the significant challenges inside the software industry. Incorrect effort evaluation often causes budget overrun, delay in delivery, and failure to satisfy legally binding commitments and indirectly affects the quality of the product itself [1]. It is therefore to be expected that a very regular cause of failure of software development projects in the field of information technology (IT) is incorrect effort estimation.

Jan-Feb 2020
ISSN: 0193-4120 Page No. 17674 - 17683

The estimated measure of work directly affects several aspects of the software development process life cycle: it might challenge or backing a decision on the development of software product depending on the investment avocation, it is used as an information parameter for determining the budget of the project and the market price, it affects the project plans, it schedules delivery of project antiques, etc. In practice, estimated effort is often less than the real effort toward the end of the project. Such idealistic estimates favor the prospective user (client) because often "hopeful" estimate justifies lower and, therefore, competitively priced products [2]. However, in such circumstances, the same project team in the product development phase meets many challenges—some of which are the most well-known, already mentioned, exceeding the set budget and delay in delivery. The average deviation or exceeding of the planned measure of work in the research presented in Jorgensen is about 30%. Reasons for the effort overrun are complex and often were not elaborated in detail in previous research on the effort assessment [3].

Then again, an accurate project effort estimate affects the efficient use of existing resources. In cases where the measure of the estimated effort exceeds the measure of the real effort (the planned sum is overpriced), there is a likelihood that human resources are committed to the project to a greater extent than necessary for the execution of the project.

Present trends impose objectives, for example, shorter length of the software development cycle and cheaper product prices. This directly targets lower project effort, yet there is as yet a parallel demand to keep up the agreed quality of the product [4]. To meet set requirements, software development models introduce the practice of defining software modules and their implementation. Those serve as the core arrangement with a likelihood to reuse certain modules in other (separate) software arrangements [5]. Core arrangement beside program code contains a tremendous number of development process antiquities, for example, requirement descriptions, architecture, use cases, test specifications, etc. Reusability practice of software ancient rarities improves the productivity and quality of new software products, while reducing the resources, cost, and time of the future software development projects [6-8].

Software effort estimation

Jan-Feb 2020
ISSN: 0193-4120 Page No. 17674 - 17683

Developing accurate software effort estimation models have placed into the research target for more than four decades. The authentic records of some large software projects have been terrible with respect to time and cost, and the consequences to business issues can be enormously harming. A survey published by the Standish Group CHAOS reported that 16% of 8000 complete project just delivered inside budget and time, while 31% cancelled before completion, and 53% overrun in budget and schedule which resulted in project failure. Mendes et al. showed that some 60% of large projects essentially overrun their estimates (with an error percentage that can fluctuate from 100% to 200%) and 15% of the software projects are never completed due to the gross misestimating of development effort. A recent survey published by Molokken and Jorgensen (2003) revealed that 30-40% of software projects overrun in budget and schedule which shows that this problem is similarly as substantial today as before three decades. However, as there are many components leading to this dilemma, awful and inaccurate software effort estimation is the significant source of such failures. More specifically, underestimation leads to understaffing and consequentially takes longer to deliver project than necessary (Albrecht and Gaffney, 1983). For instance, if manager gives a project more resources than it really requires, the project is then likely to cost more than it should. Interestingly, overestimation may lead to botch chances to offer assets for other projects in future.

Methodology

Software effort estimation by relationship method called FGRA that can possibly overcome the challenges in the former estimation by relationship methods and especially inadequacies in the existing similitude measures. The fundamental thought of likeness is certainly tied in with the process of estimation by relationship method to such an extent that a successful relationship between target project and other source projects is dependent on there being some element of comparability between them.

FGRA effort estimation model as appeared in Figure 1 comprises of three fundamental phases: Data preparation, Attribute selection, and Effort prediction. The processed dataset is then used in the second phase to identify best predictive attributes and their corresponding weights. We should note that the attribute subset selection utilizes the proposed likeness measure so as to

Jan-Feb 2020
ISSN: 0193-4120 Page No. 17674 - 17683

generate similitude grid. The reduced dataset from the second phase is then used in third phase to really perform the effort predictions.

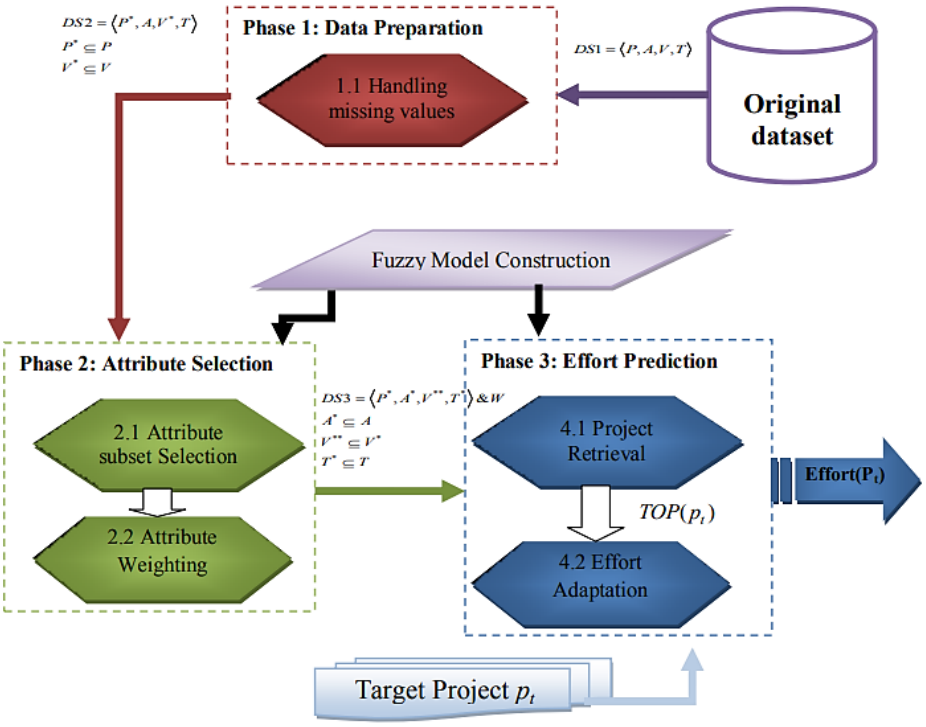

Figure 1. Framework of FGRA software effort estimation model

Data preparation

Large chronicled dataset that is required for building and adjusting software estimation models, is sometimes scarce, incomplete, insufficient, and imprecisely collected. This is often because of inadequate data collection tools and absence of measurement standard. However, utilizing these missing values could lead to serious extent of error in effort estimation. Different approaches have been appeared to handle missing values in Machine Learning, for example, Multinomial Logistic Regression, Mean ascription, List wise deletion, Regression attribution and Expectation Maximization. The generally used technique in software effort estimation is the List wise deletion wherein either the whole project record or the attribute with missing values is ignored and excluded from the dataset. Despite of its straightforwardness, it could lead to lose valuable information and therefore inaccurate estimation. Such methodology gives agreeable results for

Jan-Feb 2020
ISSN: 0193-4120 Page No. 17674 - 17683

taking care of missing values when they are few, and terrible results when missing values increase because it results in little datasets that affect building a substantial cost model.

Attribute selection phase

This phase includes identifying the most predictive attributes and their corresponding weights. As already seen in chapter four, we proposed a new attribute selection algorithm based on Kendall's line wise correlation between closeness network based project attribute and similitude framework based on project effort. Further, we likewise proposed a new attribute weighting technique based on Kendall coefficient of concordance between likeness network based selected attribute and similitude grid based on project effort. However, in chapter four we investigated the efficiency of the proposed approaches on software effort estimation utilizing normalized Euclidean distance. In this chapter we will integrate the proposed algorithms to FGRA model in which the proposed likeness measure is used to build comparability network instead of normalized Euclidean distance. The outcome of this phase is reduced dataset with corresponding attributes and weights.

Effort prediction phase

Once the predictive attributes are identified the similitude degrees between target project (pt) and all verifiable source projects (pi) are assessed along every one of these attributes. In other words, every nearby likeness ($\nabla(aj\ (pi), aj\ (pt)) : aj \times aj \rightarrow [0,1], aj \in A$) are precisely calculated based on type of contributed attributes, and their values are then aggregated utilizing GRA. Once totally aggregated likeness degrees $\Gamma(p_t, p_i)$, $i \in \{1,2,\ldots,n\}$ are computed, the source projects are ranked by their similitude degrees with target project.

So as to predict effort for pt, the top N comparable projects TOP (pt) are selected from P and their corresponding effort values are stored in E(pt). The number of analogies N that will be used to generate prediction of t p must be at first decided. Essentially, there are two perspectives to determine the suitable number of analogies, the primary methodology involves all projects that fall inside a specific comparability threshold β, TOP (pt) $=\{pi \in \mid \Gamma(pt, pi) \geq \beta\}$. This

Jan-Feb 2020
ISSN: 0193-4120 Page No. 17674 - 17683

methodology could ignore some projects which may contribute data when similitude between selected and unselected projects is negligible.

As indicated by Angelis and Stamelos (2000), when size of a dataset is genuinely little it is reasonable to consider just few analogies. Myrveit and Stensrud (1999) and Briand et al. (1999) used just a single similarity to generate prediction. Then again, Schofield (1998), Jeffery et al. (2001) and Mendes et al. (2003b) used one, two and three analogies. The second methodology has been followed in this research where number of analogies is 1, 2 and 3 as suggested by Mendes et al. (2003b) and Shepperd&Schofield (1997), which we believe is sufficient to derive a good estimate.

The second significant issue in effort prediction is the choice of transformation technique that will generate an estimate from a set of Top N comparative project efforts E(t p). Variation is mechanism used to derive a new estimate; in this way, to minimize the differences between retrieved project and current project (Sankar and Simon, 2004). It is significant step in estimation by relationship because it reflects structure of target project on retrieved source projects. We should note that when number of analogies N=1 then the variation technique is entirely not needed. The usually used transformation techniques in software cost estimation are mean of closest analogies when $N \geq 2$, median when $N>2$ (Angelis and Stamelos, 2002), inverse distance weighted mean and inverse position weighted mean (Kadoda et al., 2000) when $N \geq 2$. The inverse ranked weight mean takes the influence of each project into account where the higher ranked similarity has more influence than lower ones. For example, in case we have three analogies, the estimation by inverse ranked would be calculated as (3×Closest case + 2×Second Closest +1×Third Closest)/6.

Result

As the objective of process was defined to predict the real effort for developing the software included in the target dataset. The data was then preprocessed, explored iteratively to meet the base requirements for satisfying the defined objectives.

Jan-Feb 2020
ISSN: 0193-4120 Page No. 17674 - 17683

The real effort for each project was recorded as a constant numerical value, which was then converted into categorical values. Three categories were defined. Each corresponds to the level of the real effort involved in each projects.

The data was then explored and found in need for even more preprocessing. The multipliers were converted into their relevant numerical values based on the mode of the projects and the values given by FGRA model. The values of the multipliers were found not correlated with the prediction of the real effort and therefore, were multiplied to build a new attribute that was called project multipliers. The first individual multipliers were then deleted. Since the projects parameters in reality affect code, the newly constructed multiplier

Attribute was multiplied by the equivalent physical code attribute and the result was transformed utilizing the log work. Figure. 2 shows the density plot for the three effort classes

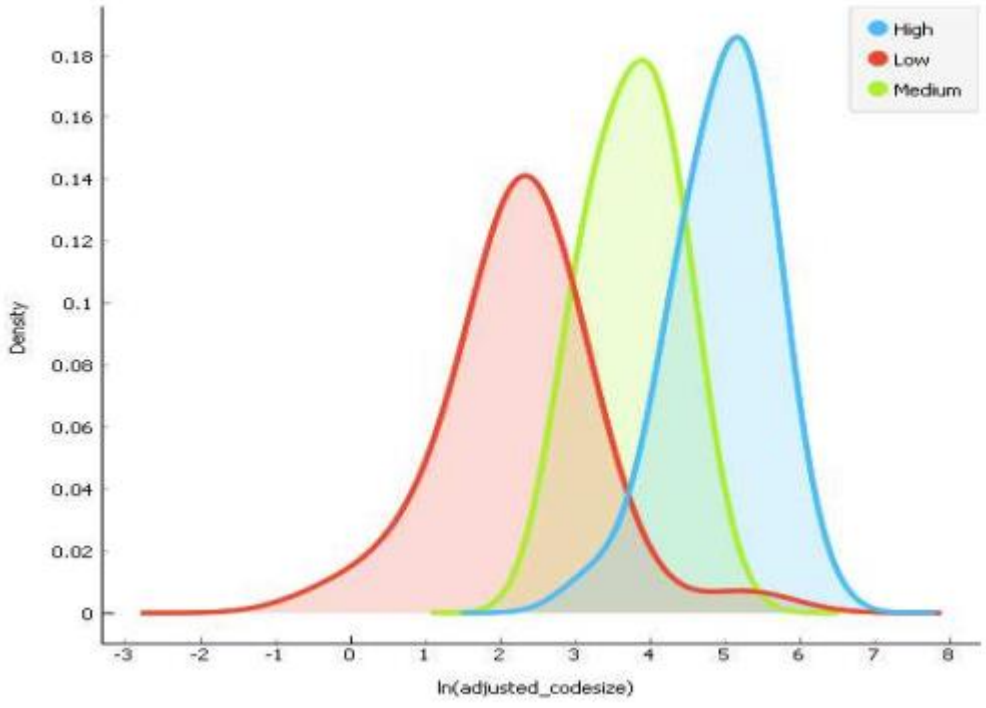

Figure 2. Density Plot

Jan-Feb 2020
ISSN: 0193-4120 Page No. 17674 - 17683

The data exploration likewise involved projecting effort over years, projects and the size of code and its LN change multiplied by the FGRA parameters.

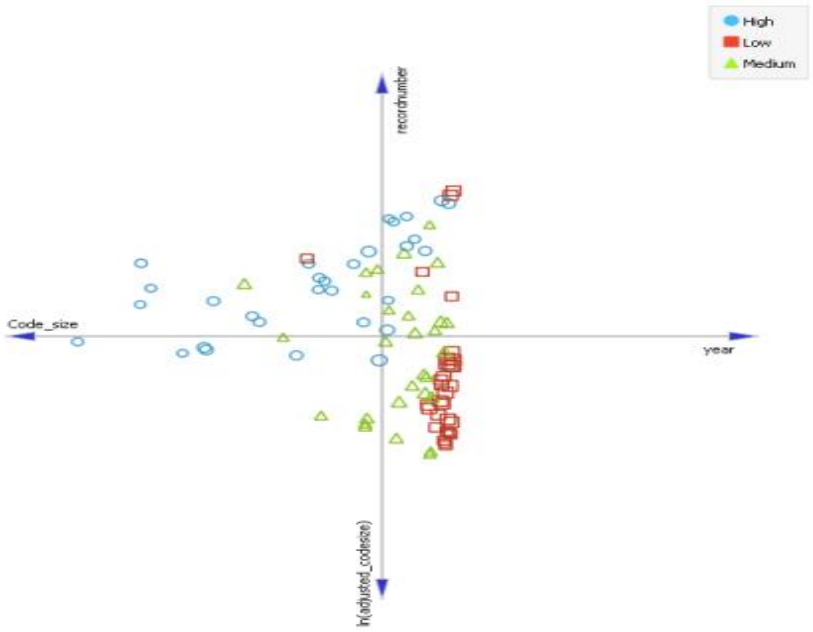

Figure 3.Projection Plot

The result of this projection gives some influence of both size and years on effort increasing while demonstrating little yet an unexpected negative correlation between size and effort. Projection plot in Figure. 3 shows the results of effort projection.

Conclusion

This paper presents new efficient and flexible methodology by joining techniques of Fuzzy set theory and Gray Relational Analysis with existing estimation by analogy method called FGRA. FGRA is additionally integrated with the attribute subset selection algorithm and weighing technique proposed in chapter four to provide a comprehensive procedure that can overcome the primary impediments in former estimation by analogy methods.

Jan-Feb 2020
ISSN: 0193-4120 Page No. 17674 - 17683

References

[1] A. Idri, *et al.*, "Analogy-Based software development effortestimation: A systematic mapping and review," *Information andSoftware Technology*, vol. 58, pp. 206-230, 2015/02/01/ 2015.

[2] J. Keung, "Software Development Cost Estimation UsingAnalogy: A Review," in *2009 Australian Software EngineeringConference*, 2009, pp. 327-336.

[3] P. Suri and P. Ranjan, "Comparative analysis of softwareeffort estimation techniques," 2012.

[4] K. V. Vasantrao, "Understanding of Software effortEstimation at the early Software Development of the life cycle-Aliterature View," *International Journal of Engineering Researchand Applications (IJERA)*, vol. 2, pp. 848-852, 2012.

[5] M. M. Kirmani and A. Wahid, "Use case point method ofsoftware effort estimation: a review," *International Journal ofComputer Applications*, vol. 116, 2015.

[6] F. S. Gharehchopogh and A. Pourali, "A new approachbased on continuous genetic algorithm in software costestimation," *Journal ,of Scientific Research and Development,*vol. 2, pp. 87-94, 2015.

[7] A. Idri, *et al.*, "COCOMO cost model using fuzzy logic," in*7th international conference on fuzzy theory & techniques*, 2000.

[8] A. Kaushik, *et al.*, "COCOMO estimates using neuralnetworks," *International Journal of Intelligent Systems andApplications*, vol. 4, p. 22, 2012.

[9] H. Leung and Z. Fan, "Software Cost Estimation," in*Handbook Of Software Engineering and Knowledge Engineering*.vol. 2, S. K. Chang, Ed., ed: World Scientific Publication, 2002,pp. 307-320.

Jan-Feb 2020
ISSN: 0193-4120 Page No. 17674 - 17683

[10] V. K. Bardsiri and M. Dorosti, "An Improved COCOMObased Model to Estimate the Effort of Software Projects," *Journalof Advances in Computer Engineering and Technology*, vol. 2, pp.11-22, 2016.

[11] "Center for Software Engineering ", "COCOMO II ModelDefinition Manual," University of South California, USA2000.

[12] T. Menzies, *et al.*, "Validation methods for calibratingsoftware effort models," in *Proceedings of the 27th internationalconference on Software engineering*, 2005, pp. 587-595.

[13] A. J. Albrecht and J. E. Gaffney, "Software function, sourcelines of code, and development effort prediction: a softwarescience validation," *IEEE transactions on software engineering*,vol. 6, pp. 639-648, 1983.

[14] H. Raju and Y. Krishnegowda, "Software sizing andproductivity with Function Points," *Lecture Notes on SoftwareEngineering*, vol. 1, p. 204, 2013.

[15] T. Fetcke, *et al.*, "Mapping the OO-Jacobson approach tofunction point analysis," in *Software Metrics*, ed: Springer, 1997,pp. 59-73.