Asian Journal of Information Technology 6 (10): 1010-1014, 2007
ISSN: 1682-3915

Estimating Software Cost from Prototyping, Change Impact Analysis and Experience using XP Methodology

O.O. Olugbara, O.O. Ekabua and M.O. Adigun
Department of Computer Sciences, University of Zululand, South Africa

Abstract: Software Cost Estimation (SCE) is an arduous task in software engineering that has become even more arduous for Business Software Development (BSD) due to changing requirements. The SCE models proposed in literature incorporate software size, approximated by Source Lines of Code (SLoC) or Function Point (FP). However, size estimation for a large portion of software takes time and is as arduous as cost estimation. This study investigates Adaptive Prototyping (Adapro) method that uses probability distribution to estimate software size, given an initial size of its prototype and the number of changing requirements. The method demonstrates the usefulness of Change Impact Analysis (CIA) in SCE. The accuracy of the method was validated with Rayleigh, Exponential and Logistic distributions on real-life projects. The result shows that Logistic and exponential distributions are suitable size estimators when compared to Rayleigh distribution, but Logistic distribution converges faster to the actual software size than exponential distribution.

Key words: Business requirements, extreme programming, change impact analysis, software prototyping, cost estimation, probability distribution

INTRODUCTION

The generation of Business Software Requirements (BSR) has become a very rigorous effort due to unpredictable nature of BSR. The process of BSR elicitation involves the collective effort of both the developers and the customers to achieve a maximum delivery satisfaction. Micro economics reports that about 25% of all software projects are unable to be delivered because of changes in requirements, lack of time and resources. In most cases, where time and cost are estimated for a software project, neglect is always given to changes resulting from addition of requirements. Practical experience shows that addition of requirements often increases effort and elongates delivery time, but this situation is often overlook.

Present experience in using extreme Programming (XP) methodology (Beck, 1999) for BSD shows that customers interest in the project depreciate, resulting to the introduction of incentives to get their maximum cooperation. These incentives result to changes, which have considerable impact on the overall cost of development. XP, unlike most traditional methodologies places a higher value on adaptability than on predictability. The proponents of XP regard ongoing changes to software requirements as a natural norm and a desirable aspect of software development projects. Software requirements must always change as there are always new evolving requirements during the development process. Changing requirements are endemic to software (Davis, 1989) and many researchers have written about software changes and their consequences (Boehm, 1987). Final requirements seldom exist for software systems since they are continually being augmented to accommodate changes in user expectations, operational environment and the like. The ability to adapt to changing requirements at any point during the development cycle is therefore, a more realistic approach than attempting to define all requirements at the beginning of the project and then expending effort to control changes to the requirements.

Estimating the cost of a software project at the begining of contracting affect an effective methodology for CIA. The CIA approach suggests that, since changes must always be made during the development, the impact analysis technique with respect to changes must set lower and upper boundary limits for accommodating the desirable changes. The more the changes, the more the time and effort expended in the project. For effective CIA, appropriate models are required (Zhai, 2002). Hence, a

Corresponding Author: O.O. Ekabua, Faculty of Business, Computing and Information Management, Centre for Systems and Software Engineering, London South Bank University, 103 Borough Road, London SE1 OAA, United Kingdom

balance CIA model should provide limits beyond which if changes are made would result to increase in development cost. CIA further requires that customers and developers need to know about the underlying model and rules. Additionally, CIA recommends that since services are contractual and not freely provided, they must be offered at a relevant level of granularity that combines appropriate flexibility with ease of assembly into the business process. On a summary, CIA proposes an increase cost for out-of-boundary changes. As the out-of-boundary change increases, the cost impact propagates or ripples.

Consequently, the problem of accurately estimating cost is a serious task for software engineers. But, organizations wishing to out-source their software projects would be highly interested in a cost estimate before a commitment is made. The ability to rapidly build a software prototype and derive cost estimate from the size of the prototype will assist the engineers a lot. The objective of this research is development of an adaptive cost method able to incorporate changing requirements.

Related work: The cost estimation techniques for large information systems use SLoC or FP to calculate software effort in person-months (Boehm, 1981). The Constructive Cost Model (COCOMO) (Boehm, 1981; Boehm *et al*., 2000) is a popular cost estimation model based on SLoC. The model supports the estimation of cost, effort and schedule slippage when planning a new project. The relationship between software size, effort and time was presented to be non-linear in Putnam and Myers (2000). Stepwise regression was suggested to be a better prediction model than linear regression (Mendes and Mosley, 2001).

The FP is a very popular model of estimating the cost of software, especially in the early stage of the product cycle (Albrecht, 1979). The Putnums Software Life Cycle Management (SLIM) (Putnam, 1978) is based on the Rayleigh's function and is suitable for large projects. The Albrecht FP model assumes that effort is mainly related to the size of the component to be changed (Niessink and Vliet, 1979). In a different opinion, the size of the change and the size of the component to be changed are equally important (Niessink and Vliet, 1988). An early cost estimation based on requirements alone was proposed in (Mukhopadhyay and Kekre, 1992). The research reported in Henry *et al*. (1996) proposed that the number of requirements changes that occur during maintenance can be used to improve effort estimates.

Artificial Intelligent (AI) techniques have recently found useful applications in software economics. The comparison of machine learning techniques in building effort prediction systems was carried out in Mair *et al*. (2000). An effort has been made to apply fuzzy logic in building software metrics for cost estimation (Gray and McDonell, 1997). Attempts have also been made to evaluate the potential of genetic programming in cost estimation (Burgess and Lafley, 2001).

The maintenance effort for a software application depends on measurable metrics that can be derived from the software development process (Hayes *et al*., 2004). The keen interest in adopting a user-centered agile software development methodology such as XP for BSD has raises new challenges for cost estimation. The requirements for business applications exhibit high degree of unpredictability. These applications involve many stakeholders with divergence opinions and the users of the applications are the primary sources of requirements generation. XP focuses on active involvement of end-users of a business application, user-derived feedback and iterative development whereby prototype software is rapidly developed, tested and modified. The cycle goes on until the final product is delivered to the customer. This methodology can result into a better design and significantly yields a satisfactory product. But, frequent changes coming from the customer can lead to complication and overly delay in the software realization, because as new requirements are discovered during iterations, it is expected that software cost will be affected as well.

The Adapro cost estimation method investigated in this paper uses SLoC for the size of the prototype software developed from the initial requirements. The number of customer influenced changes that occur during maintenance was used to improve cost. This method is different from the estimation methods derived from past projects data (Pressman, 2001). While cost methods based on previous data may be appropriate for off-the-shelve applications, there is a high probability that past data either do not exist or are inaccurate for cost estimation in the case of BSD. Hence, one is faced with the challenge of trying to adapt an existing method to a new environment. Consequently, cost models based on prototyping are of great importance in BSD, which has its peculiarities. Adapro relies on rapid prototyping and mutual consensus between the developers and customers to arrive at satisfactory cost estimation. A distinguishing characteristic of Adapro is the ability to extend an existing cost model to a new development environment by allowing estimation based on changing requirements. Once an accurate size estimate is obtained, an existing cost model can be employed to calculate cost.

THE ADAPTIVE PROTOTYPING COST ESTIMATION

There exists a number of software cost models that can estimate the amount of effort and time required for software development. The estimated effort is converted

Table 1: SCE models

Model name	Effort equatio
Intermediate COCOMO 81	Effort = a (KSLoC)^{b}
Walston-Felix	Effort = 502 (KSLoC)$^{0.91}$
Simplified slim	Effort = 56.4B (1000 KSLoC/P)$^{9/7}$
Doty	Effort = 5.288 (KSLoC)$^{1.047}$

into cost by multiplying it with an average labor rate. There are basically two classes of empirically validated software cost models and they calculate cost as a function of software size. These models include those that use SLoC to estimate software size and those that do not. The main problem with SLoC-based metrics that has led to the development of non-SLoC-based is the difficulty in estimating SLoC. But due to quantitativity and seeming objectivity, they are of interest to researchers. Adaptive cost models able to account for changing requirements as a result of customers involvement in the project is preferable to predictive cost model, because software size can be affected by changing requirements. Table 1 depicts some existing SCE models that use SLoC for cost estimation.

Under the assumption that the impact of changing requirements diminishes as the project advances, the estimation process can be described by a probability distribution. The probability, p(x) as a function of the number of changing requirements, x that the expected size ($Size_p$) of software approximates the observed size (Size) can be given by the following relation.

$$p(x) = \frac{Size_p}{Size} \tag{1}$$

Thus,

$$Size = \frac{Size_p}{p(x)} \tag{2}$$

Let W(t) be the cumulative amount of effort required for estimation in the time interval (0, t). The function W(t) is described by Exponential, Rayleigh and Logistic distributions respectively as follows (Huang *et al*., 1997):

$$W(t) = \alpha\left(1 - e^{-\beta(\theta x + c)}\right) \tag{3}$$

$$W(t) = \alpha\left(1 - e^{\frac{-\beta(\theta x + c)^2}{2}}\right) \tag{4}$$

$$W(t) = \frac{\alpha}{1 + \lambda e^{-\beta(\theta x + c)}} \tag{5}$$

The value β is the scale parameter, α is the amount of estimation effort to be consumed, λ is a constant, c is called the level of change, which is defined as the number of changes that a customer can influenced without incurring extra cost. For a constant θ the variables x and t are assumed to be related as follows.

$$t = \theta x + c \tag{6}$$

The equations used for size estimation were obtained by substituting Eq. 3-5 into 2. Thus:

$$Size(x) = \begin{cases} \dfrac{E_a}{1 - e^{-\beta(\theta x + c)}}, \text{ Exponential estimator} \\ \dfrac{E_\alpha}{1 - e^{\frac{-\beta(\theta x + c)^2}{2}}}, \text{ Rayleigh estimator} \\ E_\alpha\left(1 + \lambda e^{-\beta(\theta x + c)}\right), \text{ Logistic estimator} \end{cases} \tag{7}$$

The value $E_\alpha = Size/\alpha$ is taken as the size of the software prototype. The robustness of the method was improved upon using Least Square Sum (LSS) to fit data. The evaluation formula ψ as a function of parameter set is:

$$\text{Minimize } \psi(S) = \sum_{i=1}^{n}\left[W_i - W(x_i)\right]^2 \tag{8}$$

Apparently, using Logistic distribution for instance, Eq. 8 yields:

$$\text{Minimize } \psi(\lambda_1, \lambda_2 \lambda_3) = \sum_{i=1}^{n}\left[Size_i - E_{ai}(1 - \lambda_1 e^{-(\lambda_2 x_i + \lambda_3 c_i)})\right]^2 \tag{9}$$

Given the observed ($Size_o$) and the actual ($Size_a$) sizes, the accuracy of the method was determined using a Magnitude of Relative Error (MRE) (Conte *et al*., 1986):

$$MRE = \frac{\left|Size_a - Size_o\right|}{Size_a} \tag{10}$$

NUMERICAL ILLUSTRATIONS

To validate the Adapro method with the probability distributions investigated, experiments on different software projects were performed. The result reported in this study is a C++ Builder 6 business application software. The implementation of the project was divided into several of activities such as Graphical User Interface

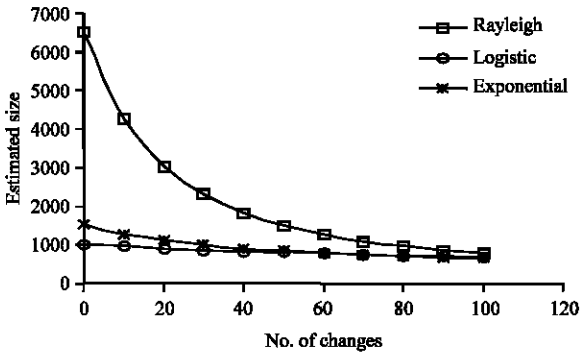

Fig. 1: Estimated size versus number of changes

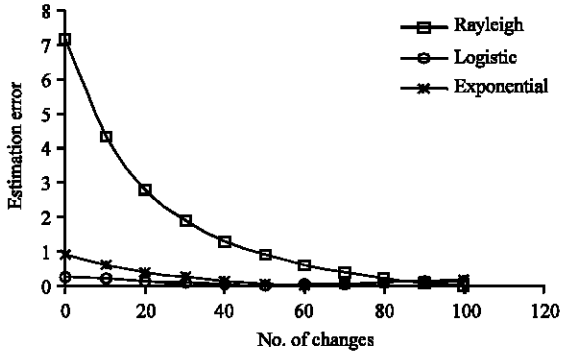

Fig. 2: Estimation error versus number of changes

Table 2: Count of prototype size and changing requirements

Development activity	Size (KSLoC)	Changes
GUI design	473	30
GUI design+DBD	493	49
GUI design+DBD+DBB	501	57
Other activities	-	62

(GUI) design, DataBase Development (DBD), DataBase Binding (DBB), User Report Design (URD), Software Logic Design (SLD), Prototype System Testing (PST), Software Code Refactoring (SCR) and Overall System Testing (OST). Table 2 depicts the parameters of the prototype software whose actual size is approximately 800 KSLoC.

The planning game session of the XP methodology was used to resolve a number of issues including the acceptable value of c. For all projects developed $\beta = 0.01$, $\lambda = 1.5$, $\theta = 1.0$ and for this particular example $c = 40$. It was discovered that large number of changes were made during GUI designed followed by DBD. During the development, different data forms were presented by the customers to minimize the number of changes. The simplicity of use of the application was a top priority concerned of the customer and this led to more changes being influenced during GUI design. The customer influenced changes were very minimal during SLD and most changes were actually influenced by the developers during this activity.

Figure 1 shows the plot of estimated/observed software size against the number of changing requirements. Figure 2 shows the corresponding plot of error against the number of changing requirements. From the figures, Logistic and Exponential distributions give good estimates of software size than Rayleigh distribution, but Logistic distribution converges faster to the actual software size than Exponential distribution.

CONCLUSION

In this study, we have investigated a method for estimating software size and error to perform adaptive maintenance of business applications based on change impact technique. We applied the method to several real-life projects, which produced acceptable cost estimate consensus between the developers and the customers of the business applications. Finally, both Logistic and exponential distributions are suitable for prototyping business requirements when compared to Rayleigh. Conclusively, estimating software size from prototyping can be an effective approach for cost estimation in a business software project.

REFERENCES

Albrecht, A.J., 1979. Measuring Application Development Productivity. Proceedings SHARE/GUIDE IBM Applications Development Symposium, pp: 14-17.

Beck, K., 1999. Embracing change with extreme programming. IEEE. Comput., 32: 70-77.

Boehm, B., 1987. Improving Software Productivity. IEEE. Comput., pp: 43-57.

Boehm, B.W., 1981. Software Engineering Economics. Prentice Hall, New York.

Boehm, B., E. Horouitz, R. Madachy, D. Reifer, B.K. Clark, B. Steece, A.W. Brown, S. Chulani and C. Abts, 2000. Software Cost Estimation with COCOMO II. Prentice Hall, New York.

Burgess, C.J. and M. Lefley, 2001. Can Genetic Programming Improve Software Effort Estimation? A Comparative Evaluation. Inform. Software Technol., 43: 863-873.

Conte, S., H. Dunsmore and V. Shen, 1986. Software Engineering Metrics and Models. Benjamin/ Cummings.

Davis, A., 1989. Software Requirements: Analysis and Specification. Prentice-Hall, New Jersey.

Gray, A. and S.G. MacDonell, 1997. Applications of Fuzzy Logic to Software Metric Models for Development Effort Estimation. Proceedings of the 1997 Annual Meeting of the North American Fuzzy Information Processing Society NAFIPS., pp: 394-399.

Hayes, H.J., S.C. Patel and L. Zhao, 2004. A Metric-Based Software Maintenance Effort Model. IEEE Proceedings of the 8th European Conference on Software Maintenance and Reengineering.

Henry, J., R. Blasewitz and D. Kettinger, 1996. Defining and implementing a measurement-based software maintenance process. Software Maintenance: Res. Practice, 8: 79-100.

Huang, C., S. Kuo and I. Chen, 1997. Analysis of a software reliability growth model with logistic testing-effort function. Proceedings of the Eighth International Symposium on Software Reliability Engineering, 2: 378-388.

Mair, C., G. Kadoda, M. Lefley, K. Phalp, C. Schofield, M. Sheppard and S. Webster, 2000. An investigation of machine learning based prediction systems. J. Sys. Software, 53: 23-29.

Mendes, E. and N. Mosley, 2001. Comparing effort prediction models for web design and authoring using boxplots. Proceedings of the 24th Australasian Conference on Computer Science, Gold Coast, Queensland, Australia, pp: 125-133.

Mukhopadhyay, T. and S. Kekre, 1992. Software effort models for early estimation of process control applications. IEEE. Trans. Software Eng., 18: 915-924.

Niessink, F. and H.V. Vliet, 1997. Predicting Maintenance Effort with Function Points. International Conference on Software Maintenance.

Niessink, F. and H.V. Vliet, 1988. Two case studies in measuring software maintenance. Proceedings of the International Conference on Software Maintenance, pp: 76-85.

Pressman, R.S., 2001. Software engineering. A Practitioner's Approach, McGraw-Hill.

Putnam, L.A., 1978. General empirical solution to the macro software sizing and estimating problem. IEEE. Trans. Software Eng., 4: 345-361.

Putnam, L.H. and W. Myers, 2000. What We Have Learned. Crosstalk.

Zhao, J., 2002. Change impact analysis for aspect oriented software evolution. Proceeding 5th International Workshop on Principles of Software Engineering, Orlando, Florida.