International Journal of Technical Research and Applications e-ISSN: 2320-8163,
www.ijtra.com Volume 2, Issue 6 (Nov-Dec 2014), PP. 65-68

DEVELOPMENT OF A SOFTWARE MAINTENANCE COST ESTIMATION MODEL: 4TH GL PERSPECTIVE

Mohammad Islam[1], Dr. Vinodani Katiyar[2]
[1]Research Scholar, Shri Venkateshwara University, Gajraula, UP, India
[2]Professor, SRM, University, Lucknow, UP, India
[1]mohdislam3@gmail.com, [2]drvinodani@gmail.com

ABSTRACT- The software industry has had significant progress in recent years. The entire life of software includes two phases: Production and Maintenance. Software maintenance cost is increasingly growing and estimates showed that about 90%, if software life cost is related to its maintenance phase. Extraction and considering the factors affecting software maintenance cost help to estimate the cost and reduce it by controlling the factors. Cost estimation of maintenance phase is necessary to predict the reliability, improve the productivity, project planning, controlling and adaptability of the software. Though there are various models to estimate the maintenance cost of traditional software like COCOMO, SLIM, Function Point etc., but till now there is no such model to estimate the maintenance cost using fourth generation language environment. Software maintenance will continue to exist in the fourth generation environment, as systems will still be required to evolve. In this kind of situation there is needed to develop a model to estimate the maintenance cost using fourth generation environment. We propose a systematic approach and development for software maintenance cost estimation model using fourth generation language environment on the basis of COCOMO II. This model is based on three parameters: SMCE with Fourth Generation Language Environment, ACT (Annual Change Traffic), Technical and Non-Technical factors which affect the maintenance cost. The favorable results closely matching and it can be achieved by using model implementation.

Keywords- Cost Estimation, Software Maintenance, Fourth Generation Language Environment, Cost affecting factors (Technical and Non-Technical), Improved Cost Estimation model, ACT (Annual Change Traffic).

I. INTRODUCTION

A. Cost Estimation

In recent years software has become the most expensive exponent of computer system projects. The bulk of the cost of software development is due to the human effort and most maintenance cost estimation methods focus on this aspect and give estimates in terms of Person-Months. Accurate software cost estimates are critical to both development and customers. They can be used for generating request for proposals, contract negotiations, scheduling, monitoring and control. Cost estimation is an imprecise science, as there are many variables such as human, technical, environmental and political which can affect the ultimate costs of software and the resources required to maintain it. Some of the factors appear more obvious than other. To fully estimate software maintenance costs these factors need to be identified and weights assigned to them.

Underestimating the cost may result in management approving proposed systems that then exceed their budgets with underdeveloped functions and poor quality and failure to complete on time. Overestimating may result in too many resources committed to the project or during contract bidding which can lead to loss of jobs. Accurate cost estimation is important because:

- Projects can be easier to manage and control when resources are better matched to real needs.
- It can help to classify and prioritize development projects with respect to an overall business plan.
- Customers expect actual development costs to be in line with estimated costs.
- It can be used to determine what resources to commit to the project and how well these resources will be used.

B. Software Maintenance

Software maintenance is an important activity in software engineering. Over the decades, software maintenance costs have been continually reported to account for a large majority of software costs. Software maintenance is defined as the process of changing, modifying, updating, repairing or existing operational software, but leaving its primary functions intact (Boehm, 1981, pp.54-55). This definition excludes major enhancements (Boehm, 1981, pp.534-535) and hence differs from Swanson's typology (Swanson and Chapin, 1995). In other words the maintenance is about actions taken when a product does not function properly. Software maintenance workload is very large; although in different applications of its maintenance cost vary widely but averagely, the maintenance cost of large software development costs as high as 4 times. Several surveys indicate that software maintenance consumes 60% to 80% of the total life cost; these surveys also report that maintenance costs are largely due to enhancements (often 75% - 80%), rather than corrections (Canfora-2000). The relative cost for maintaining software and managing its total cost (Koskinen-2010). International Electro-technical Commission activities in accordance with the needs of software maintenance, it will be divided into the following five categories: (a) Repair of maintenance (b) Preventive maintenance (c) Integrity of maintenance (d) Adaptability maintenance and (e) Evolution of maintenance.

C. Fourth GL Environment

The term fourth generation language refers to a class of data processing language developed in the mid 1970's that offer simplified expressions for common data processing tasks. These languages allow for systems development in significantly less time than with third generation language.

Fourth generation languages were developed to make life easier for the application programmer. With most fourth generation languages there are a set of predefined defaults which the compiler or interpreter uses to make assumptions about the user's needs. One of the advantages of fourth generation languages is that it allows parts to be rewritten more quickly than with a third generation language.

Martin [MARTIN83] has said that a characteristic of a fourth generation language is that an analyst can obtain results faster than he could write specifications for a programmer. The

analyst then works hand in hand with the user, creating what the user asks for and refining it in a step by step fashion to adapt it better to the user's needs.

In the United Kingdom 45% of installations were making significant use of fourth generation languages in 1986, and a further 30% were planning to introduce them in the near future [IDPM86], so in a few years software maintenance using fourth generation language is likely to be a major factor to many companies.

Although the term fourth generation language is in common use, they consist of a range of products, and it would be more applicable. Fourth generation languages are not just one type of tool, they consist of a wide range of products, and to enable them to be compared, it is necessary to classify them into categories. A report by the Institute of Data Processing managers [IDPM86] produced a list of 4 classifications of fourth generation languages and this is produced below:

(a) Application builders (b) Transaction processing builders (c) Management information systems (d) End user products.

II. RELATED RESEARCH WORKS

Software cost estimation has attracted tremendous attention from the software engineering research community. A number of studies have been published to address cost estimation related problems, such as software sizing, software productivity factors, cost estimation models for software development and maintenance.

- Mr. Boehm studied the various cost factors in the simple or complex public system. The results of his research are published in details in the book (Software Architectures: Critical Success Factors and Cost Drivers). Many researchers focused on models and different methods of cost estimation, but what is important is to update and review each model factors. These models include analog models such as the Delphi method or estimations based on professional experience, model such as analysis of performance indicators and models of machine learning algorithms including neural network, genetic programming, fuzzy logic and many other models.

- Henry Raymond (2010) in a study used the estimation techniques along with the knowledge of the project team, project manager and the president to design a predictive model for estimating the software. This model suggests that the maintenance plays an important role in the success of IT projects. Though the effective use of technology for estimating the time and cost is necessary but is not sufficient. To predict the exact time and cost, the management needs the knowledge, knowledge integration and sharing it.

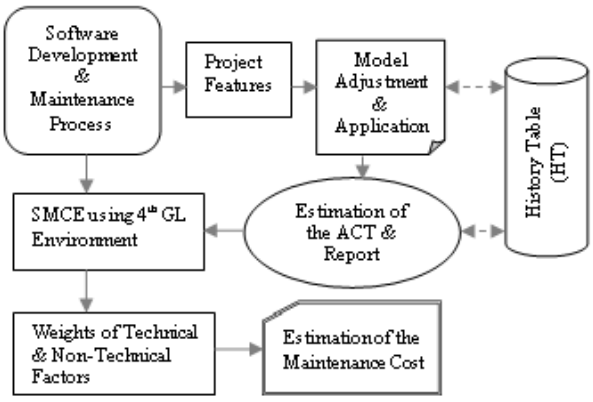
- The studies by Sneed and Jorgensen (2009) provided us with a sound basis for our approach by identifying: (a) the kind of factors being critical to the success of a maintenance operation and (b) evaluating the precision of different types of arithmetic models. Nevertheless, most models and approaches proposed were either not easily generalizes due to highly specialized scenarios, too abstract to implement or not meeting our requirements. In particular, the challenge to predict maintenance costs for a huge number of heterogeneous applications turns out to differ much from estimating maintenance cost benefit for a single application or a single system being under development.

- Many estimation models have been proposed and applied over the years. The review of major estimation models that have been development, continued to be applied and marketed by respective developers, these models including SLIM, SEER-SEM, PRICE-S, Knowledge Plan and COCOMO. There are several reasons for this selection: First, they represent the core set of models that was development in the early 1980's and 1990's. Second, they are still being investigated and used widely in practice and literature. There long robustness and usefulness. Third, these models perform estimation for a broad range of software development and maintenance activities, covering a number of phases of software lifecycle such as requirements, architecture, implementation, testing and maintenance.

- Although the area of software maintenance estimation has received less attention as compared to that of new development, given the important of software maintenance, a number of models have been introduced and applied to estimating the maintenance costs. These models address diverse sets of software maintenance work, covering, error corrections, functional enhancements, technical renovations and reengineering. They can be roughly classified into three types based on the granularity level of the estimation focus: Phase, Release and Task Level maintenance estimation models.

III. PROPOSED SOFTWARE MAINTENANCE COST ESTIMATION MODEL

COCOMO (Constructive Cost Model) is used as a base model to estimate the cost of software project. This model was developed by Barry W. Boehm and published in 1981 using data collected from 63 projects. We proposed a systematic approach for software maintenance cost estimation model using fourth generation language environment on the basis of COCOMO II. This model is based on three specific parameters: SMCE with Fourth Generation Language Environment, ACT (Annual Change Traffic), Technical and Non-Technical factors which affect the maintenance cost. The model which we have proposed is shown below:



(Implementation of Modeling Process)

This model proposes the approach to estimate maintenance cost of software. This process can be processed in two ways. Firstly if the work is of only maintenance then works on Software Maintenance Cost Estimation (SMCE) with 4-GL Environment. Then depending on technical and non technical

factors we will be able to estimate the cost of maintenance of the software. If we are familiar with the development process of the subject software then flow of model will include its Project Features. Project feature includes selected model adjustment and application with its characteristic; Annual Change Traffic could be estimated using the History Table which includes the database. Maintenance cost includes the software maintenance cost estimation using fourth generation languages environment with the help of weights factors. Maintenance cost could be estimated using the result of ACT report, weights of technical and non-technical factors and development cost.

To estimates the software maintenance cost, there are three main parameters used:

(a) Software Maintenance Cost Estimation with Fourth Generation Language Environment.
(b) Existing Technical and Non-Technical Factors.
(c) ACT (Annual Change Traffic).

A. Software Maintenance Cost using Forth Generation Languages.

Software maintenance is consuming vast quantities of data processing resources which have meant that new software cannot be produced quickly enough. One solution to this problem has been the use of fourth generation languages which allow software to be developed more quickly than would otherwise be the case. This change has led to an increase in the amount of software to be maintained. Grindley [IDPM86] reported that some companies with experience of fourth generation languages found it economically sensible to consider rewriting their systems rather than maintaining and patching existing software. There are several types of effect which this move to fourth generation languages can have on software maintenance:

- Simple hidden errors can be avoided, a fourth generation language can deal with certain aspects of the system automatically, and for example it can determine the first and last records.
- Many fourth generation languages are linked to data management systems with built in data dictionaries. The programmer cannot misrepresent the data or fail to declare variables.
- Many fourth generation languages are self documenting. Poor documentation is likely to be a cause of maintenance difficulties with third generation languages.
- Fourth generation language make the understandability of a program clearer, and therefore easier for maintenance by the third person.
- Many fourth generation languages disallow ill-structured program constructs which can cause trouble later.

B. Factors Affecting Software Maintenance Cost.

Software maintenance costs affect the main factors.

Technical Factors

- **Maintenance Staff Ability:** Maintenance is a highly human intensive activity. It requires a lot of training to make new people adept in maintenance task of a software product or service. The maintenance effort and cost effort increase substantially if the team members are shuffled across groups very often or if they keep leaving their jobs quite frequently. Also maintenance staff ability in terms of maintenance experience can have significant impact on the maintenance activity.
- **Internal Complexity:** It defines how much the internal working of component or system is complex. **A** weight value 0 means easy- to- integrate components are available that has very simple interface as well as simple implementation.
- **Documentation Quality.** If the documentation is poor or system code or design is poorly documented, then it will be very costly to find and correct any errors that are present in the system. This observation points towards the fact that documentation quality also has a serious effect on maintenance effort.
- **Testing Quality.** As the experience of software engineering has shown that the number of errors can be significantly reduced by applying an effective testing strategy. With reduced errors, maintenance effort can be quite low. So, better testing quality reduces maintenance effort.
- **System Life Span.** A system with longer lifespan requires more maintenance efforts than a system with shorter one. Many small scale faults in the system can be ignored if the lifespan of the system is short (a few months for example). However, even these small shortcomings can cause a lot of damage for a system in a long run if it has a longer life span (a few years at least). The system life span can have significant impact on degree of hardware dependability with respect to application type.
- **Code Quality.** Locating faults in an unstructured code or the code that does not implement the guiding principles of its architecture, is very difficult. This ultimately affects software maintenance effort. However, as we are concern with architecture based software maintenance, so we will ignore this factor.
- **Application Type.** Application type represents different application areas. Each application is characterized by special attributes as given in their work.
- **Interface Complexity:** How much complex is interfacing of the components? If interface complexity is high then the maintenance cost of Component Based Software will be high.
- **CASE Tools:** CASE (Computer Aided Software Engineering) tools are software programs that are designe**d** to assist human programmers with the complexity of the processes and the artifacts of software engineering. CASE stands for a large number of applications reaching from simple editing tools to environments supporting the whole life cycle. Table 1 is shown below**:**

C. Non-Technical Factors

- **Understandability.** When programmers try to perform some maintenance of a system developed by other programmers, the difficulty of understanding the system limits maintenance. Therefore, it is important that the maintainer gains a complete understanding of the structure, behavior and functionality of the system being maintained.
- **Probability.** During the lifetime of the system, each scenario will have certain likelihood of occurrence. Therefore, each scenario is assigned a probabilistic importance
- **Technology Newness:** A potential cause to software maintenance risk is the newness of the technology being implemented. It is also affected by its volatility, which implies the frequency with which it keeps changing.
- **Organization Maturity:** Organization maturity in terms of its quality certification and/or CMM level defines a minimum guarantee level for quality of development process. A highly placed organization is assumed to have quality skilled staff, defined and repeatable processes along with procedures for defect prevention and continuous improvement. This surely reduces the dependability on

experts, reduced effort requirement for quality software development.

D.Estimation of ACT (Annual Change Traffic):

In a survey of 63 products in various application areas, Boehm [B0EHM81] developed a formula for estimating software maintenance costs. The estimation is calculated in terms of the Annual Change Traffic (ACT), defined as "The fraction of a software product's source instructions which undergo change during a (typical) year, either through addition or modification". The ACT quantity is used, in conjunction with the actual or estimated development effort in person months, to derive the annual effort for software maintenance. ACT is another parameter that is used to estimate the maintenance cost. It includes the proportion of original instruction that undergo a change during a year by addition or modification, if ACT is given. For estimating the ACT of future software project we start with the existence of a series of given characteristics of a software project. The characteristics must be believed to important influences upon ACT. The characteristics should be evaluated and revised periodically by the company's experts to identify critical characteristics of the projects covered in the History Table (HT). Based upon the data in the HT, each characteristic will be assigned a weight pj which permits us to give appropriate recognition to every characteristic based upon accumulated ACT data. Each project will only have two possibilities for every characteristic that is, to have it or not.

IV. CONCLUSION

Software maintenance cost estimation models helps to improve the impact of maintenance cost factor. Barry W. Boehm's in-depth analysis of 17 factors derived from the actual use of the software based on the specific situation of the affected software, the cost of maintenance, the weight factors and work to make proper adjustments software maintenance cost estimation closer to the actual value. We propose a systematic approach for software maintenance cost estimation model using fourth generation language environment on the basis of COCOMO II. This model is based on three parameters which are given. Favorable results can be achieved by using model implementation. In this paper the problems of estimating the cost of the maintenance process have been solved with our model using fourth generation language environment and data collected from previous projects based on COCOMO II model. By applying the proposed model and procedures to these historical data, control over current and future maintainability can be improved and thereby unnecessary and unproductive maintenance costs can be avoided. This model is applicable for providing the accurate estimates, improving the adaptability of software, developing and understanding between the user, customer and third party. As part of the future work, we will improve our cost estimation model by extending the data pool to cover additional applications. The software estimation model has to be recalibrated and extended to more reflect more closely the software development and maintenance practice. The future perspective this model can be enhanced to calculate the maintenance cost of large data projects.

REFERENCES

[1] Byoung-Chol Lee and Sung Yul Rhew "An Empirical Study on Adjustment Factors to Estimate Maintenance Cost of Applications Developed Using Components", SoongSil University, Seoul, 156-030, Republic of Korea, *Lecture Notes on Software Engineering, Vol. 2, No. 1, February 2014.*

[2] Marounek P. "Simplified Approach to effort estimation in software maintenance", University of economic, Prague, Faculty of information and statistics, Journal of systems integration, 2012: 51-63.

[3] Marounek P. "SW Support and maintenance: Extension of ontology about COE concept", simplification of effort estimation, thesis, Prague, VSE-FIS, 2012.

[4] T. Wijayasiriwardhane, R. Lai, K. C. Kang, "Effort Estimation of Component based software development", a survey IET Software, vol. 5, pp. 216-228, 2011.

[5] Roger S. Pressman, Software Engineering: A Practitioner's Approach Seventh Edition, McGraw-Hill Higher Education, 2010.

[6] Deutsche Post DHL, "Deutsche Post DHL investors' MAIL-facts and figures", 2010. http://investors.dpdhl.de/reports/2010/factbook/the-segments/mail-facts-figures.html

[7] Nguyen Vu. "Improved Size and Effort Estimation Models for Software Maintenance", University of Southern California, 2010. http://csse.usc.edu/csse/TECHRPTS/PhD_Dissertations/files/Nguyen_Dissertation.pdf

[8] Nguyen V., Boehm B.W., Danphitsanuphan P. (2010), "A Controlled Experiment in Assessing and Estimating Software Maintenance Tasks", APSEC Special Issue, Information and Software Technology Journal, 2010.

[9] V. Nguyen, B. Boehm, and P. Danphitsanuphan, "Assessing and estimating corrective, enhancive and reductive maintenance tasks: A controlled experiment", IEEE, 2009, pp. 381-388.

[10] Kitchenham, B.A. and Mendes, E. (2009). "Why comparative effort prediction studies may be invalid." In *Proceedings of the 5th international Conference on Predictor Models in Software Engineering*, pp. 1-5.

[11] Shukla, R and Misra, A. K. 2009. AI Based Framework for Dynamic Modeling of Software Maintenance Effort Estimation, Proceedings of International Conference on Computer and Automation Engineering, 313-317.

[12] Ren YC, "Research on Software Cost Estimation and its Expert System", Doctor's degree of Liaoning Technical University, 2008.

[13] Shen J. "Development of a Software Effort and Cost Estimation Tool Based on EFMSEC", Faculty of Graduate Studies, University of Calgary, 2008.

[14] Boehm B.W., Valerdi R. (2008), "Achievements and Challenges in COCOMO-Based Software Resource Estimation," IEEE Software, pp. 74-83, September/October.

[15] Nguyen V., Steece B., Boehm B.W. (2008), "A constrained regression technique for COCOMO calibration", Proceedings of the 2nd ACM-IEEE international symposium on Empirical software engineering and measurement (ESEM), pp.213-222.

[16] Shukla, R. and Misra, A. K. 2008. Estimating software maintenance effort - A neural network approach, Proceedings of the 1st India Software Engineering Conference - ISEC, Hyderabad, India, 107-112.

[17] Li SM, He M, Yang D, et al., "Software Cost Estimation Method and Application", Journal of Software, vol. 18, no. 4, pp. 775-795, 2007.

[18] Boehm BW. Software Architectures: "Critical Success Factors and Cost Drivers", IEEE transactions on Software Engineering, 2007: 965-971.

[19] Bhatt, P., Shroff, G., Anantram, C. and Misra, A. K. 2006. An influence model for factors in outsourced software maintenance, J. Software Maintenance and Evaluation: Res. and Practice, 18, 385-423.