Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X

IJCSMC, Vol. 3, Issue. 6, June 2014, pg.565 – 572

RESEARCH ARTICLE

Enhancement in COCOMO Model Using Function Point Analysis to Increase Effort Estimation

[1]Komal Garg, [2]Paramjeet Kaur, [3]Shalini Kapoor, [4]Shilpa Narula

[1]M.Tech (Scholar), Department of CSE, GNI Mullana, Ambala, INDIA

[2]Assistant Professor, Department of CSE, GNI Mullana, Ambala, INDIA

[3]Assistant Professor, Department of CSE, GNI Mullana, Ambala, INDIA

[4]Assistant Professor, Department of CSE, GNI Mullana, Ambala, INDIA

komal.garg160@gmail.com [1], bsg.152003@gmail.com [2],

fairy.shalini@gmail.com [3], shrieya04@gmail.com [4]

Abstract: COCOMO is constructive cost model, considered as the most accurate model for effort estimation. Function point is another method to calculate the efforts. Effort estimation is a method to estimate efforts before the development of software. There are many models that measure the efforts accurately. My proposed work is to estimate the efforts using hybrid technique of previous models. The generated result will estimate closer efforts than COCOMO. In this paper, we have presented effort estimation models, COCOMO, Dotty model, Baily-Baisily model, Walston-felix model, Halstead and the proposed one. We have taken 20 projects with defined values of cost drivers. The result shows that the hybrid formula provides more accurate results than the other estimation models. Same dataset is used with 15 cost drivers.

Keywords: COCOMO, Function Point Analysis, Cost drivers, Effort estimation

I. INTRODUCTION

Software are distributed among the various system, hence it can works in one or more than none processes. It is also works on the various operating systems. So with the change of time the need of software is also increase. To fulfill this need, the software engineering is required. As software requirement is increases day by day. So it is necessary to maintain the good quality software. To develop good quality software, software engineering is required. For this, the developer's needs to adopt the software engineering concepts, strategies, and practices to avoid the conflicts that are occur during the development process. Software engineering is an approach to develop, maintain and operate the software. The software development plays a crucial role in software engineering. Many specific techniques are required to develop software. The most common thing in development process is the requirement gathering and customer needs. If a developer fails to complete the needs of the customer than he or she may fails to develop good quality software. Software can be said to of good quality, if it is capable to complete the needs of the customer. The customer can be satisfied in terms of quality, cost and design of the system software. Many developers adopt the techniques like systematic and organized approach to develop software [11].A software development process is use to translate the software product, in which the customer translate all the needs to the developers that what kind of changes a customer required. Software engineering is about evolving, building, and maintaining software systems [9]. Software engineering is a set of problem solving skills, expertise, methods and techniques applied upon a multiplicity of domains to develop and produce useful systems that resolve many problems like practical problems [10]. Software engineer is essential to hold software engineering projects which find out, make, make software and tells its performance. A controlled and methodically approach is adopted by software engineers about their work using some techniques and tools depending upon the resources presented and problem to be solved [8]. To estimate the size of the software there are different-different types of models are available. COCOMO 1 and COCOMO 2 are cost estimation models which help to estimate the accurate project size and cost. In this paper we will discuss COCOMO model with function point analysis. In section 2nd we will do literature survey. In section 3rd Function Point Analysis will be discussed. After that Proposed Methodology and experimental results respectively will be discussed.

II. Review of Literature

A. Felfernig and A. Salbrechter (2003) presented about the Knowledge-based configuration which is a promising application of AI approaches in software organizations .The increasing complexity of software products and maintenance of the product has lead to increase in the development cost and the cost of maintenance too. With these circumstances of project complexity, the impressive segregation of effort estimation techniques considering the distinct feature of configuration system development is still a critical issue. They also discuss Function

Point Analysis (FPA) for estimating the size of knowledge-based configuration projects and present a framework for a organization-specific implementation. The application and propagation of Function Point Analysis (FPA) is proven to be accurate for effort estimation in the development of knowledge-based configuration systems. A framework for a business-specific application has been taken which decreases the overall errors as compared to the project errors estimated by other models. Using this approach effort further work will include the analysis of domain-specific complexity [1]. **Parastoo Mohagheghi (2005),** in this paper they try to modify the effort estimation methods that are based on use case points. In this paper authors describe how we can modify the prediction models and what kind of knowledge or information we can get by doing modification in estimation models. They also try to compare the estimated data with the actual data. The other estimation models predict efforts through KLOC, FP and expert judgment. Including use case points all these estimation models are based on UML. UML diagrams helps a lot in effort estimation. They apply UCP as their early experience with it is good. In UCP, there are actors and use case with assigned weights. Similarly, TF and EF have range from 0 to 5. They take data for the release 1 in which 23 use cases are present. Out of those 15 were modified, 1 was new and rest are not modified. They calculate effort by 23*15(WF)*36. In this a) they assume actors as complex and average actors b) they count UUCP c)calculate modified use case point d) assign 1 to technical and environmental factors e) assign UUCP= MUCP. At last they calculate efforts for primary and secondary changes. The result shows the estimated efforts is 17 % lower than the actual efforts [2]. **Shinji Kusumoto et al (2005)** use case point model is one of the most reliable prediction models that are used during requirement capturing phase. From this, it is clear that this activity is performed at early stage of software life cycle model. This model provides traceability to the rest of the models i.e. analysis model, design model, implementation and so on. The main measuring component in UCP is actors, use cases, technical factor and environmental factors. However measuring the complexity of these components are difficult, ambiguous and even unreliable. Therefore, to overcome this problem, an automated tool has been developed i.e. U-EST. This tool helps to determine the actor and use case complexity. In this paper, the authors also compared the result i.e. complexity measured by automated tool and the complexity measured by the experts or manually. The automated tool is implemented in two languages: java and xereces. The input is taken from the xml. They have to use the analyzer in order to extract information about the actor and use cases. In this paper, they also show the architecture of U-EST which shows how input is taken from xml and the analyzer elicit the actors and use cases. The complexity analyzer measures their complexity. At last, from this, UUCP is calculated [3]. **Zhihao Chen, Barry Boehm (2005)** describes Cost prediction is essential in software industry for planning, monitoring, controlling and software risks and time schedule. Estimation models, like COCOMO, can avoid irrelevant resources that are assigned to a project. In this paper, they discuss that effort estimation by COCOMO can be corrected via WRAPPER- a feature subset selection method invented by the community of data mining. Taking data from data sets through PROMISE repository, they show WRAPPER importantly and dramatically improve COCOMO's estimating power. They have shown that, for the

developed data sets, features subset selection always effectively grows mean PRED (30) values without increasing variance. Often, the improvements can be dramatic; e.g. the project2 in this paper. In the future, they are planning to do more satisfactory work to better analyze the implications of FSS on COCOMO, and validate the idea of the extended decreased parameter model. Also, they will try to reuse this experiment with other induction methods such as genetic algorithms. Further, they also want to test it on other data sets. Here, they have used COCOMO-I or COCOO81 dataset since they wanted to define a repeatable software cost estimation experiment.

While the promise repository contains several data sets [4]. **A. Chamundeswari and Chitra Babu (2010)** describes the estimation models plays an important role while estimating efforts and cost of the project. One of the most appropriate techniques for estimating size is the function point analysis. However it is one of the reliable technique but still it is difficult to find or identify the components of function point analysis. In this approach, they proposed an approach which helps to identify and classify the components of FPA. The applicability of new approach is also compared with the existing approach. The main idea behind this to develop a new enhanced approach for predicting the size of OO software using function point analysis methodology. This can be achieved by a) integrating components of object model to the components of FPA b) Knowing the values of parameters that have some kind of dependencies within the class c) By predicting the size of OO software using function point analysis. To predict software size, we have to determine various applications of software. For this, we need to know external and internal boundaries of application. FPA has five components: input, output, inquiry, internal logical files, and external interface files. In this paper, the authors try to identify and classify the ILF and transaction factors. The new approach is tested on a small set of projects and the OMFP is calculated. The estimated results are compared with the result of function point on the same set of projects. This new approach provides a solution to identify components of FPA [11].

III. Function Point Based Estimation

FPA is worldwide accepted by international organization for standards that is used to estimate the system's functional size. The functional sizes show how much functionality is required. It is independent of any technology which used to implement the system [6].

FPA show the size of the project in terms of functional units. So its unit of measurement is function units. An easiest function point is used to predict the number of members involved in the project and the size of the project [3]. It includes five parameters that come under two categories. Input, output and inquiries are the part of data Functionality and Internal logical files and external interface files are the part of transaction functionality. The five parameters are:

Input: This activity is performed by the user. It can be in the form of text, dialog-box, screens and notes which can be created or modified by the end user and other users.

Output: It is an activity that is shown to the end user according to their requirements which includes output screens, images, files etc. It simplifies the complex data and processes it.

Inquiries: When we input something and receive desired output as a result. That output is directly retrieved from the database.

Logical Internal Files: The logical group or data which is controlled by the application. It is a single file or we can say flat file or table file in a relational database**.**

External Interface file: The file is generated by one application and is controlled by another application and provides interaction between both the applications. It includes major logical groups [7].
The functional size may be used:

1. To control the costs and budget the development applications.

2. To plan the costs of the application portfolio under annual maintenance.

3. For cost estimation, determines the size of the software. It is used to find out the productivity after the completion of the project.

IV. Problem Domain

COCOMO model is software cost estimation model of software development. COCOMO starts estimate from the design phase to integration phase of cost and schedule of the project. But separate estimation model should be required for remaining phase. It is not a realistic perfect model because assumptions made at early phase may vary with time. A new estimate may show budget overruns or budget under run for the project. This may lead to partial development of the system.

V. Proposed Methodology

COCOMO model is software cost estimation model of software development. The model use regression formula to estimation cost using historical data with present and future characteristics. COCOMO starts estimate from the design phase to integration phase of cost and schedule of the project. But separate estimation model should be required for remaining phase. It is not a realistic perfect model because assumptions made at early phase may vary with time. A new estimate may show budget overruns or budget under run for the project. This may lead to partial development of the system. However it is considered as the best model among all. A new proposed method is being discovered that gives better result than the COCOMO. In this new method, the way to implement the cost drivers is different from COCOMO but the dataset is

same for both the models. In our proposed work, we are using Intermediate COCOMO model in which extra feature is added i.e. 15 cost drivers. These 15 cost drivers have fixed values which are multiplied to get effort adjustment factor (EAF). The effort estimation for intermediate COCOMO Model [8].

Effort = a*(kloc) ^b*effort adjustment factor

COCOMO2 is the current version in which there are 22 cost drivers (basically 5 scaling factor and 17 cost drivers). The dataset used is NASA in which there are 63 projects which contains value for 15 cost drivers, KLOC and actual effort. In our work, 20 projects were taken from the given dataset.

COCOMO model is based upon the KLOC lines. But complexity increase as the lines of code increase. So it is difficult to estimate the effort with more complexity. Function point estimation model is based upon functional size to estimate efforts. So to reduce complexity of COCOMO model in our proposed work, merge COCOMO and function point model and enhance accuracy with the help of functional size as compare to KLOC in COCOMO model for effort estimation. The tool that we will use to implement my proposed work is MATLAB.

VI. Experimental Results

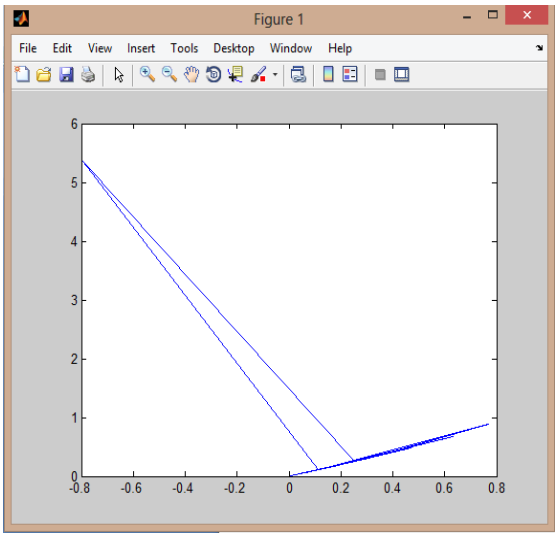

Fig. 1.1 Hybrid Technique

The Hybrid method shows the best result as compared with other models that are being used for effort prediction. So it is a suitable model for effort estimation.

Calculated MMRE of 20 projects for proposed formula

MMRE=0.2641

VII. Application Area

COCOMO is factual and easy to interpret. One can clearly understand how it works.
Accounts for various factors those affects cost of the project. Works on historical data and hence is more predictable and accurate. But there are some disadvantages of COCOMO model that it ignores requirements and all documentation.
It ignores customer skills, cooperation, knowledge and other parameters. It oversimplifies the impact of safety/security aspects. It ignores hardware issues. It ignores personnel turnover levels. It is dependent on the amount of time spent in each phase.

VIII. Conclusion

A software development process is use to translate the software product, in which the customer translate all the needs to the developers that what kind of changes a customer required. A new technique is to estimate the software efforts. The performance of the developed method is tested on NASA software project data and results are compared with many estimation models (Basic cocomo, Intermediate COCOMO, Dotty model, Walston-felix model, Halstead and Baily baisily). The result shows that the proposed method has lowest MMRE than other models. In future the proposed work has wider scope. The Proposed model required to improve accuracy of the existing estimation models. With more accuracy we can produce or estimate better results which are helpful for future use. New model helps to remove problems which occur in existing models and create problem to produce better results. By removing this, enhancement in accuracy is possible.

ACKNOWLEDGMENT

The authors would like to thanks Ms. Paramjeet kaur, Ms. Shalini kapoor, and Ms. Shilpa narula for their extensive help and constant discussions.

References

[1] A. Felfernig, A. Salbrechter (2003), "*Applying Function Point Analysis To Effort Estimation In Configuration Development*", Proc. Of the Sixteenth International Conference on Software Engineering & Knowledge Engineering
[2] Parastoo Mohagheghi (2005), "*Effort Estimation of Use Cases for Incremental Large-Scale Software Development*". Proc. of IEEE international conference on software engineering.
[3] Shinji KUSUMOTO (2004), "*Effort Estimation Tool Based on Use Case Points Method*" Proc. of Software Metrics, 10th International Symposium, Pp 292-299.
[4] Zhihao Chen (2005)," *Feature subset selection can improve software cost estimation accuracy*" Proc. of 2005 workshop on Predictor models in software engineering**.** Pp 1-6

[5] Archana Srivastava1, Dr. Syed Qamar Abbas2, Dr.S.K.Singh3, "Enhancement In *Function Point Analysis*" , Proc of International Journal of Software Engineering & Applications, vol 3, Pp 129-136

[6] Barry Boehm, Chris Abts & Sunita Chulani (2000), " *Software development cost estimation approaches", A survey- Annals of Software Engineering,* 10(1-4), 177-205.

[7] Bente Anda, Hege Dreiem, Dag I.K. Sjøberg and Magne Jorgensen (2002) "*Estimating Software Development Effort based on Use Cases –Experiences from Industry*", Proc. Of 4th International Conference on the Unified Modeling Language, Modeling Languages, Concepts and Tools, pp. 487-502

[8] Bingchiang Jeng, Dowming Yeh, Deron Wang, Shu-Lan Chu And Chia-Mei Chen et al, (2011) *" A Specific Effort Estimation Method Using Function Point*", Proc. Of Journal of Information Science and Engineering, Vol 27.

[9] Manpreet Kaur, Sushil Garg (2012), "*Analysis of Neural Network based Approaches for Software effort Estimation and Comparison with Intermediate COCOMO*", Proc. Of International Journal of Engineering and Innovative Technology (IJEIT), pp. 197-200.

[10]M.Sheppered and C. Schofield (1997), "*Estimating software project effort using analogy*", IEEE transaction, Pp 736-743.

[11] Shinji KUSUMOTO (2004), "*Effort Estimation Tool Based on Use Case Points Method" Proc. of Software Metrics,* 10th International Symposium, Pp 292-299.

