Software Effort Estimation Techniques in the IT Sectors-A Review

M.Senthil Kumar[1] Dr. B. Chidambara Rajan[2]
[1]Assistant Professor (Senior Grade) [2]Professor & Principal
[1,2]Department of Computer Science & Engineering
[1,2]Valliammai Engineering College

Abstract— Software Effort Estimation is an important area of research in software engineering. Various effort estimation models (COCOMO, SLIM, SEER, etc.) have been developed in the past which follows mathematical model for estimation. Out of all these techniques, mathematical model is widely used under which three common techniques are used. These three techniques include use case method, function point method and a derivative of function point method called cosmic function point. Each method has its own implementation methodology, which will be explores all these methods and demonstrates the pros and cons of these methods in the real time software effort estimation. Finally a comparison between all these methods is made .The values of this work is in helping better understand how software engineers make decisions between the models, when they start estimation. This work also enables project managers to elicit software project features based on the models to the client in crystal clear manner.

Key words: Software Effort Estimation, array, Ku- Band

I. INTRODUCTION

The success of any project lies in accurate estimation of cost, effort and size of the project before developing the project. There are several methods exists, in order to estimate a project any point of time during the product lifecycle. Various estimation techniques have been developed in the past which follows mathematical model for estimation. SLIM (Software life cycle model), COCOMO (Constructive Cost Model), SEER (System Evaluation and Estimation of Resources) are some of the model based techniques for software estimation. Projects’ related data is used as input in these techniques and past projects’ data is used for calibrating the models. When past projects’ data is not available then experts’ knowledge is used for estimation. Delphi and Rule-based techniques comes under this category. Delphi technique is based purely on the experts’ judgment whereas rule based technique is adopted from the artificial intelligence domain in which a set of rules work together to get the output i.e. the estimates.

Out of all these techniques, mathematical model is widely used under which 3 common techniques are used. These three techniques include use case method, function point method and a derivative of function point method called cosmic function point. Each method has its own implementation methodology, which will be used to estimate any project. The following sections discuss all these methods along with its advantages and disadvantages. Finally a comparison between all these methods is made.

II. MATERIALS AND METHODS

A. Use Case Method:

The use case points method is a software sizing and estimation method based on use case counts called use case points. The software size (UCP) is calculated based on elements of the system use cases with factoring to account for technical and environmental considerations. The UCP for a project can then be used to calculate the estimated effort for a project[5].

1) Classifying Actors and Use Cases

The total unadjusted actor weights (UAW) is calculated by counting how many actors there are of each kind, multiplying each total by its weighting factor, and adding up the products.

$$UAW = (\text{Total no. of simple actors} * 1) + (\text{Total no. of average actors} * 2) + (\text{Total no. of complex actors} * 3) \quad (1)$$

Similarly, each of the requirements is classified under use cases and each use case is then defined as simple, average or complex, depending on number of transactions in the use case description, including secondary scenarios. A transaction is a set of activities, which is either performed entirely, or not at all. Counting number of transactions can be done by counting the use case step[7].

Once all use cases have been classified as simple, average or complex, the total weight (UUCW) is determine by summing the corresponding weights for each use case.

$$UUCW = (\text{Total No. of Simple Use Cases} \times 5) + (\text{Total No. Average Use Case} \times 10) + (\text{Total No. Complex Use Cases} \times 15) \quad (2)$$

2) Technical and Environmental Factors:

The Technical Complexity Factor (TCF) is calculated by multiplying the value of each factor by its weight and then adding all these numbers to get the sum called the TFactor. The following formula is applied:

$$TCF=0.6+ (0.01*TFactor) \quad (3)$$

The Environmental Factor is calculated by multiplying the value of each factor by its weight and adding the products to get the sum called the EFactor. The following formula is applied.

$$EF=1.4+ (-0.03*EFactor) \quad (4)$$

Finally the UCP can be calculated once the unadjusted project size (UUCW and UAW), technical factor (TCF) and environmental factor (EF) have been determined. The UCP is calculated based on the following formula:

$$UCP= (UUCW+UAW)*TCF*EF \quad (5)$$

B. Function Point Method:

Function point analysis measures size of the software on the basis of the functionalities to be provided by the software. The method quantifies the functionalities of software by the information provided by the user based on logical design. FPA estimates the size of software in terms of function point counts (FPC) which can be converted into SLOC easily if the equivalent SLOC for unit FPC is available [3].

1) Steps for Counting Function Points:

Required information for counting is obtained from the software requirement specification. The steps for counting function points are as following:

1) Identify data functions (External Interface files and Internal Logical Files) and rate them.
2) Identify transaction functions (External Input, External Output and External Inquiry) and determine their complexity.
3) Compute unadjusted function points. Number of EI, EO, EQ, ILF and EIF for each complexity level (Simple, Average and Nominal) is obtained and the corresponding weight for each complexity level is multiplied with the count to finally get the unadjusted function point count. Details of function point count are available in appendices.
4) Determine the ratings of 14 general system characteristics.
5) Calculate value adjustment factor (VAF).

$$VAF = (TDI * 0.01) + 0.65 \quad (6)$$

Where, TDI = Total Degree of Influence obtained by multiplying the ratings of general system characteristics.

1) Calculate correctness factor (CF)

$$CF= (\sum_{i=1}^{8} \text{Value} * \text{Complexity})*0.01$$

2) Compute function point counts.

$$FPC = UFP * VAF+CF \quad (7)$$

C. *Cosmic Function Point Method:*

Cosmic function point is a method of measuring a ‘functional size’ of software. ‘COSMIC’ stands for the Common Software Measurement International Consortium, a grouping of software measurement experts from around the world who, in 1998, saw the need to improve on traditional function point methods. The COSMIC method can be used to provide a measure of the size of a software development project’s work-output that may be used to derive performance measures ‘productivity’ (= size/effort), etc. As the size measure depends only on the required functionality and is independent of any technology used, such measures can be used to compare performance across projects using different technologies [4].

The process for measuring the COSMIC functional size of a piece of software*:*

The COSMIC measurement process has three phases, as shown in the Figure below:

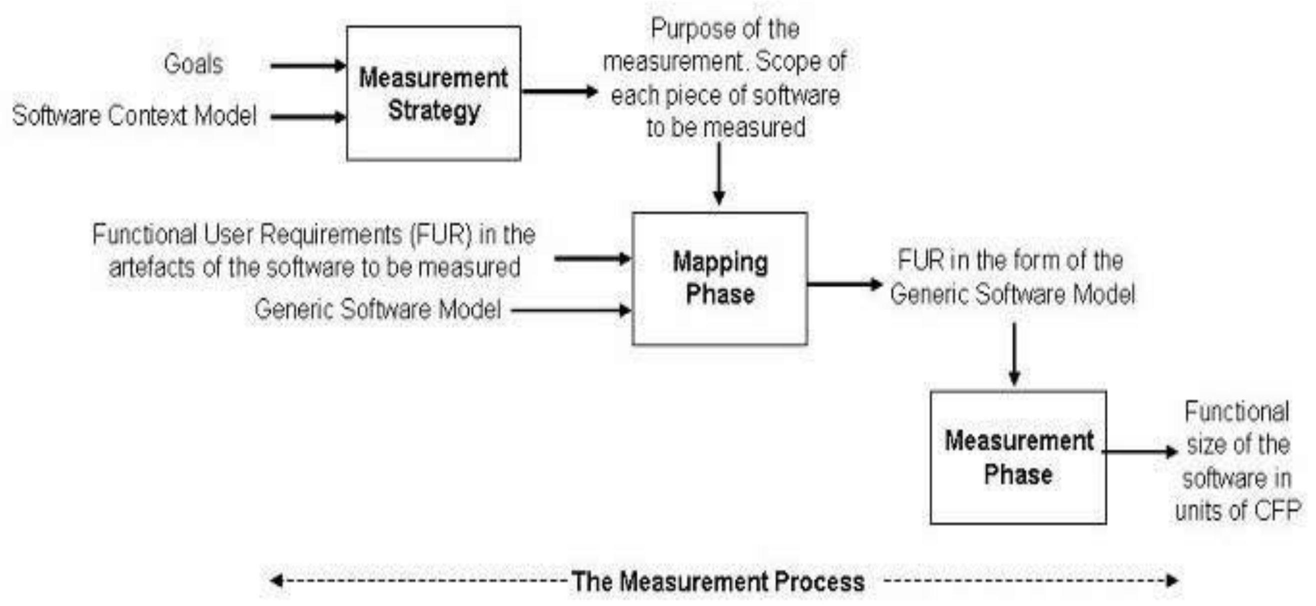
Goals
Software Context Model
Measurement Strategy
Purpose of the measurement. Scope of each piece of software to be measured
Functional User Requirements (FUR) in the artefacts of the software to be measured
Generic Software Model
Mapping Phase
FUR in the form of the Generic Software Model
Measurement Phase
Functional size of the software in units of CFP
The Measurement Process

Fig. 1: Software Context Model

The measurement strategy takes as input, the goals of the project and the Software Context Model. The strategy defines the scope and purpose of the measuring each piece of software. The next process is mapping phase, which takes as input, the functional user requirements in the artifacts of the software to be measured along with the Generic Software Model. The output of this phase will be the generic software model in the name of Functional User Requirement. The third phase called the measurement phase takes this as input and returns the Functional Size of the software in units of Cosmic Function Point [8].

III. Results and Discussion

The survey on previous section discusses three different methods for software effort estimation viz. Use case method, Function point method and COSMIC function point method. This detailed study helped us to derive a comparison among these three methods and discuss the advantages and disadvantages of each of these three methods.

Sl	Title	Author	Year	Method	Concept	Future Enhancement
1	Initial Hybrid Method for Analyzing Software Estimation, Benchmarking and Risk Assessment	J. Frank Vijay and C. Manoharan	2009	Use case point method	Available works on the effort estimation methods were discussed and a hybrid method for effort estimation process was proposed. As an initial approach to hybrid technology, they	Their future work will be to use the effort estimation based on this Hybrid Tool in the hybrid technology proposed for risk assessment and benchmarking. They will also extend this technique for developing an automated tool for assessing risk

	Using Design of Software				developed a simple approach to SEE based on use case model called the "use case point's method".	and benchmarking
2	Effort Estimation Tool Based on Use Case Points Method	Shinji Kusumotoy, Fumikazu Matukaway, Katsuro Inouey	2007	Use case point method	To effective introduction of UCP method, they have developed an automatic use case measurement tool, called U-EST. This paper describes the idea to automatically classify the complexity of actors and use cases from use case model.	The relationship among UCP, function point and actual software development effort will be derived in future and evaluate the usefulness and applicability of the estimation by UCP method.
3	Project Estimation With Use Case Points	Roy K. Clemmons	2006	Use case point method	This article provides an introduction to the Use Case Points method that employs a project's use cases to produce a reasonable estimate of a project's complexity and required man-hours.	The author encourages more projects to use the UCP method to help produce software on time and under budget.

Table 1: Results of the survey with respect to Usease Point Analysis

Sl	Title	Author	Year	Method	Concept	Future Enhancement
1	A Specific Effort Estimation Method Using Function Point	Bingchiang Jeng, Dowming Yeh, Deron wang, Shu-Lan Chu and Chia-Mei Chen	2011	Function point method	This research suggests an approach that simplifies and tailors a generic function point analysis model to increase ease of use. It redefines the function type categories in the FPA model, on the basis of the target application's characteristics and system architecture.	Two research directions exist in this. First, studies could evaluate whether the tailored FPA model always fits a business application domain and still maintains comparable estimation accuracy. Second, the ideas presented herein might be adapted to another estimation model and determine how it performs.
2	Efficient effort estimation system viz. function points and quality assurance coverage	H. Azath and R.S.D. Wahidabanu	2011	Function point method	This study is a basis for the improvement of software effort estimation research through a series of quality attributes along with constructive cost model (COCOMO).	None
3	Estimating Non-functional Properties of Component-based Software Based on	Marcus Meyerhofer, Klaus Meyer-Wegener	2005	Function point method	The COMQUAD project defines system architecture and a development methodology for component-based software with quantitative properties and adaptively, thereby respecting non-functional properties from design to provision at runtime.	Their future work would be in the areas of space and time which is applicable to NFPs as well.

Table 2: Results of the survey with respect to Function Point Analysis

Sl	Title	Author	Year	Method	Concept
1	A comparative study case of Cosmic - FFP, Full Function Point and IFPUG methods	Vinh T. Ho, Alain Abran, Fetcke Thomas	2010	Cosmic Function point method	This study compares the designs of these three measurement methods viz., full function point, Cosmic full function point and IFPUG function point through a common framework, from the software models to the measurement process. A case study on a warehouse software portfolio allows illustrating in detail an empirical comparison on the measurement with these three methods.
2	From Story Points to COSMIC Function Points in Agile Software Development – A Six Sigma perspective	Thomas Fehlmann, Luca Santillo	2010	Cosmic Function point method	This work investigates benefits from adopting a standardized Functional Size Measurement (FSM) method, such as COSMIC Function Points, in place of Story Points. Using a Transfer Function (from the Six Sigma practice) that transforms size into effort spent within a particular agile team, defect density prediction is made using sensitivity analysis.
3	How to use COSMIC Functional Size in Effort Estimation Models?	Cigdem Gencel	2008	Cosmic Function point method	This study brings suggestions on how to use COSMIC functional size as an input for effort estimation models. It explores whether the productivity values for developing each functionality type deviates significantly from a total average productivity value computed from total functional size and effort figures.

Table 3: Results of the survey with respect to Cosmic Function Point Analysis

Method	Advantages	Disadvantages
Use case point method	• Widely used in most of the real time software projects. • Easy to identify actors and use cases from the use case diagram, which was developed by an UML specialist. • Typically applicable during the early stages of software development since it does not require much details as needed by the function point method.	• Support of special tool is required to measure actor weight and use case weight during estimation process. • Does not give much attention to functional and non-functional characteristics of a software system. • It is very rigid for changing requirements at later stages in the software development and hence lacks accuracy.
Function point method	• It considers each of the functional characteristics of a piece of software project in the overall estimation process. • It also considers some non-functional characteristics and calculates the value adjustment factor, which will be included in the total function point. This ultimately improves the overall effort calculated. • In this method, the effort relates linearly to the function points. .	• This method was designed to measure only business software in the application layer. • It is not suitable for projects that have a dynamic requirement changes during project life time. • It cannot be used for estimation at early stages of project lifecycle since it requires complete functional aspects of the project, which may not be available in the client’s initial requirements.
COSMIC function point method	• It was designed to measure business application, real-time, and infrastructure software (such as operating system software) and hybrids of all these types, in any layer of multi-layer software architecture, and components of these at any level of decomposition. • Its basic concepts are compatible with modern methods of determining software requirements and constructing software. • It includes a number of approximate variants of the standard size measurement method which can be used for rapid measurement and/or	• It is not widely used in many real time projects since the metrics used in this method are not standardized. • It includes several approximate variants and hence it may affect the accuracy of the estimated results. • It is not suitable for small scale projects • It doesn’t have any considerations on non-.

Table 4: Comparison of pros and cons of the three methods

IV. CONCLUSION

Effective software effort estimation is very important in the budgeting, project planning & control, trade-off and risk analysis of project management. In this approach, it is aimed at addressing the problem of each mathematical effort estimation models that are used in estimation process. Based on these reviews, the small and new software companies can substantially benefit from the analytical report of this paper. This review suggests that companies must prefer models which are relevant to the in-house project data. In practice, an estimated 15% of software projects are never completed due to the, improper selection of effort estimation models. The above analyses will greatly aid software personnel in choosing the best software economic policy based on cost, effort and efficiency.

Further research will focus on finding the best mathematical model which would be more applicable for small scale real time projects in the market and try to improve the accuracy.

REFERENCES

[1] Anda B,Dreiem H,Sjoberg D.I.K, Jorgensen M, "Estimating software development effort based on use cases-Experiences from industry" in Proceeding of Fourth International Conference on the UML,pp.487-504,2001.

[2] Azath.H and Wahindabanu R.S.D,"Efficient effort estimation system Viz Function Point and Quality Assurance Coverage", IET Software, pp.335-341, 2012.

[3] Bingchiang jeng et al, "Specific Effort Estimation Method using Function Point", Journal Information Science and Engineering, pp.1363-1376, 2011.

[4] Cigdem Gencel,"How to use Cosmic Functional Size in Effort Estimation Models?" in Proceeding on IWSM/Metrikon.pp.1-5,2008.

[5] Frank Vijay.J, Manokaran.C, "Initial Hybrid method for analyzing software estimation, Benchmarking and Risk Assessment using Design of Software" ,Journal of Computer Science,pp.717-724,2009.

[6] Marcus Meyerhafer et al,"Estimating Non-Functional Properties of Component-Based software Resource consumption", Electronic notes in Theoretical computer Science. pp. 25-45, 2005.

[7] Roy K.Clemmons, "Project Estimation with Use Case Points", The Journal of Defense Software Engineering, pp.18-22, 2006.

[8] Thomas Fehlmann, Luca santillo,"From Story Points to Cosmic Function Points in Agile software Development- A six sigma Perspective", Software Metrics Kongress, pp.1-11, 2010.

[9] Vinh T.ho, Alain abran, Fetcke Thomas," A Comparative Study Case of COSMIC-FFP, Full Function Point and IFPUG Methods", in Proceeding ITON-10,pp.7-11,2010