Review of Computer Engineering Research
2015 Vol.2, No.2, pp.39-46
ISSN(e): 2410-9142
ISSN(p): 2412-4281
DOI: 10.18488/journal.76/2015.2.2/76.2.39.46

A DETAILED ANALYSIS OF SOFTWARE COST ESTIMATION USING COSMIC-FFP

Gaurav Kumar[1]† --- Pradeep Kumar Bhatia[2]

[12]Department of Computer Science & Engineering, Guru Jambheshwar University of Science & Technology, Hisar, Haryana, India

ABSTRACT

Software cost estimation is one of the most challenging tasks in software engineering. For the estimation, Function points are useful in the business application software domain and problematic in the real-time software domain. Full Function Points (FFP) are useful for functionality-based estimation, specifically for real-time and embedded software. Functional size measurement method that has user view of functional requirements developed by Common Software Measurement International Consortium (COSMIC) called COSMIC-FFP. By using COSMIC-FFP model, an early prediction of the functional complexity of the software throughout the software development life cycle within given budget constraints, reliability can be done. In this paper, a detailed analysis with process flow of COSMIC-FFP model has been discussed.

Keywords: COSMIC-common software measurement international consortium, CFSU-cosmic functional size unit, FUR-functional user requirement, FFP - full function points, IFPUG-international function point user group, FPA – function point analysis, PL- project leader.

Received: 14 January 2015/ **Revised:** 17 February 2015/ **Accepted:** 21 February 2015/ **Published:** 24 February 2015

1. INTRODUCTION

Accurate software estimation is an important requirement in today's highly competitive world. Estimation is used to calculate the size of the development work to be carried out. The purpose of FFP is to extend the IFPUG FPA accuracy of the real time systems estimation. The size estimation is done using Cosmic FFP 2.2. This Procedure is applicable to the Development Projects/Change Requests in existing projects/products. COSMIC-FFP focuses on the "user view" of functional requirements and is designed to measure the functional size of 'data-rich' business/management information systems, 'control-rich' real-time software, multi-layer systems and/or multi-tier architecture. It is not designed to measure the functionality of software which is 'algorithm-rich', that is software with complex mathematics, games, streaming software i.e. audio,

† Corresponding author

video. COSMIC-FFP version 2.0 uses a two-phase approach for functional size measurement (mapping and measurement), a simplified set of base functional components (BFC) and a scalable aggregation function [1]. It also measures software complexity which is related to the size of the software and unpredictability (uncertainty) of its behaviour. Functional Size is independent of software language and development methods and is defined as size of the software derived by quantifying the Functional User Requirements i.e. what the software is expected to do for its users. The process of COSMIC-FFP Measurement consists of three main phases [2]:

a) Setting the Measurement Strategy

b) Mapping the 'Functional User Requirements' (or 'FUR') of the software to be measured to the COSMIC-FFP concepts

c) Measuring the resulting COSMIC-FFP model of the FUR

For effort estimation, main input can be considered as Functional size which is related to effort. Team size, development type, programming language type, organization type and application type are some of the other factors that have significant impact on this measurement.

The paper is organized as follows. Section 2 shows work done in the field with the reference of literature review. Section 3 describes the procedure flow with the help of flow diagrams. Section 4 presents size estimation guidelines with the help of identification of different components. Section 5 represents Effort calculation. Section 6 concludes the paper with some discussion and future work.

2. LITERATURE REVIEW

Gaurav and Pradeep Kumar [3] proposed an automated cost estimation model using Neural Network with KLOC as input and uses COCOMO model parameters. It helps project manager to calculate software cost using fast and realistic estimate of the project effort and development time.

Tharwon [4] gave a detail analysis for software sizing measurement and commented on the findings and future trends and challenges of the software size estimation models. Kenneth and Rogardt [5] made an effort to minimize manual effort for estimation of Code Size. They defined a UML model to capture all information and developed a tool for automated estimation of Code Size based on CFP.

Cigdem [6] described a study on how to use COSMIC functional size as an input for effort estimation models. The study 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.

The results of multiple case study in which COSMIC method was used for size measurement is explained. Manar Abu [7] proposed a model to assess the quality using COSMIC-FFP and predicts the functional complexity of the behavior of software from the very initial requirements phase; with a mechanism for generating black-box test cases, test case prioritization and test set

adequacy monitoring and optimization within given budget constraints, and an early prediction of reliability based on Markov chains is calculated.

Çigdem, et al. [8] presented a case study on implementing COSMIC FFP and Use Case Points (UCP) methods to an industrial project. The estimated effort is compared with the actual development effort utilized for the system.Gennaro, et al. [9] addressed the problem of estimating effort in developing web applications. They used an adapted Cosmic FFP method on design documents to count data movements. They carried out an empirical analysis to verify the usefulness of method to predict effort for web application development.

3. PROCEDURE FLOW

The COSMIC-FFP measurement method breaks down the software architecture into software layers where each software layer can receive requests from the above layers and can request for services from the below layers. Measurement system includes various phases as explained under.

A. Mapping Phase

The mapping phase is based on information provided by the functional user requirements [10]. Approved SRS is the Entry criteria & Input for this phase.

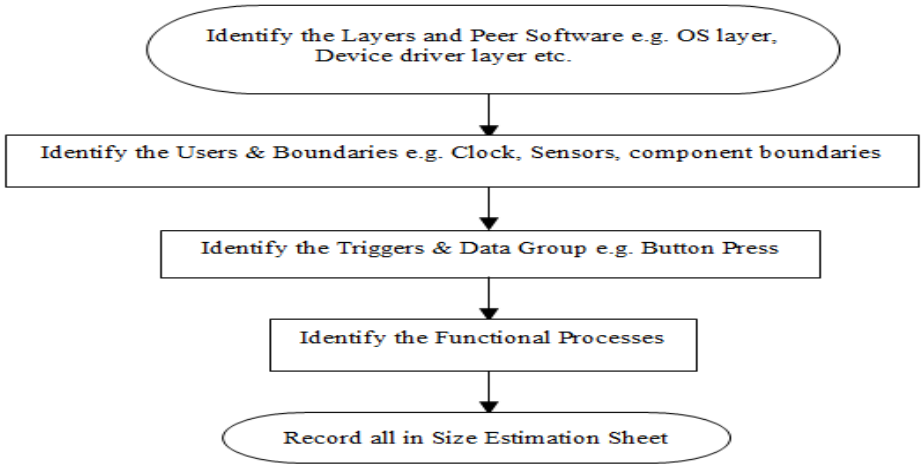

Figure-1. Processing during Mapping Phase

B. Measurement Phase

In this phase, a set of functional processes is established. Each of those processes encompasses a unique set of data movements or data manipulations.

The Cosmic FFP software model distinguishes four types of data movements:

- Entry, data is moved from the user across the process boundary inside the functional process.

- Exit, data is moved from inside the functional process across the process boundary to the users.
- Read, moves data inside the process from a persistent data store (for example, a database).
- Write, moves data from inside the process to a persistent data store.

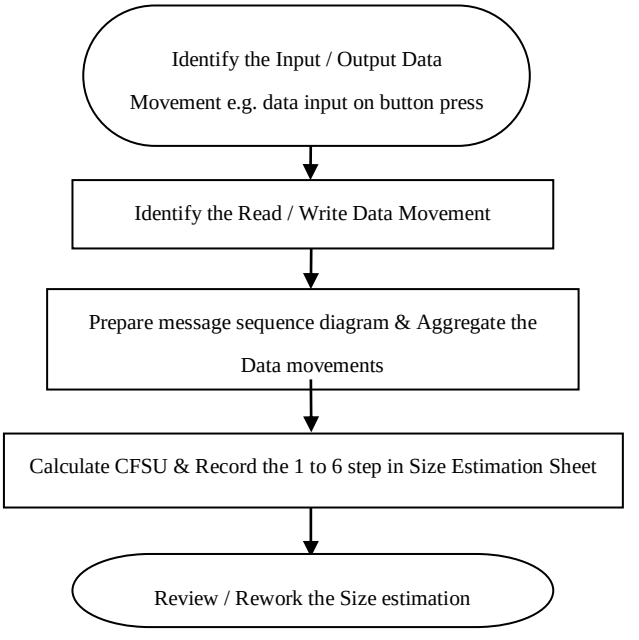

Figure-2. Processing during Measurement Phase

4. SIZE ESTIMATION GUIDELINES

A. Identifying Layers

a) Functional service software packages such as database management systems, operating systems or device drivers are generally considered as distinct layers.

b) If software is conceived using an established architectural paradigm of layers then uses that paradigm to identify the layers.

c) The software application level is generally considered to occupy the highest layer level.

d) When in doubt, use the concepts of coupling and cohesion to distinguish between interacting layers.

B. Identifying Boundaries

a) Identify triggering events, and then identify the functional processes enabled by those events. The boundary lies between the triggering events and those functional processes.

b) For real-time or technical software, use the concept of layers to assist in the identification of the boundary.

C. Identifying Functional Processes

a) A triggering event-type gives rise to a triggering Entry-type, i.e. the movement of a data group-type defined as comprising a certain number of data attribute-types. If an occurrence of a specific event-type triggers the entry of a data group comprising data attributes A, B and C, and then another occurrence of the same event-type triggers an entry of a data group which has values for attributes A and B only, this is not considered to be a different triggering event-type. It is considered to be the same for the purpose of identifying COSMIC-FFP functional processes. Thus only one entry and one functional process are identified, manipulating data attributes A, B and C.

b) In the case of real-time software, a functional process is also triggered by an event. It terminates when a point of asynchronous timing is reached. A point of asynchronous timing is equivalent to a self-induced wait state.

D. Identifying Data Group

Measurement practice says that in business application software, a data group is identified for each 'entity-type' (or 'Third Normal Form' relation) found in the normalized data model of the measured software. These are usually data groups showing indefinite persistence and the software is required to store data about the entity-types concerned. In COSMIC-FFP, the term 'Object of interest' is used instead of 'entity-type' or 'TNF relation' related to specific software engineering methods e.g. in the domain of management information software, an Object of interest could be 'employed' (physical) or 'order' (conceptual) – the software is required to store data about employees or orders.

E. Identifying ENTRY (E)

a) Clock-triggering events are always external to the software being measured. Therefore, an event occurring every 3 seconds is associated with an ENTRY moving one data attribute, for instance. Even if such a triggering event is generated periodically not by hardware, but a software functional process, the later can be ignored in the measurement since it occurs, by definition, outside of the boundary of the software being measured.

b) Unless a specific functional process is necessary, obtaining the time from the system's clock is not considered as an ENTRY. For instance, when a functional process writes a time stamp, no ENTRY is identified for obtaining the system's clock value.

F. Identifying EXIT (X)

When measuring size from the End User Measurement Viewpoint, by convention all software messages generated without user data (e.g. confirmation and error messages) are considered to be separate occurrences of one message-type. Therefore, a single EXIT is identified

to represent all these messages within the scope of the functional process where these messages are identified.

For instance, consider functional processes A and B identified within the same layer. "A" can potentially issue 2 distinct confirmation messages and 5 error messages to its users and "B" can potentially issue 8 error messages to its users. In this example, one EXIT would be identified within functional process "A" (handling 5+2=7 messages) and a separate EXIT would be identified within functional process "B" (handling 8 messages) [11].

G. Identifying READ (R)

a) The data movement retrieves data attributes from a data group in persistent storage.

b) The data movement retrieves data attributes belonging to only ONE data group, that is, data about a single Object of interest. Identify one READ for each Object of interest for which data attributes are retrieved in any one functional process

c) The data movement does not receive or exit data across the boundary or write data.

d) Within the scope of the functional process where it is identified, the data movement is unique, that is, the processing and data attributes identified are different from those of any other READ included in the same functional process.

e) During a functional process, the Read (or a Write) of a data group can only be performed on the data describing an Object of Interest to the User. Constants or variables which are internal to the functional process, or intermediate results in a calculation, or data stored by a functional process resulting only from the implementation, rather than from the FUR, are not data groups and are not taken into account in the functional size.

H. Identifying WRITE (W)

a) The data movement moves data attributes to a data group on the persistent storage side of the software.

b) The data movement moves the values of data attributes belonging to only ONE data group that is data about a single Object of interest. Identify one WRITE for each Object of interest for which data attributes are referenced in any one functional process.

c) The data movement does not receive or exit data across the boundary, or read data.

d) Within the scope of the functional process where it is identified, the data movement is unique, that is, the processing and data attributes identified are different from those of any other WRITE included in the same functional process.

e) A requirement to delete a data group from persistent storage is measured as a single Write data movement.

f) During a functional process, the step of storing a data group that does not persist when the functional process is complete is not a Write; examples are updating variables, which are internal to the functional process or producing intermediate results in a calculation.

5. EXPERIMENTAL RESULTS

One Cosmic Functional Size Unit (CFSU) is assigned for each entry/exit of a data group between front end & software and for each read/write operation by a data group between software & back end [12]. COSMIC-FFP function point count, measured in cfsu (COSMIC Functional Size Unit), is computed by summing all Entries, Exits, Reads, and Writes.

$$Size = \sum Size_{Entry} + \sum Size_{Exit} + \sum Size_{Read} + \sum Size_{Write} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (1)$$

$$Effort_{Specify} = 4.0342 \times Size^{0.9903} \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (2)$$

$$Effort_{Build} = 12.313 \times Size^{1.015} \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (3)$$

$$Effort_{Test} = 5.2124 \times Size^{1.024} \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (4)$$

A case study applied on 3 projects for the COSMIC analysis has been shown in Table 1 by considering equation (1-4) given above.

Table-1. CFSU Size Measurement & Effort Calculation

Project	No. of functional processes	No. of Entries	No. of Exits	No. of Read	No. of Write	Size	Effort Specify (in person hours)	Effort Build (in person hours)	Effort Test (in person hours)	Total Effort (in person hours)
1	24	77	88	27	30	222	849.866	2964.23	1317.35	5131.44
2	32	60	59	20	25	164	629.67	2179.86	966.13	3775.66
3	41	94	72	14	49	229	876.40	3059.12	1359.90	5295.42

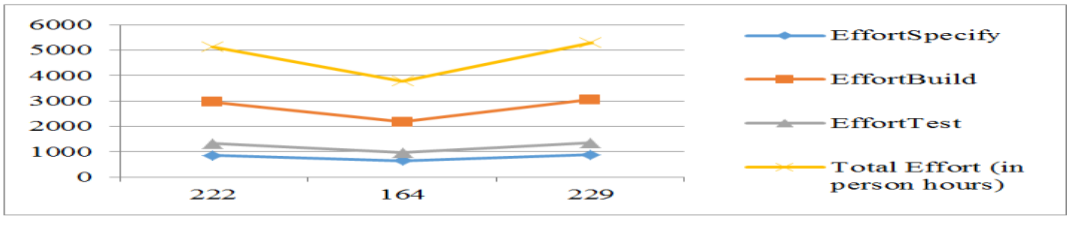

Figure-3. Graph showing various efforts depending on size.

5. CONCLUSION

This study uses new estimation methodology for analysis of software estimation using COSMIC FFP showing relationships of project effort w.r.t. functional size unit. Cosmic FFP provides simple, easy to use, proven & practical solution for estimation of software size,

productivity, performance and hence quality. The method can be adapted using modern methodology like neural network, fuzzy, genetic programming etc. to provide more accurate, automated results in terms of effort calculation.

Funding: This study received no specific financial support.

Competing Interests: The authors declare that they have no competing interests.

Contributors/Acknowledgement: All authors contributed equally to the conception and design of the study.

REFERENCES

[1] B. Barry, A. Chris, and C. Sunita, "Software development cost estimation approaches – a survey," *Annals of Software Engineering*, vol. 10, pp. 177-205, 2000.

[2] R. Bridges, "Estimating with confidence: Applying COSMIC method for estimation in the avionics industry," *European SEPG*, pp. 1-22. [Accessed 11-14 June 2007], 2007.

[3] K. Gaurav and B. Pradeep Kumar, "Automation of software cost estimation using neural network technique," *International Journal of Computer Applications*, vol. 98, pp. 11-17, 2014.

[4] A. Tharwon, "The development and achievements of software size measurement," in *Proc. International Multi Conference of Engineers and Computer Scientists (IMECS)*, 2012.

[5] L. Kenneth and H. Rogardt, "A model-based and automated approach to size estimation of embedded software components," in *Proc. of Springer 14th International Conference*, 2011, pp. 334–348.

[6] G. Cigdem, "How to use COSMIC functional size in effort estimation models," in *Proc. of Springer International Conferences IWSM*, 2008, pp. 196-207.

[7] T. Manar Abu, "Exploratory study on an innovative use of COSMIC-FFP for early quality assessment," PhD Thesis in Department of Computer Science and Software Engineering , Montréal, Concordia University, Canada, 2007.

[8] G. Çigdem, B. Luigi, D. Onur, and E. Pinar, "A case study on the evaluation of COSMIC-FFP and use case points," in *Proceedings of Smef*, 2006, pp. 121-140.

[9] C. Gennaro, F. Filomena, G. Carmine, T. Genoveffa, and V. Giuliana, "A cosmic-FFP based method to estimate web application development effort," in *Proc. of Springer 4th International Conference on Web Engineering*, 2004, pp. 161-165.

[10] H. Harold Van, "Early estimating using COSMIC-FFP," in *Proc. of 2nd Software Metrics Eauropean Forum (SMEF)*, 2005.

[11] S. Oligny, A. Abran, and C. Symons, "COSMIC-FFP – some results from the field trials," *Common Software Measurement International Consortium (COSMIC), Citeseer*, pp. 1-12, 2000.

[12] Lois and Clark IT Services, "Project size estimation using cosmic FFP," in *Proc. of 2nd Software Metrics Eauropean Forum (SMEF,)*, 2005.