A Study on Improvement of Cost Estimation Tool

Ankit Fulzele [1], Rahul Kumar Chawda[2]

[1] Student of 6 th Sem MCA Department , Kalinga University, Raipur

[2] Assistant Professor, Department of Computer Science, Kalinga University, Raipur

rahul.chawda3@gmail.com [2]

Abstract Unified Modeling Language (UML) is a standardized general-purpose modeling language in the field of software engineering. The Unified Modeling Language includes a set of graphic notation techniques to create visual models of object-oriented software intensive systems. In software engineering, a class diagram in the UML is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes.

Keyword Unified Modeling Language, attributes, operations

I. INTRODUCTION

A survey says, almost one-third projects exceed their budget and is delivered late and two-thirds of all projects overrun their original estimates. It is impossible for a manager or system analyst to accurately predict the cost and effort required to develop software. Without accurate cost estimation capability, project managers can't determine how much time and manpower the project should take and that means the software portion of the project is out of control from its beginning. It is difficult to understand and estimate a software product that cant be seen and touched. Software grow and change when it is written. In project management, the most challenging task is cost estimation. It is necessary to correctly estimate required resources and schedules for software development projects. Software cost estimation process includes the following:

- ˆ Estimation of the size of the software product to be produced
- ˆ Estimation of the effort required
- ˆ Development of preliminary project schedules
- ˆ Estimation of overall cost of the project

Bottom Up Approach

A bottom-up approach is the piecing together of systems to give rise to grander systems. It emphasize mainly on coding and early testing, which begins as soon as the first module has been specified. This approach, runs the risk that modules may be coded without having a clear idea of how they link to other parts of the system, and such linking may not be as easy as first thought. Main benefit of the bottom-up approach is re-usability of code. Using bottom-up estimating method, cost of each software components is estimated and then combine all the results to arrive at an estimated cost of overall project. It aims at constructing the estimate of a system from the knowledge accumulated from the small software components and their interactions. COCOMO model is developed using this approach.

COCOMO Model

The Constructive Cost Model (COCOMO) is an algorithmic software cost estimation model. It is a regression

model which uses basis regression formula with parameters that are derived from historical project data and current as well as future project characteristics. Software development effort is calculated in terms of program size by COCOMO. Program size is estimated in thousands of source lines of code (SLOC). COCOMO assumes that the system and software requirements have been defined already, and that these requirements are stable.

II. LITERATURE REVIEW

In software project development, size evaluation is one of the main tasks with reliable cost and effort estimations. To estimate the size of a software system several measures have been defined so far. Some are as follows:

- **Function point Approach**
- **Class Point Approach**

T. Uemura et al., Function Point Analysis (FPA) was developed by IBM in response to a number of problems arising in measuring the size of system in terms of lines of codes. FPA measures size of an application system in two areas: the specific user functionality and the system characteristics. The specific user functionality is the measurement of functionality delivered by the application for user request. The five function types identified are: external output, external enquiries, external input, external interface files and internal logical files. For each function identified as one of the five function types given, it is further classified as low, average or high and a weight is given to each. The sum of weights tells about the size of information processing and is referred as Unadjusted Function Points.

S. Kanmani et al., Class Point approach provides a system level estimation of the size of Object Oriented products. It has been derived by recasting the ideas underlying the function point analysis within the Object Oriented paradigm and by suitably combining well-known OO measures. The process of Class Point size estimation is composed of 3 main phases, corresponding to analogous phases in function point approach. The following steps show how class point is calculated:

- ˆ Information procession size estimation

1. Identification and classification of class
2. Calculation of complexity level of each class
3. Calculating Total Unadjusted Class Points
   - ˆ Estimation of Technical Complexity Factor(TCF)
   - ˆ Final evaluation of Class Point

M. Jorgensen et al, FPA technique applies different formula while measuring size of system for software development and maintenance. So, the type of function point count should be deter mined at the outset. Three types of function point counts:

1. Enhancement project function point count
2. Development project function point count
3. Application function point count

G. Costagliola et al., the unadjusted function point reflect the functionality of logical system provided to the user. Five function types are used to determine unadjusted function point. Those function points are:

- ˆ Internal Logical File (ILF)
- ˆ External Interface File (EIF)
- ˆ External Input (EI)
- ˆ External Output (EO)
- ˆ External Inquiry (EQ)

Each function type is assessed for its complexity (low, average or high) as follows

- Depending on the number of file type referenced (FTR) and data element type (DET), EI, EO and EQ

are given complexity ratings; and Depending on number of record element types (RET) and data element types

- (DET), EIF and ILF are given complexity rating .

III. CONCLUSION

The Extended Class Point Approach provides system-level size estimation of Object Oriented product and from empirical validation, it exhibits better performance than the Class Point Approach. Software developed for class point calculation is simple to use. Calculating Adjusting Class Point value for large number of softwares using this tool, compared the actual effort and the estimated effort using regression analysis. Through the effort estimation, we can conclude that, how much effort the project has used and then we can have a deeper knowledge of developer teams professional skill level. Leaders of company need this kind of data to manage the company and arrange tasks according to the developer's professional skill.

References

1. Nikki Panlilio-Yap. Software estimation using the slim tool. In Proceedings of the 1992 conference of the Centre for Advanced Studies on Collaborative research - Volume 1, CASCON '92, pages 439–475. IBM Press, 1992.
2. K. Pillai and V.S. Sukumaran Nair. A model for software development effort and cost estimation. Software Engineering, IEEE Transactions on, 23(8):485–497, 1997
3. Barry W. Boehm, Clark, Horowitz, Brown, Reifer, Chulani, Ray Madachy, and Bert Steece. Software Cost Estimation with Cocomo II with Cdrom. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1st edition, 2000.
4. T. Uemura, S. Kusumoto, and K. Inoue. Function point measurement tool for uml design specification. In Software Metrics Symposium, 1999. Proceedings. Sixth International, pages 62–69, 1999
5. CPM IFPUG. Counting practices manual, release 4.1. 1. IFPUG–International Function Point Users Group, 2000.
6. T. Fetcke, A. Abran, and Tho-Hau Nguyen. Mapping the oo-jacobson approach into function point analysis. In Technology of Object-Oriented Languages and Systems, 1997. TOOLS 23. Proceedings, pages 192–202, 1997
7. M. Jorgensen and M. Shepperd. A systematic review of software development cost estimation studies. Software Engineering, IEEE Transactions on, 33(1):33–53, 2007
8. J.E. Matson, B.E. Barrett, and J.M. Mellichamp. Software development cost estimation using function points. Software Engineering, IEEE Transactions on, 20(4):275–287, 1994.
9. G. Costagliola, F. Ferrucci, G. Tortora, and G. Vitiello. Class point: an approach for the size estimation of object-oriented systems. Software Engineering, IEEE Transactions on, 31(1):52– 74, 2005.
10. SangEun Kim, William Lively, and Dick Simmons. An effort estimation by uml points in early stage of software development. In Software Engineering Research and Practice'06, pages 415–421, 2006.
11. Wei Zhou and Qiang Liu. Extended class point approach of size estimation for oo product. In Computer Engineering and Technology (ICCET), 2010 2nd International Conference on, volume 4, pages V4–117–V4–122, 2010.2
12. S. Kanmani, J. Kathiravan, S. Senthil Kumar, and M. Shanmugam. Neural network based effort estimation

using class points for oo systems. In Proceedings of the International Conference on Computing: Theory and Applications, ICCTA '07, pages 261–266, Washington, DC, USA, 2007. IEEE Computer Society.

13. M. Shepperd and C. Schofield. Estimating software project eff ort using analogies. Software Engineering, IEEE Transactions on, 23(11):736–743, 1997.