Volume 5, Issue 8, August 2015 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Design & Development of Cost Measurement Mechanism for Reengineering Project

Abhilasha Yadav[1], Anand Rajavat[2]
[1]Student, [2]Associate Professor
Shri Vaishnav Institute of Technology & Science
Rajiv Gandhi Technical University, Indore, (MP) India

Abstract: Small Projects are very easy to estimate and accuracy is not very important. But as the size of project increases, required accuracy is not very important. But as the size of project increases, required accuracy is very important which is very hard to estimate. A good estimate should have amount of granularity so it can be explained. Since the effort invested in a project is one of the most important and most analyzed variables. So the prediction of this value while we start the software projects, it helps to plan any forthcoming activities adequately. The proposed tool is expected to be more accurate & viable to conduct cost estimation process web based project & type of risk to indicate in web based project & product metrics.

Keywords: Cost estimation, Function points, Re-engineering, Reverse Engineering

I. INTRODUCTION

Software cost estimating has been an important but difficult task since the beginning of the computer era in the 1940s. As software applications have grown in size and importance, the need for accuracy in software cost estimating has grown, too. Even more serious, a significant percentage of large software systems run late, exceed their budgets, or are canceled outright due to severe underestimating during the requirements phase. In fact, excessive optimism in software cost estimation is a major source of overruns, failures, and litigation. Figure 1. Illustrates the basic principles of modern commercial software Cost-estimating tools.

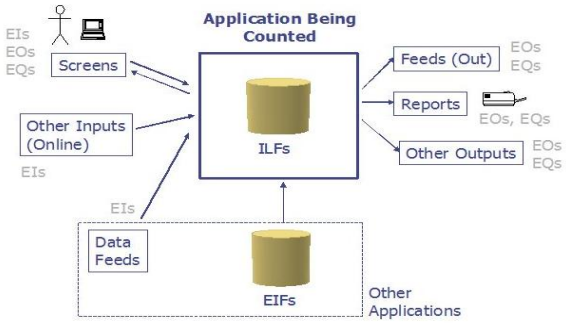

Figure 1. Software-estimating principles.

Every form of estimation and every commercial software cost-estimating tool needs the sizes of key deliverables in order to complete an estimate. Size data can be derived in several fashions, including the following:

- Size prediction using an estimating tool's built-in sizing algorithms
- Sizing by extrapolation from function point totals
- Sizing by analogy with similar projects of known size
- Guessing at the size using "project manager's intuition"
- Guessing at the size using "programmer's intuition"
- Sizing using statistical methods or Monte Carlo simulation

II. LITERATURE SURVEY

A good overview of these methods can be found in [21] Most of these methods can be classified into four major categories: expert judgment, analogy-based, group consensus, and decomposition. Group consensus techniques such as Wideband Delphi [21] and Planning Poker use a group of people instead of individuals to derive estimates of size. Apart from the four main size estimation method categories mentioned above, other categories also exist. Probabilistic methods such as PERT sizing [31] are an example. Fuzzy logic-based size estimation methods such as hose described in [31] are another example.In COCOMO, the number of statements or logical SLOC, is the standard SLOC input. Logical SLOC is less sensitive to formats and programming styles, but it is dependent on the programming languages used in the source code [22].

For software maintenance, the count of added/new, modified, unmodified, adapted, reused, and deleted SLOC can be used. Software estimation models usually aggregate these measures in a certain way to derive a single metric commonly called effective SLOC or equivalent SLOC [23, 24, 25]. SLOC has been widely accepted for several reasons. It has been shown to be highly correlated with the software cost; thus, they are relevant inputs for software estimation models [21,23].An experienced programmer may write fewer lines of code than an inexperienced one for the same purpose, resulting in a problem called productivity paradox [26]. A third limitation is a lack of a consistent standard for measurements of SLOC. As aforementioned, SLOC could mean different things, physical lines of code, physical lines of code excluding comments and blanks, or logical SLOC. There is also no consistent definition of logical SLOC [22]. The lack of consistency in measuring SLOC can cause low estimation accuracy as described in [27]. A forth limitation is that SLOC is technology and language dependent. Thus, it is difficult to compare productivity gains of projects of varying technologies.

III. PROPOSED PROCESS DESCRIPTION

Find function point from Exiting Code using Proposed Algorithm:

Step 1: Determine function point count

- Development project
- Enhancement project
- Application

Step 2: Identify the counting scope and application boundary

- The unadjusted function point count (UFP) reflects the specific countable functionality provided to the user by the project or application.
- The UFP computation is divided into two categories with five sub-categories:
  - Data Functions
    - Internal Logical Files
    - External Interface Files
  - Transactional Functions
    - External Inputs
    - External Outputs
    - External Inquiries

Step3: function point functionality offer to the user to convene internal and external data requirements

Step4: Data functions are Internal logical files/External interface files.

Step5: An internal logical file (ILF) is a user exacting group of logically linked

Step6: An external interface file (EIF) is a user specific group of sensibly related data

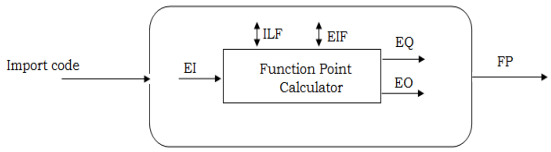

Figure 2 block diagram of proposed System

IV. PROPOSED PROCESS IMPLEMENTATION

The tools for performing this task are static analyzers. Since the user system will normally contain java language, there must be an analyzer for each language. The important thing is that the results of the analysis are the same. In the calculation of the AFP, calculating the value adjustment factor (VAF) is an earmark of the general functionality provided to the user. The VAF is derived from the sum of the degree of influence (DI) of the 14 general system characteristics (GSCs). . The third and the last stage is the final calculation of the function points. With the help of the following equation we can get the total points of an application.

AFP = UFP * VAF…….. (3)

Where AFP = adjusted function points; UFP = unadjusted function points; and VAF = value adjustment factor. Five information domain characteristics are determined and counts are provided in appropriate table location. In the UFP is the measuring parameters, i.e. External Input (EI), External Output (EO), External Query (EQ),Internal Logical File (ILF), External Interface File (EIF) and UFP is the unadjusted function point.

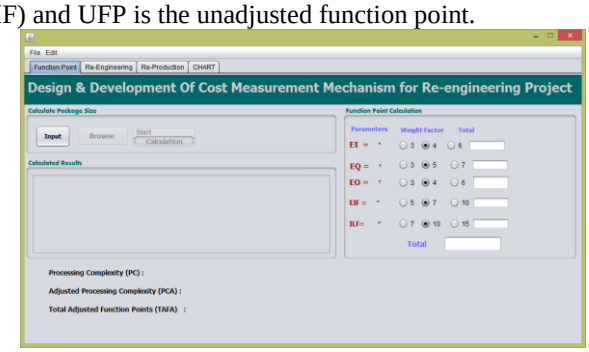

Figure3. GUI of function point calculator

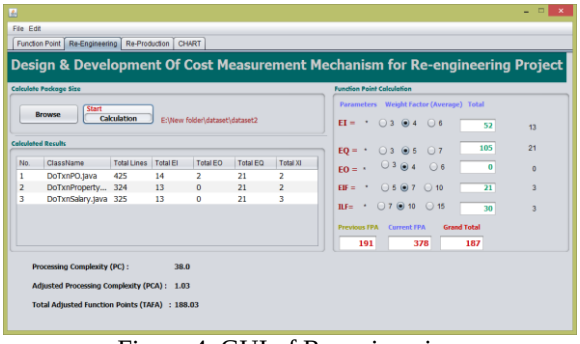

Figure 4. GUI of Reengineering

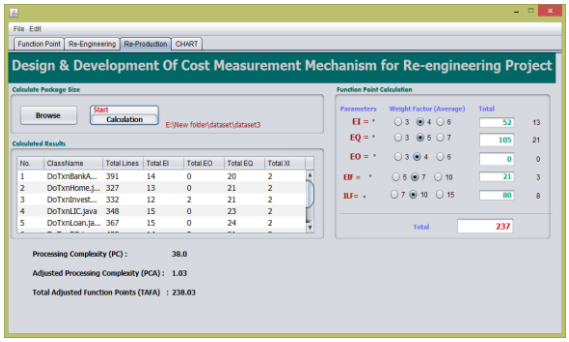

Figure 4. GUI of Redevlopment

V. RESULT ANALYSIS

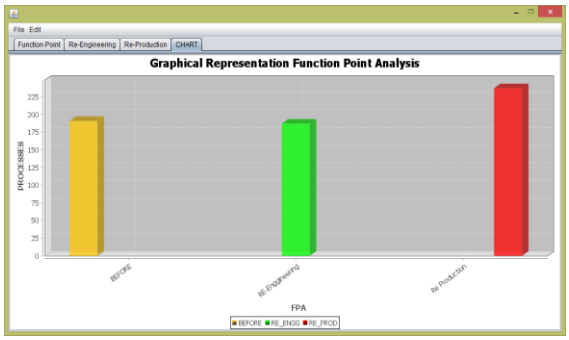

Figure 5 Final Function point analysis

VI. CONCLUSION

It is clear that there are still many factors not considered. Every estimation approach uses only a subset of the potential measurements which could have an influence on the outcome of the project. Like the universe in which we live, the dimensions of the estimation problem are not infinitive but they are growing faster than our ability to comprehend them. Thus there will never be a comprehensive, all encompassing estimation method. There will be only practical approaches based on the experience of those which created them within a limited domain of application. The smaller that domain is, the greater the accuracy of the method. For that reason the author has limited his estimation approach to the reengineering domain, a domain for which he is familiar. Even within this limited domain, there is still much to be done. As pointed out before, there is a need to revise and refine the time estimation. Studies are required on the relation between effort and time in reengineering projects. There is also a need to study more closely the impact of quality

REFERENCES

[1] Verner, June M. and Tate, Graham, "A Model for Software Sizing", Journal of Systems and Software, IEEE Software, pp. 173-177, July 1987.

[2] Albrecht, Allan J. and Gaffney (Jnr) , John E., "Software Function Source Lines of Code and Development Effort Rediction: A Software Science Validation", IEEE Transactions on Software Engineering, Vol. SE-9, No. 6, pp. 639-647, Nov. 1983.

[3] N. E. Fenton and S. L. Pfleeger, 1997. Software Metrics: A Rigorous and Practical Approach, 2nd Edition Revised ed. Boston: PWS Publishing.

[4] L. M. Laird, and M. C. Brennan, 2006. Software Measurement and Estimation: A Practical Approach, Wiley-IEEE Computer Society Pr, ISBN: 0-471-67622-5.

[5] Forselius, P., 2004. Moving from Function Point Counting to Better Project Management and Control, IWSM/MetriKon Presentation.

[6] C. R. Symons, Software Sizing and Estimating - MkII FPA (Function Point Analysis) , John Wiley and Sons, Chichester, U.K., 1991.
[7] A. Abran, M. Maya, J. M. Desharnais, and D. St-Pierre, "Adapting function points to real-time software," American Programmer, Vol. 10, 1997, pp. 32-43.
[8] C. Jones, Applied Software Measurement: Assuring Productivity and Quality, McGraw-Hill, New York, 2008.
[9] N. A. S. Abdullah1, R. Abdullah2, M. H. Selamat2, A. Jaafar2,Software Security Characteristics for Function Point Analysis, Proceedings of the 2009 IEEE IEEM
[10] Scaffidi, C., Shaw, M., and Myers, B. The "55M End User Programmers" Estimate Revisited. Technical Report CMUISRI-05-100, Carnegie Mellon University, Pittsburgh, PA, 2005.
[11] G. Fischer, E. Giaccardi, Y. Ye, A. G. Sutcliffe, and N. Mehandjiev. Meta-design: A manifesto for end-user development. Communications of the ACM, 47 (9) :33–37, September 2004.
[12] Contributions, Costs and Prospects for End-User Development, Alistair Sutcliffe, Darren Lee & Nik Mehandjiev
[13] "Object based designing of pattern using U2ML" in proceeding of International conference of advances in computer vision and information technology (ACVIT-2007)
[14] Boehm, B., Clark, B., Horowitz, E., Westland, C., Madachy, R., and Selby, R. Cost models for future software life cycle processes: COCOMO 2.0. Annals of Software Engineering, Special Volume on Software Process and Product Measurement (1995) .
[15] Appendix C: COCOMO II Process Maturity led by Dr. Barry Boehm
[16] http://sunset.usc.edu/research/COCOMOII/Docs/modelman.pdf
[17] http://www.devshed.com/c/a/Practices/An-Overview-of-Function-Point-Analysis/3
[18] http://www.cs.toronto.edu/%7Esme/papers/2005/ESEC-FSE-05-Aranda.pdf
[19] http://www.crosstalkonline.org/storage/issue-archives/2011/201101/201101-Stark.pdf
[20] http://approachtoproject.com/component/k2/item/10-software-estimation-techniques.html
[21] Boehm B.W. (1981), "Software Engineering Economics", Prentice-Hall, Englewood Cliffs, NJ, 1981.
[22] Nguyen V., Deeds-Rubin S., Tan T., Boehm B.W. (2007), "A SLOC Counting Standard," The 22nd International Annual Forum on COCOMO and Systems/Software Cost Modeling. DOI = http://csse.usc.edu/csse/TECHRPTS/2007/usc-csse-2007737/usc-csse-2007-737.pdf

