A Software Cost Estimation using Function Point Method

Ashutosh Tiwari[1] A.K.Shukla[2]
[1]M.Tech. Student [2]Assistant Professor
[2]Department of Computer Science & Information Technology
[1,2]Sam Higginbottom Institute of Agriculture, Technology & Sciences, Allahabad

Abstract— Numerous organizations around the globe have designed commercial as well as educational apps for online, the best known example of any hypermedia process. But acquiring good Net applications is actually expensive, mostly with regard to time and degree of difficulty for the authors. Our study tries to be able to predict the time and effort needed for the development of web pages of a certain category, here we have now restricted each of ourselves on the domain associated with news sites. Our document is suggesting a fairly easy method pertaining to efforts estimation of internet based projects. Software Cost Estimation (SCE) is often a process associated with predicting the actual efforts as well as cost with regard to money, schedule as well as staff for virtually any software process, Software price estimation is an old arts come with the beginning of computer industry in 1940s and contains been developed many times until forming function items by Albrecht with 1979.Previous scientific tests comparing the actual prediction exactness of hard work models developed using over the internet and single-company information sets are inconclusive. This Thesis as a result replicates some sort of previous study by analyzing how successful estimate work for net projects that participates in an individual company. The hard work estimates utilized in our analysis was obtained through effort estimation technique, namely Web case-Point Computation.

Key words: Single-Company Effort Model, Software Cost Estimation, Software Cost Estimation Techniques, Web Applications, Web Projects.

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. In the early days of software, computer programs were typically less than 1000 machine instructions in size (or less than 30 function points), required only one programmer to write, and seldom took more than a month to complete. The entire development costs were often less than $5000. Although cost estimating was difficult, the economic consequences of cost-estimating errors were not very serious.

Today some large software systems exceed 25 million source code statements (or more than 300,000 function points), may require technical staffs of 1000 personnel or more, and may take more than five calendar years to complete. The development costs for such large software systems can exceed $500 million; therefore, errors in cost estimation can be very serious indeed.

Software is now the driving force of modern business, government, and military operations. This means that a typical Fortune 500 corporation or a state government may produce hundreds of new applications and modify hundreds of existing applications every year. As a result of the host of software projects in the modern world, software cost estimating is now a mainstream activity for every company that builds software. In addition to the need for accurate software cost estimates for day to-day business operations, software cost estimates are becoming a significant aspect in litigation. Over the past fifteen years, the author and his colleagues have observed dozens of lawsuits where software cost estimates were produced by the plaintiffs, the defendants, or both.

Figure 1.1 illustrates the basic principles of modern commercial software cost-estimating tools.

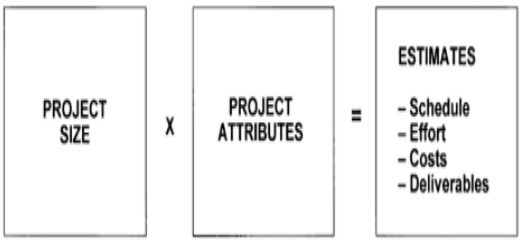

Fig. 1.1: Software-Estimating Principles

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. PROPOSED SYSTEM

The tool is expected to be more accurate & viable to conduct cost estimation process web based project & type of risk to indicates in web based project & shown by project & product metrics. This tool provide following characteristics:

A. Analysis of the Reengineering Requirements

To collect the user requirements the author uses a questionnaire with three classes of reengineering requirements size requirements, complexity requirements and quality requirements. The size requirements are an inventory of all the system components affected by the reengineering project.

B. Analysis of the Existing System

After analyzing the requirements, the user may decide that there is insufficient benefit to be gained or that the whole thing is simply too expensive. If, however, he decides to proceed, the next step is to analyze the software itself. This

has to be done automatically within a few days otherwise it will be too expensive.

C. Transformation of Source Metrics

In the third step the results of the source analysis are transferred to another tool – Estimator tool – for the purpose of making the final cost estimation. The metrics obtained from the source analysis are stored in an XML file. This file is read in by the Estimator tool and the metrics moved to a relational table.

D. Computation of AFP (Adjusted Function Point)

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).

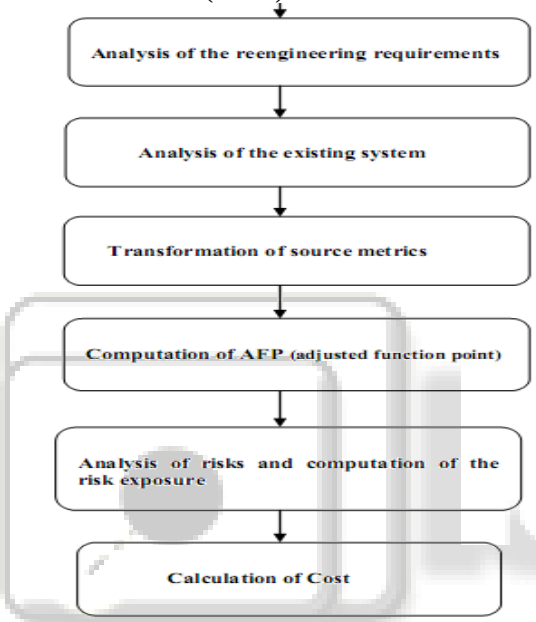

Fig. 1.2: System Architecture

E. Analysis of Risks and Computation of the Risk Exposure

Each software model has some weakness & also has some advantages. There are three dimensions of software risk i.e. technical risk, organization & environmental risk. Risk assessment provides a snapshot of the risk situation & is part of a viable risk management program.

F. Calculation of Cost

In the final step the tool Estimator tool calculates the effort and the time required to make the project. To compute the effort required the adjusted system size (AS) is first multiplied by the quality adjustment factor (QA). The result is then divided by the adjusted productivity factor (AP) to give the basic effort required. If risks are involved this basic effort will be adjusted by the risk Exposure (RE). The final effort is:

Effort = [(AS x QA) / AP] x RE

III. Result Analysis

Consider the following inputs:

- Internal Logical Files (ILF) - 02 and weight low = 7
- External Interface Files (EIF) - 02 and weight avg = 7
- External Inputs (EI) - 03 and weight high = 6
- External Outputs (EO) - 03 and weight low = 4
- External Queries (EQ) - 04 and weight avg = 4

TDI = 42

Then FPA

UFP = 2*7 + 2*7 + 3*6 + 3*4 + 4*4 = 74

VAF = 0.65 + TDI/100 = 0.65 + 42/100 = 1.07

FP = UFP * VAF = 74 * 1.07 = 79.18

Effort in Person Month = FP divided by no. of FP's per month (Using your organizations or industry benchmark)

Schedule in Months = 3.0 * person-month^1/3

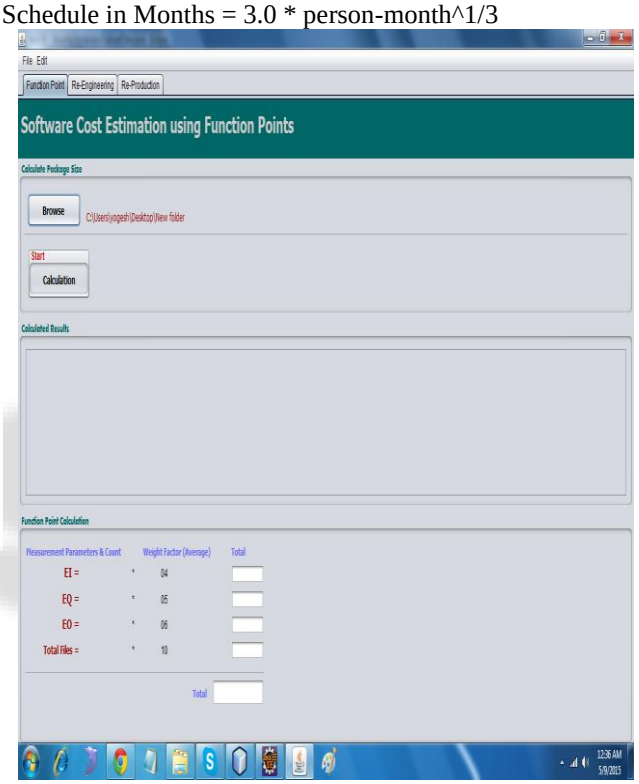

Fig. 1.3: Parameters of Function Points

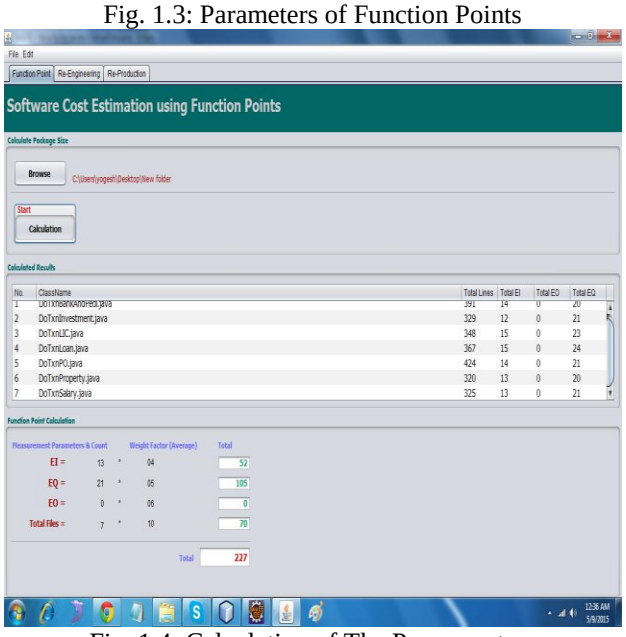

Fig. 1.4: Calculation of The Paperameters

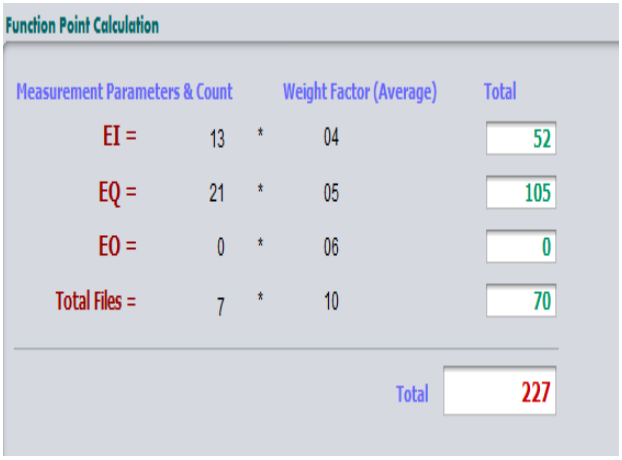

Fig. 1.5: Results of the Methods

Accurately predicting the size of software has plagued the software industry for over 45 years. Function Points are becoming widely accepted as the standard metric for measuring software size. Now that Function Points have made adequate sizing possible, it can now be anticipated that the overall rate of progress in software productivity and software quality will improve. Understanding software size is the key to understanding both productivity and quality. Without a reliable sizing metric relative changes in productivity (Function Points per Work Month) or relative changes in quality (Defects per Function Point) cannot be calculated. If relative changes in productivity and quality can be calculated and plotted over time, then focus can be put upon an organizations strengths and weaknesses. Most important, any attempt to correct weaknesses can be measured for effectiveness.

IV. Conclusion

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 improvement on maintenance effort.

The experience of the author has been that users are reluctant to pay for quality improvement because they are unable to quantify their benefits. The effect of the environment and, in particular, the effect of tools upon productivity must be analyzed. Finally, there is still much to be done in assessing what effect risks have on the project schedule. Here there is a definitive need for more empirical studies. In light of these many open issues, it is clear that the method presented here can only be considered as just another stage in a continual leaning process.

Reference

[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] lbrecht, 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

