International Journal of Technology Research and Management
ISSN (Online): 2348-9006
Vol 1 Issue 2 May 2014

A New Way to Estimate the Size and Effort of Software for Expert User Programming

*Alka Soniya, **Pawan Ratadiya
*,**Bhopal (India)
*soniyaalka16@gmail.com, **pawan_ratadiya1@yahoo.com

Abstract: *- The software system development came in to existence around 60 years ago. Right from the start to until date the software engineering is continuously evolving the new techniques for developing fast, cheap and top quality software. During this paper, we've given an outline of existing size and effort estimates for software. Of these estimates are delineated a lot of or less on their own. Size & effort estimation could be a very talked-about task. Everything revolves around cost, schedule and quality. One such evolving field of software development is estimation models for software size and effort. Software size estimation is one among the most necessary inputs for software cost and effort estimation. Therefore improving the accuracy of software size estimation ultimately results in improving the accuracy of the software effort and cost estimates. These estimates are utilized in staffing, scheduling, planning, budgeting etc. however after we figure these estimates, solely high level project necessities are accessible to us. Using this high level data to provide correct software size estimates could be an extremely difficult task.*

1. INTRODUCTION

Software cost estimating has been an important but difficult task since the beginning of the computer era in the 1940s. The size of software applications have grown in size and importance. Therefore the demand for the correct software size estimation has conjointly grown up.

In the early days of software, computer programs were generally but a thousand machine directions in size or less than thirty function points, only one programmer needed to write down, and also the whole rarely completed in around one month. The entire development costs were typically less than $5000. Although cost estimating was troublesome, the economic consequences of cost-estimating errors weren't terribly serious.

Today some large software systems exceed twenty five million source code statements, typically need technical staffs of a thousand personnel or additional, and the project cycle take more than 5 calendar years to complete. Also the event costs for such large software systems will exceed $500 million.

Therefore, even little errors in cost estimation are very serious indeed. Also if a major proportion of enormous software systems run late, then it'll lead to olympian their budgets. Typically excessive optimism in software cost estimation may be a major reason of overruns, project failures etc..

Now days, software is engine of recent business sector, government sector, and even in military operations. It easy means a typical Fortune five hundred corporation or a state government could turn out hundreds of new applications and modify many existing applications per annum. As a result, software cost estimating is currently a thought activity for every company that builds software.

In addition to the requirement for correct software cost estimates for day to day business operations.

Also the software cost estimates have become a major aspect in litigation. Several authors over the years observed dozens of lawsuits wherever software cost estimates were produced by the plaintiffs, defendants or both.

Figure 1 illustrates the essential principles of recent commercial software cost-estimating tools.

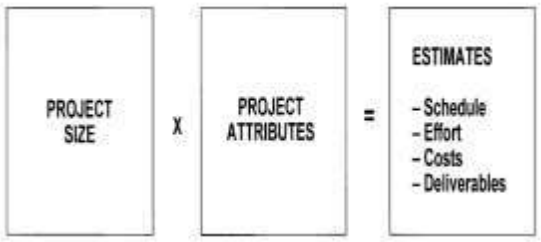

Figure 1 Software-estimating principle [8]

Every form of estimation and each commercial software cost-estimating tool wants the sizes of key deliverables so as to complete an estimate. Size data can be derived in many fashions, including the following [8]: Size prediction using an estimating tool's built-in sizing algorithms. Sizing by extrapolation from function purpose totals. Sizing by analogy with similar projects of known size. Guessing at the scale using "project manager's intuition". Guessing at the size using "programmer's intuition". Sizing using statistical strategies or monte carlo simulation. For agile strategies and those comes using iterative development, sizing of the whole application could also be deferred till the early increments are complete. Even for Agile and iterative projects it's potential to make an approximate prediction of ultimate size simply by comparing the nature of the project to similar projects or using size approximations based on the category, and nature of the software.

2. LITERATURE SURVEY

At the start of the project, there's a lack of information. Because of this lack of data most of the estimation models like- FPA, COCOMO uses a one_size_fits_all approach to calculate size and effort. It doesn't provide correct leads to most of the modern application development. A number of strategies for estimating size are proposed within the literature. A good summary of those strategies may be found in [13, 1, 2, 3] Most of those strategies may be classified into four major categories: expert judgment based, analogy based, group consensus based, and decomposition based.

As is clear from the name, size estimation based on expert judgment takes advantage of the past experience of a professional. The conventional developer is requested to estimate the scale of a project based on the information accessible concerning the project. One of the main benefits of this approach is that consultants can spot exceptional size drivers. The accuracy of such estimate is completely dependent on the experience and memory of the professional. Analogy-based size estimation strategies [4, 5, 6] use one or a lot of benchmarks for estimating the scale of a new project. In such strategies, the characteristics of the new project are compared with the benchmarks. On basis of that comparison, the scale of the new project is adjusted based on the similarities and differences between the new project and also the benchmarks. Pair-wise comparison could be a special case of analogy-based sizing that uses one reference point. This sort of estimation will only be employed when appropriate benchmarks are available. The main advantage of this approach is that it uses relative sizing which prevents most of the issues associated with absolute sizing like personal bias and incomplete recall.

Group consensus techniques like wideband Delphi [13] and planning Poker use a group of people instead of individuals to derive estimates of size. In this technique, estimation activities are coordinated by a moderator who describes the scenario then estimates, and at last compiles the results. At the end of the first round, divergences are discussed and individuals share rationales for their estimation values. More rounds of estimation could also be necessary to succeed in a consensus. The most advantage of those techniques is that they improve understanding of the matter through group

discussion. This iteration and group coordination requires more time and resources than techniques counting on one person.

Decomposition techniques for estimating size use a more rigorous approach. Here two complementary decomposition techniques are available: top-down and bottom-up. Top-down estimation focuses on the product as an entire. Estimates of the size are derived from the worldwide product characteristics and are then allotted proportionately to individual components of the product. The bottom-up estimation focuses on the individual components. This size is estimated for every individual part. The size of the product [7] is then derived by summing the size of the individual components. Since these 2 techniques are orthogonal to every other therefore the benefits of one are the disadvantages of the other. The top-down approach incorporates a system level focus however lacks a detailed basis. The bottom-up approach incorporates an additional detailed basis however tends to ignore overall product characteristics.

3. FUNCTION POINT ANALYSIS

Function point Analysis [8] is an objective and structured technique to measure software size by quantifying its practicality provided to the user. It is based on the necessities and logical design. FPA technique breaks the system into smaller elements so they are often higher understood and analyzed. The FP counts are often applied to development projects, the enhancement projects, and on the prevailing applications as well. The FPA has 5 major segments through that it captures the functionality of the appliance. These are: External Inputs (EIs), External Outputs (EOs), External Inquiries (EQs), Internal Logical Files (ILFs) and External Interface Files (EIFs). initial three are treated as Transactional Function types and last two are data knowledge function Types. Function point Analysis consists of performing the subsequent steps:

- find the type of function point count.
- find the application boundary.
- determine and rate transactional function types to calculate their contribution to the Unadjusted function point count (UFP).
- determine and rate the data function types to calculate their contribution to the UFP.
- Calculate the value Adjustment factor (VAF) by using General System Characteristics (GSCs)
- at end calculate the adjusted function point count

4. END USER PROGRAMMING [9]

End-User Programming system aims to give some programmable system functionality to those who are not skilled programmers. The most successful computer program of all times is the spreadsheet applications. The explanation behind its success is that end users will program it without going into the background details of logic and programming. However, end user programming is rare in alternative applications and wherever it exists sometimes requires going conventional programming, for example AutoCAD provides LISP for customization, and Microsoft applications use Visual Basic. The more convenient mechanism for users is to customize existing applications and build new ones as and when required.

End-user programming is outlined as “Creating an information structure that represents a collection of instructions either by explicit coding or by interaction with a tool. The instructions are executed by a machine to supply the desired outputs or behavior” [9].

End-User Programming are going to be driven by increasing computer literacy and competitive pressures for speedy and user driven information processing solutions. Such trends will force the software marketplace toward having users develop most data processing applications themselves via application generators. The most popular example application generators are spreadsheets, query systems, and inventory systems [11].

End-user programmers who incorporates a good deal concerning their applications domain and comparatively very little about computer science in distinction to the infrastructure developers can typically know a good deal concerning computer science and comparatively very little about applications [12].

Effort estimation for software projects has established to be an elusive and expensive problem in software engineering. The stakeholders expect precise estimates within the early stages of a project. However dependably producing those numbers is extremely difficult and may well be technically impossible. Author boehm et al. report that estimating a project in its initial stages yields estimates which will be off by the maximum amount as a factor of four. Even at the purpose when careful specifications are produced, the professional estimates are expected to be wrong by 500th. [10]

The expert user programming also affects the size of software. By as well as it within the list of general system characteristics, we've got created a provision for taking user facilities into consideration, at the time of estimating the size of a project. it's clear that our proposed FPA provides additional accurate size estimates. It will narrow the gap between size calculable and actual size. Which can lead to additional accurate effort and cost estimates. that ultimately results in increased productivity and proper staffing , planning, scheduling.

5. CONCLUSION

In this paper, we have presented a survey of some popular software size estimation techniques. The advantages and disadvantages of each technique are mentioned. It'll facilitate in coming up with a additional accurate software size estimation technique.

References

[1]. Verner et al, A Model for Software Sizing", Journal of Systems and Software, IEEE Software, pp. 173-177, July 1987.

[2]. Albrecht et al, 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]. Contributions, Costs and Prospects for End-User Development, Alistair Sutcliffe, Darren Lee & Nik Mehandjiev

[10]. 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) .

[11]. Boehm B.W. (1981), "Software Engineering Economics", Prentice-Hall, Englewood Cliffs, NJ, 1981.