ISSN No. 0976-5697

Volume 7, No. 1, January-February 2016

International Journal of Advanced Research in Computer Science

REVIEW ARTICLE

Available Online at www.ijarcs.info

A Review: Software Metrics and Effort Estimation in Aspect Software Development Program

Aarti Hans
Department of Computer Science and Technology NCU
University Gurgaon,
Haryana, India

Sonal Gahlot
Department of Computer Science and Technology NCU
University Gurgaon,
Haryana, India

Abstract: In this paper, we present a survey of software effort estimation andSoftware size estimationwhich is key characteristicshas been designated for the determination of limiting the examinestandards. In SPMfield, managementis the always needed for the organization knowledge purpose. According to our topic, we had analyticallystudied2bestuseful techniques that are software effort estimation andsoftware size estimation. A modestanalysis of our models with currentuniversal models is presented in this survey paper.

Keyword: Software cost management, SLOC, software metrics, etc.

I. INTRODUCTION

In effort estimation, various software cost estimation techniques are available to predict effort & cost for software developers. The estimation technique is not modesttechnique to sort for thepreciseestimation of the determinationessential to progress a software organization. Software metrics give the information about project based measurable its characteristics & also provide good criteria for measuring the similarity of several software products. In this survey paper we will make originalestimations on the base of a high equaloperator requirement.

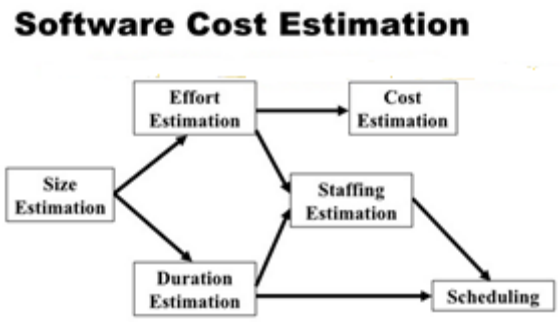

Figure 1: An architecture of Software cost estimation

The aimof the estimation of software is tocalculategiven metrics & give a publicusual in standings of software that is metrics &characteristics that are connected to cost approximation [2, 3].The Software cost estimation scheme is a fundamental for theapproval or rejection of the progress of software project. Numerous Software cost estimation techniques hasbeen in training with their personalpowers and boundaries. Presently the object-oriented method for

Software cost estimation is created on the Line of Code (LOC), its function points, functions and classes etc.

A. Process &Size oriented metric in LOC

Most commonly used size oriented metric is LOC, this software is used to metric measure and lines of code of the programming code of the computer source program. The main objective of LOC is to predict the programming productivity and effort to develop the program. It is widely used technique for estimating lines of source code per programmer month [2]. This approach was first developed at the time when most of the programming was performed on line-oriented languages e.g. Assembly, FORTRON and COBOL. But with the advent of the era of C++ and Java, where program may consists of macros, declaration statements, comments, etc. and other related issues made the usability of this technique problem some. And the most common anomaly related to LOC is, when will be the one program language needsto extra lines of code than another to device the similar functionality, efficiencyevaluations will be unusual [2].

B. Function of oriented metric

Function of oriented metricis use for the measurement to complete functionality delivery by the computer system as control factor. Function Point (FP) is mostly used function-oriented metric [4, 5, and 6]. Four models based on FP are ISO certified [7]. The anomaly mentioned above existed due to the implementation language is avoided by introducing "functionality" as the deciding factor. Computation of the Function Point is constructedfor thefeatures of the software's factsarea and its difficulty [1]. FP investigation was the first author namely Allan Albrecht that presentedthe function point in 1979. Functional requirements of users are categorized into five measurement parameters i.e.

1. External inputs
2. External outputs
3. External inquiries

4. Internal logical files and

5. External logical files.

Ones requirements are categorized, then their respective complexity is associated with each functional requirement, it is based on subjective judgment of the organization. Formula for computing the Function Point = count total X [0.65 + 0.01 X∑ (Fi)] Where, calculation total is sum of totallyaccessesgotten from the practicalnecessities, Fi (1 to 14) are importancealteration factors. The constant standards in calculation and the premiumissues that are functional to evidenceareatotals are resolute empirically.

II. SLOC

Source Lines of Code (SLOC or LOC) is one of the most widely used sizing metrics in industry and literature. It is the key input for most of major cost estimation models such as COCOMO, SLIM, and SEER-SEM. Although the SEI and the IEEE have established SLOC definitions and guidelines to standardize counting practice, inconsistency in SLOC measurements still exists in industry and research. This problem causes the incomparability of SLOC metric among organizations and the inaccuracy of cost estimation

The estimated SLOC in a proposed software system is used as input to many cost estimation models as described previously in this report. But how are the SLOC accurately estimated in the early stages of the software development lifecycle? A SLOC estimate of a software system can be obtained from experience, the size of previous systems, the size of a competitor's system, and breaking down the system into smaller pieces and estimating the SLOC of each piece.

Putnam suggests that for each piece, three distinct estimates should be made:

- Smallest possible SLOC – a
- Most likely SLOC – m
- Largest possible SLOC – b

Then the expected SLOC for piece Ei can be estimated by adding the smallest estimate, largest estimate, and four times the most likely estimate and dividing the sum by 6. This calculation is represented by the following formula:

$$E_i = \frac{a + 4m + b}{6}$$

The expected SLOC for the entire software system E is simply the sum of the expected SLOC of each piece:

$$E = \sum_{i=1}^{n} E_i$$

Where n is the total number of pieces.

Requirement of Software Effort Estimation in Project

In software organization cost, time & accurate powercompulsory to progress a software etc. is a most challenging part of any organization. The determinationapproximation model can be on paper in the resultingoverall nonlinear form

$$PM = A * Size^{E} * \prod_{i}^{p} EM_i$$ Where

PM = power or effort estimate in person months

A = multiplicative constant

Size = predictable size of the software or system, dignified in KSLOC. Aggregate size has limitedpreservativeproperties on the effort. Size is mentioned to as an additive factor.

EM = effort multipliers. These aspects have universaleffects on the cost of the completeorganization.

E = it is well-defined as a function of scale factors.

Similar to the energy or effort multipliers, the scale factors have universalpropertiesthrough the scheme but their special effects are related with the size of developments. They have neweffect on the cost of larger-sized schemes than smaller-sized schemes. The estimation process itself is not very well defined at many software industries. In any organization an accurate estimation is the first step to an effective estimate. If size of project increase, then necessary accuracy is a very essential that is difficult to estimate.Estimating the effort with a large value of consistency is a difficult which has not been resolved yet.

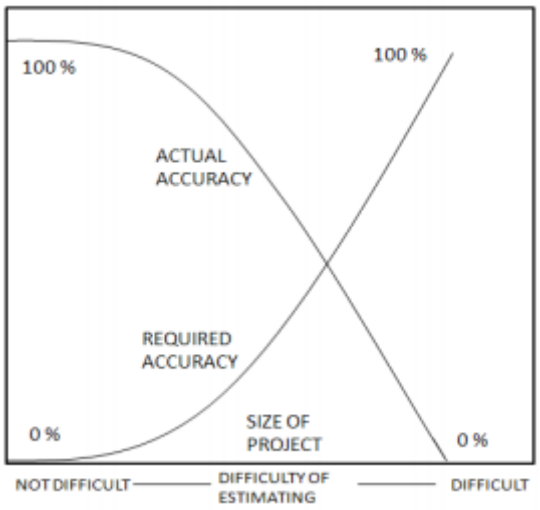

Figure 2: Accuracy of Estimating

III. RELATED WORK

James C. Benneyan [1**]** in this research author proposed the SPC (statistical process control)approaches have establishedthe very developingimportance in the healthcare unrestricted to help advance clinical and organizationalprocedures.

Anton Ellis et al. [4]projected a technique forrepresenting object oriented source code metrics onto the sub-characteristics of maintainability declared in ISO 9126.

Oman and Hagemeister [5] in this paper Maintainability Index (MI) proposed that work as an accuratelyconcludes the maintainability of software schemeconstructed upon the position of the source code.

Welker and Oman [6], suggested measuring maintainability in terms of cyclomatic complexity, lines of code(LOC) and lines of comments. The intending to this object is to provethat how automatic software maintainability investigation can be used to lead a software-related assessmentmanufacture. Author has used metrics-based software maintainability simulations to 11 manufacturing software systems and used the outcomes for fact-finding and process-selection decisions. The outcomesspecify that automatic maintainability calculation can be used to maintenance buy-versus-build decisions, pre- and post-reengineering study.

Hayes et al. [7]projected a perfect approach that approximate Adaptive software maintenance effort in terms of difference lines of code (DLOC) i.e. number of added, deleted and updated lines.

Polo et al. [8], has developed for the use of number for adaptationrequirements, mean effort per variationapplication and type of improvement to survey maintainability.

In one more research Hayes and Zhao [9], projected a maintainability model that classifies software components as "easy to preserve" and „not easy to preserve". The model assistances the originators to classify the components those are not easy to keep up, earlierincorporating them. From the review of writings it has been perceived that numerousinvestigatorsplannedsome models for maintainability approximation, but in maximum of these revisions, maintainability approximationrest on on the proceduresoccupied after the coding phase. Because of this, maintainability forecasts are complete in the finalphases of SDLC, and it developed very hard to recover the maintainability at that phase.

IV. CONCLUSION

In this paper we surveyed on software estimation method, providing numerous accepted estimation models. The cost of software maintenance accounts for a large portion of the overall cost of a software system. Therefore, we need to effectively manage software maintenance activities. As in the conventional software systems, we can apply measurement based approach to estimating and predicting maintenance efforts.It was observed that little work has been done in maintainability of the web applications and open source software. The values of understandability, modifiability and maintainability are of immediate use in the software development process.

V. FUTUREWORK

In future work, we can implement a semi-formal description of a fixed of undeveloped metrics to size crosscutting in structures that use characteristics. Our metricsclassify features inside a crosscutting range that ranges from varied to similar according to the comparative number and types of crosscuts structuresappliance.
These classification assistancesmeasuring the real use of aspects for feature application and delivers a measurableoutline to measure and analyze the effect of aspects for product line growth. We will put on our metrics to a nontrivial product line case study applied using Aspect oriented and communicate our results to the constantvaluation using MATLAB tool.

VI. REFERENCES

[1] James C. Benneyan, ―Design, use, and performance of statistical control charts for clinical process improvement‖, North-eastern University, Boston MA, September 16, 2001.

[2] Patrick Naughton& Herbert Schildt "java: The complete reference", McGraw-Hill Professional, UK, 2008.

[3]Mike ÓDocherty, "Object -Oriented Analysis and Design Understanding System Development with UML 2.0", John Wiley & Sons Ltd, England, 2005.

[4] P. Antonellis, D. Antoniou, Y. Kanellopoulos, C. Makris, E. Theodoridis, C. Tjortjis, and N. Tsirakis, "A Data Mining Methodology for Evaluating Maintainability According toISO/IEC-9126 Software Engineering Product Quality.Standard," Proc. 11th IEEE Conference onSoftware Maintenance and Reengineering (CSMR2007), 21 – 23 Mar.2007, Amsterdam, Netherlands, 2007.

[5] P.W. Oman and J.R. Hagemeister, "Construction and Testing of Polynomials Predicting Software Maintainability," Journal of Systems and Software, vol. 24, no. 3, pp. 251- 266, 2013.

[6] K.D. Welker and P.W. Oman, "Software Maintainability Metrics Models in Practice," Journal of Defense Software Engineering, vol. 8, no. 11, pp. 19 - 23, 1995.

[7] J.H. Hayes, S.C. Patel, and L. Zhao, "A Metrics-Based Software Maintenance Effort Model," Proc. 8th European Conference on Software Maintenanceand Reengineering (CSMR'04), 24 – 26 Mar. 2004, pp. 254 – 258,IEEE Computer Society, 2004.

[8] M. Polo, M. Piattini, and F. Ruiz, "Using Code Metrics to Predict Maintenance of Legacy Programs: a Case Study," Proc. of InternationalConference on Software Maintenance, ICSM 2001, pp. 202-208, IEEE Computer Society, Florence Italy, 2001.

[9] J.H. Hayes and L Zhao, "Maintainability Prediction: a Regression Analysis of Measures of Evolving Systems," Proc. 21st IEEE InternationalConference on Software Maintenance, 26 - 29 Sept. 2005, pp. 601 -604, 2005.