Available online at www.sciencedirect.com

SciVerse ScienceDirect

Procedia Technology 4 (2012) 761 – 765

Procedia Technology

C3IT-2012

Story Points Based Effort Estimation Model for Software Maintenance

Jitender Choudhari[a], Dr. Ugrasen Suman[b]

[a] *Medi-Caps Institute of Technology and Management, Indore, India*
[b] *School of Computer Science and IT, Devi Ahilya University, Indore, India*

Abstract

Software is developed with prior requirements and it is maintained continuously with rapid progresses in domain, technology, economy, and other fields. Software maintenance is vital and it requires more efforts and resources than development phase. There are fewer methods available for maintenance effort estimation. An iterative maintenance life cycle using extreme programming is a model for software maintenance based on agile methodology that uses RC stories as requirement artifacts. Agile methodology uses analogy and expert opinion based estimation techniques such as planning poker. But in software maintenance, historical data and experts may not be present or accurate. Therefore, a heuristic method is required for the calculation of maintenance efforts. In this paper, we propose a Software Maintenance Effort Estimation Model (SMEEM) for software maintenance estimation. The SMEEM technique uses story points to calculate the volume of maintenance and value adjustment factors that are affecting story points for effort estimation. The proposed model will be illustrated with various types of maintenance projects. The model generates accurate and effective results as compared to other software maintenance effort estimation models. This model is applicable only for agile and extreme programming based maintenance environment.

Keywords: Story point; software maintenance; RC story; maintenance project effort estimation; SMEEM.

1. Introduction

Software maintenance is the process of modifying a software product after delivery to correct faults or to implement new functional requirements. Software maintenance helps to improve performance, reliability, and adaptability for change request in the product in the modified environment. It is categorized as adaptive, corrective, preventive and perfective [1]. The software maintenance planning can involve activities such as duration, staff, costs, effort and size estimation to keep control of the process. It reduces the risks and the inefficiencies related to the maintenance work. Most organizations are concerned about the maintenance costs of software, which is increasing continuously [2]. Therefore, it is becoming necessary for these organizations to control their software maintenance efforts and costs effectively. Software maintenance is not acknowledged with consideration and admiration that it deserves in field of software estimation models. The experts have observed that maintenance costs are difficult to estimate, due to different products, process, and internal organizational factors related to the product.

doi:10.1016/j.protcy.2012.05.124

There exist fewer effort estimation models for software maintenance as compared to software development. The software maintenance effort can be estimated by existing estimation models such as COCOMO 2.0 reuse model, FP and ACT model. The different estimation models take SLOC, FP and object point as sizing units for determining maintenance effort. ACT includes source code which is new, changed or reused lines delivered [3, 4]. Function points components such as, the application functionality, user requirements, conversion functionality, and 14 Value Adjustment Factors (VAF) with varying intensity range based software characteristics are used to determine software maintenance effort estimation [5, 6]. This approach is known as Function Point (FP) version of software maintenance. COCOMO 2.0 reuse model is COCOMO version for software maintenance that uses KSLOC as sizing unit. It includes 17 effort VAFs such as required software reliability, database size, product complexity, required reusability, documentation, etc in multiplicative manner. It uses five scale values for each factor such as precedentedness, development flexibility, architecture or risk resolution, and team cohesion and process maturity.

Organization with chronological data for annual maintenance cost estimation uses ACT model which is derived originally from COCOMO model. There are some basic limitations in ACT model such as, results of estimation are likely to be unreliable if the systems are completely new and has no historic basis for estimating the ACT. ACT has no way to determine scientific quantitative aspect for representing software maintainability and thus introduces a significant risk into the model. FP models VAF are not applicable for the maintenance environment project. COCOMO 2.0 reuse model is treated as management model. The modification factors are determined during planning stage of maintenance project and not before that. This is treated as a restriction of the model. The above existing models of maintenance estimation are based upon traditional software development methodology and it may produces realistic result for same methodology. If the above mentioned models are applied in agile and extreme programming based maintenance project, then these models can provide an unrealistic results. Agile methodology uses analogy and expert opinion dependent estimation techniques such as planning poker but in lack of past data the method becomes unpredictable [7, 8].

Iterative maintenance life cycle using extreme programming is a process model for software maintenance [9]. It uses RC story as a requirement artifact and it should be written by the end users of software for maintenance [10]. It provides end user collaboration and simplifies requirement engineering process of software maintenance. The frequent tribulations such as poor visibility of the project, lack of communication in maintenance process can be resolved by RC story format. In case of maintenance using iterative maintenance life cycle using extreme programming [9, 10] there is a need of algorithmic approach for maintenance estimation. In this study, Software Maintenance Effort Estimation Model (SMEEM) is proposed which is based on the story point.

In this paper, Section 2 discusses the proposed VAFs that can affect effort of a maintenance project. The working procedure of SMEEM model is described in Section 3. The illustration of proposed model is evaluated with results in Section 4. Section 5 covers conclusion and future scope.

2. Factors Affecting software maintenance estimation

A maintenance effort is difficult to estimate and is affected by factors such as documentation quality, structuredness, modularity, reusability of existing software modules, conformity with software engineering standard, familiarity with programming language, changeability/readability of programming language and domain knowledge.

System understanding has undeviating influence on the quality of software document. In software maintenance process, documentation of the system artifact such as, architecture, high-level design, low-level design, test plans etc., are aimed to offer comprehensive, clear and short information about the system [11]. System's functionality and their relationships are described by documentation factor. Software abstract in a short duration can be achieved with the help of system documentation thereby improving its maintenance. Structuredness is the desired attribute in software maintenance, and is used in all of its sub tasks. It shows the effects of the modified part on the other parts in the system. A good structure is a vital aspect for software quality [12]. It is a feature that presents a specific pattern of organization of its

interdependent components. Structuredness is representative of ordered and systematic software design. Structuredness minimizes complexity of software component by minimizing coupling between modules and improving software analyzability which is identified by the scope of error inspection method. Modularity is a characteristic that plays an important role in maintenance process by dividing software into numerous manageable components that communicating with each other. It allows understandability of a system component wise instead of understanding the whole system. It can help to modify components of a system independent from other components. The general metrics of software design modularity consists of coupling, cohesion, and separation of concerns [13]. Errors or function can be identified effortlessly and proficiently, thereby reducing maintenance efforts. Modularity approach simplifies testing and increases errors inspections and cause of the errors.

Software reusability improves efficiency, value and maintainability of software products by implementing or updating software systems using existing software systems or legacy systems assets [14]. A reusability approach may possibly save up to 20% of development costs. In maintenance process, different activities such as program comprehension, readability and modification are supported by conformity with software engineering standards factor. It is considered as the productivity factors that reduces maintenance effort. Software maintainer's skills entail awareness with programming language. This factor includes experience of technology of existing system so that maintenance team can easily understand the system. Software maintenance is a knowledge intensive task. Industrial studies suggest that 40% to 60% of the software maintenance effort is reduced by proper understanding of application domain during requirement analysis of the system. The changeability or readability of source code is software quality factor, which is increased by modularity and structure of source code. It is also affected by the programming language of existing system.

3. Software Maintenance Effort Estimation Model (SMEEM)

The software industry has many approaches for software maintenance effort estimation based on traditional software development estimation process. We have proposed a SMEEM to predict maintenance effort in terms of Story Point (SP). It incorporates the VAF such as documentation quality, structuredness etc., which are discussed in Section 2. The proposed estimation approach uses RC stories and old software as input and performs all the phases in the proposed SMEEM framework. The SMEEM divides software maintenance estimation process into five steps described in the following subsections.

3.1. Factor Count (FC) Computation

Intensity of VAF of maintenance project are identified on the grade of low, medium and high (in scale of range from 0 (Low) to 5 (High)). The grades are decided based upon the importance in the maintenance project. The computation of the summation of all grades of various factors for a project denoted as Factor Count (FC) using following equation.

$$\text{Factor Count (FC)} = \sum_{i=1..7} VAF_i \quad (1)$$

3.2. Story Point Assignment

Assign Story Point (SP) to each story based upon the size using planning poker technique by maintenance team.

3.3. Adjusting Story Point

After story point (SP) assignment using planning poker technique, Adjusted Story Point (ASP) can be computed with the help of following equation:

$$ASP = SP - [\text{Critical Factor} * ((FC * Max\_VAF) / (FC + Max\_VAF))] \quad (2)$$

Where, value of Critical Factor (CF) shows the type of project such as web based application, MIS and critical project. FC is summation of VAF which is derived from equation (1) and Max_VAF is summation of highest intensity of VAF.

3.4. Calculate Size of Maintenance (SoM)

The Size of Maintenance (SoM) will be computed using following equation.

$$\textit{Size of Maintenance (SoM)} = \sum_{i=1..n} ASP_i \quad (3)$$

Where, ASPi is ASP of i^{th} RC story and n is total number of RC stories of maintenance project.

3.5. Calculate Duration of Maintenance (DoM)

Calculate Duration of Maintenance (DoM) by SoM and ASP developed in iteration.

Cost of a single RC story or component of a maintenance project can be computed from the above DoM and current imbursement paid to maintainers. At the end summation of all RC story cost provides overall maintenance cost for a project.

4. Evaluation of SMEEM

Here, we have considered RC story as a requirement artifact in a maintenance project with having story point 20 as an example. Maintenance project category includes web based application, MIS and critical project with CF .60, .35 and .20, respectively. The performance of proposed approach of maintenance has been investigated for various intensity levels of VAF. The values of ASP have been computed using the proposed model at different intensity levels of VAF for various project categories. Table 1 shows the ASP computations for an example having Story Point (SP) 20. For this example, in a critical project with specified input of CF is 0.20, and ASP has been computed as 16.88. It has been observed that inclusion of VAF and CF in such projects changed the estimated ASP to approximately 84.4 % of its previous SP and estimated more realistic values. The performance measurement of SMEEM is shown in Figure 1 with three values Factor Count (FC), which is summation of VAF as a X- axis, ASP as Y-axis both are with respect to Critical Factor (CF), which show project category.

Table 1. Computation of ASPs with an example of Story point 20

Factor Count (FC)	Critical Factor (CF) = 0.60	Critical Factor (CF) = 0.35	Critical Factor (CF) = 0.20
7	16.50	17.96	18.83
10	15.33	17.28	18.44
14	14.00	16.50	18.00
18	12.86	15.84	17.62
21	12.12	15.41	17.37
25	11.25	14.89	17.08
28	10.67	14.55	16.88
31	10.13	14.24	16.71
35	09.50	13.87	16.50

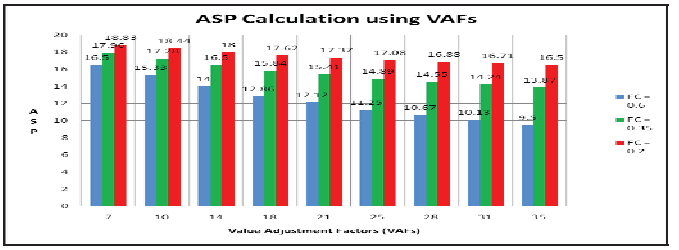

Fig. 1. Graph Showing Result of ASP Computation

5. Conclusions and Future Scope

The existing models of maintenance estimation are based upon traditional software development methodologies which produce realistic result for same methodology. If existing models are applied in agile and extreme programming based maintenance project, then these models can provide an unrealistic result. In case of maintenance using iterative maintenance life cycle using extreme programming that uses RC stories as requirement artifacts, there is a need of algorithmic approach for maintenance estimation. In this study, SMEEM is proposed, which is based on the story points to calculate the volume of maintenance. The SMEEM of effort estimation incorporates VAF with different intensity levels. The proposed model is illustrated with various types of maintenance projects. It is designed to help the project manager in charge of software maintenance to calculate the estimated software maintenance effort in terms of ASP, size, cost and duration. It generates the more realistic and precise estimation results. This model is applicable only for agile and extreme programming based maintenance environment. The future work will focus on refinement of proposed SMEEM using other factor of agile and XP environment.

References

1. B.P. Lientz, B.E.Swanson, Software maintenance management, Addison- Wesley, 1980.
2. W. Harrison, C. Cook, Insights on improving the maintenance process through software measurement, In: Proceedings of the International Conference on Software Maintenance, IEEE Computer Society Press, San Diego CA, (1990) 37–45.
3. B.W. Boehm, Software Engineering Economics, Prentice-Hall, (1981) 596–599.
4. H. Schaefer, Metrics for optimal maintenance management, In: Proceedings of the Conference on Software Maintenance, IEEE Computer Society Press: Washington, (1985) 114–119.
5. AJ. Albrecht, AD/M Productivity Measurement and Estimate Validation, IBM Corporation: New York, 1984.
6. C.R. Symons, Function point analysis: Difficulties and improvement, IEEE Transactions on Software Engineering,14 (1) (1988) 2–11.
7. M. Cohen, Agile Estimation and Planning, Pearson Low Price Edition Asia, 2006.
8. C. Stiendl, P. Krogdahl, Estimation in Agile Projects, Proceedings of Best Practices in Project estimation Conference at IBM Academy of Technology, 2005.
9. J. Choudhari, U.Suman, Iterative Maintenance Life Cycle Using eXtreme Programming, In: Proceedings of the International Conference on Advances in Recent Technologies in Communication and Computing (ARTCom), IEEE Computer Society, (2010) 401 – 403.
10. J. Choudhari, U.Suman, Designing RC Story for Software Maintenance and Evolution, Accepted in Journal of Software, Academy Publisher, 2011.
11. S.C.B.D. Souza et al, A study of the documentation essential to software maintenance, Presented at The Proceedings of the 23rd Annual International Conference on Design of Communication: Documenting & Designing for Pervasive Information, Coventry, United Kingdom, 2005.
12. M.H. Ammann, R.D. Cameron, Measuring program structure with inter-module metrics, In: Proceedings of the eighteenth annual international computer software and applications conference, (1994) 139-144.
13. C. Yuangfang, S. Huynh, An evolution model for software modularity assessment, In: Proceedings of the 5th International Workshop on Software Quality,(2007) 3-3.
14. Gill, S.Nasib, Importance of Software Component Characterization for Better Software Reusability, ACM SIGSOFT Software Engineering Notes, 31 (1) (2006) 1-3.