Available online at www.sciencedirect.com

SciVerse ScienceDirect

Procedia Technology 4 (2012) 716 – 722

Procedia Technology

C3IT-2012

Estimation of Software Development Effort from Requirements Based Complexity

Ashish Sharma[a], Dharmender Singh Kushwaha[b]

[a] *Department of Computer Engg. & Applications, GLA University, Mathura 281 406, India*
[b] *Department of Computer Science & Engg., Motilal Nehru National Institute of Technology, Allahabad, India*

Abstract

Software development effort estimation deals with predicting the effort required to develop quality software. The efficient software development requires accurate estimates, because inappropriate estimates causes' trouble during the implementation of software processes. Hence, this paper aims to propose a measure for the estimation of software development effort (SDE) on the basis of requirement based complexity of yet to be developed software. The requirement based complexity has its basis on software requirements specification (SRS) of the proposed software, in order to carry out a systematic and accurate estimation of SDE. For validation purpose, the proposed SDE measure is also categorically compared with various other established SDE estimation practices proposed in the past like algorithmic models, function point count, use case point and lines of code (LOC). The result obtained validates the claim that the proposed SDE measure is systematic, comprehensive one and compares well with other prevalent SDE measures. Hence, it is even more useful because the complexity and SDE estimates are obtained at very early stage of software development life cycle (SDLC) as compared to other software development effort practices proposed in the past.

Keywords: Improved requirement based complexity, Requirement based development effort function point, IEEE-830:1998, Requirement based development effort, Software Requirement Specification (SRS)

1. Introduction

Software development effort estimation is the process of calculating the most realistic value of required effort in person-month to develop or maintain the yet to be developed software on the basis of inputs like software requirements, function points, size, use case points etc. Further, the effort estimates are also used as input to project plan, iteration plan, budgets, investment analysis, pricing processes and bidding rounds. Most of the research has focused on either construction of formal software effort estimation models or considers use case or function points for the estimation of size or development effort, but does not consider the SRS of the software for such computation. Therefore, there is a vital need to find a method through which the software development effort can be estimated on the basis of requirement based complexity of the proposed software.

doi:10.1016/j.protcy.2012.05.116

2. Related Works

This section presents a survey of leading papers describing the work carried out so far in the area of software development effort estimation. Barry Boehm [1] aims at estimating the development effort from famous COCOMO - I, for small to medium sized software projects. The value of development effort depends on size of software and empirically determined constants. Further, Barry Boehm et. al. [2] also discusses the revised version of COCOMO-I as COCOMO-II that uses some fixed values for one constant and the other constant depends on the scaling factors. Yinhuang Zheng et.al. [3] computes development effort for programming measurement and estimation on the basis of a constant 5.5, as multiplier for the development effort. Stephen Mc Donell [4] estimates the development effort on the basis of data collected from organization that captures environmental factors and also considers differences among the given projects. Albrecht and Gaffney [5] discusses about function point as a unit of measurement to express the software functionality. Matson et. al. [6] uses function point for the software cost estimation with a very high constant factor i.e. 585.7 for the calculation of development effort. Clemmons R.K. [7] discusses the functional scope of the project by analyzing the contents on the basis of use case measures Magne Jorgenson [8] attempts to provide six criteria for evaluation of the development methods like - automation, comprehensive assessment, objectivity, specification, testing and validity for the software to be developed. Bernard et. al. [9] discusses the result of two case studies and two experiments to show the impact of effort estimates on software project. It also works on pre-planning of effort estimation and proposes that too low effort estimates lead to less effort and more errors compared with more realistic effort estimates. Noureldin AZ Adem and Zarinah M [10] considers both functional and non functional requirements and discusses about an automated tool to estimate the size of software projects on the basis of two processes namely - goal and scenario based requirements elicitation technique and text based function point extraction guidance rules. IEEE Computer Society [11] explains the IEEE recommended practices for the correct and appropriate way of writing software requirement specification (SRS) document. This is IEEE 830:1998 standard. Geoffrey K Gill and Chris F. Kemerer [12] discusses about complexity density ratio as a useful predictor for software maintenance productivity that is derived on the basis of cyclometric complexity. Charles R Symon [13] discusses about function points analysis (FPA) and compares the original FPA with FP mark-II, as an improvement. The paper [14] discusses about IEEE standard for software productivity measurement on the basis of effort and lines and code. Sharma Ashish and Kushwaha D.S. [15,16] proposes and establishes the improved requirement based complexity (IRBC) measure obtained on the basis of SRS of the proposed software. The paper also proposes an object based semi-automated model for the tagging and categorization of elicited software requirements. Maurice J Halstead [17] discuss about a measure based on the principle of count of operators and operands and their respective occurrences in the code for the estimation of difficulty, effort and time. Kushwaha D.S. and Misra A.K. [18] discuss CICM and modeling of test effort based on component of the shelf (COTS) & component based software engineering (CBSE). The [19] discusses about importance of defining the user requirements and their impact on software development process.

3. Computation of an Improved Requirement Based Complexity (IRBC) from SRS

It has been established that the complexity of the software has a direct bearing on the required amount of effort [18]. For systematic, planned and accurate estimation of software development effort, it is necessary to compute the improved requirement based complexity of the proposed software first. The details of computation of IRBC are available in [15, 16]. However figure 1 shows a procedure for the estimation of IRBC on the basis of SRS of the proposed software written IEEE-830:1998 [11] format. The next section discusses about the application of IRBC for the estimation of software development effort for yet to be developed software.

1. [Input Output Complexity (IOC)] $IOC = IC + OC + SC$
   IC: Input Complexity, OC: Output Complexity, SC: Storage Complexity
2. [Requirement based Complexity (FR)] $FR = \sum_{i=1}^{n}(Functionality)i \sum_{j=1}^{3}(Sub-\Pr ocess\_Decompositon)ij$
3. [Non-Functional Requirement based Complexity (NFR) using ISO-9126]

$$NFR = \begin{cases} \sum_{i=1}^{8}(Attribute)_i \sum_{j=1}^{m} X\,(Subattribute)_{ij} \\ 1 \text{ otherwise for inbuilt quality attributes} \end{cases}$$

4. [Requirement Complexity (RC)] $RC = FR + NFR$
5. [Product Complexity (PC)] $PC = IOC * RC$
6. [Personal Complexity Attributes (PCA)] $PCA = \prod_{i=j=1}^{5}(AttributeV\ \ alue)ij$
7. [Design Constraints Imposed (DCI)] $DCI = \sum_{i=0}^{n}(Constra\,\text{int}\_type)i \sum_{j=0}^{6}(Number)ij$
8. [Interface Complexity (IFC)] $IFC = \sum_{i=0}^{n}(ExternalInterfaces)i \sum_{j=0}^{m}(Number)ij$
9. [Software deployment location Complexity] $SDLC = \sum_{i=1}^{n}(LC)i \sum_{j=0}^{m}(UCC)ij$

$\underline{n}$

Figure 1: Procedure for the computation of IRBC

4. Proposed Approach for the estimation of Software Development Effort

Accurate estimation of software development effort (SDE) is a challenge for every software project, because it has a strong impact on cost, schedule, functionality and quality of the software to be developed. Hence a framework for the estimation of proposed requirement based software development effort (RBDEE) is shown in figure 2.

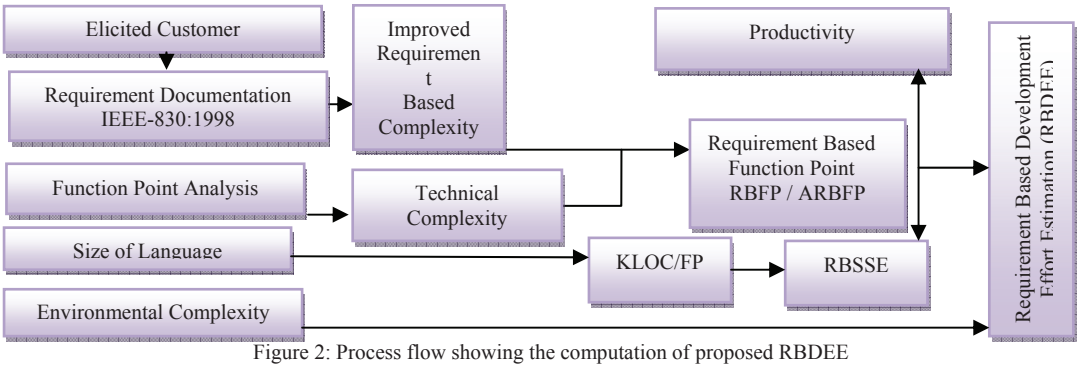

Figure 2: Process flow showing the computation of proposed RBDEE

The following sub-sections elaborate on methodology for the estimation of software development effort from IRBC measure.

4.1 Requirement Based Development Effort Function Points (RBDEFP)

Since IRBC comprises of an exhaustive set of parameters and contributing complexity measures, that encompasses all the parameters that is required to compute function point analysis [13] of any software. Hence IRBC is used as a principal component for the estimation of RBDEFP. Also to make the software development effort estimation more precise and practically possible, technical complexity factors (TCF) as proposed in [13] are also considered that consist of fourteen factors to provide general application characteristics to the proposed software. These characteristics have impact on software productivity relating to various technical issues pertaining to proposed software development. The need and applicability of these factors are determined on the basis of degree of influence (DI) that ranges from zero (not present or no influence) to five (strong influence throughout). The value of TCF [13] is computed by summing the score of fourteen different complexity factors on the basis of their degree of influence (DI) and is mathematically represented as:

$$TCF = 0.65 + 0.01 \text{ X } \sum Fi \qquad (1)$$

Where Fi shows the required and applicable factors for the proposed software. Finally, the obtained IRBC and TCF serve as basis for the estimation of RBDEFP. The calculation of function point is dimensionless, on arbitrary scale. The function point measure isolates intrinsic size of the system from environmental factors, and facilitates study of factors, that influence productivity [13]. It is also observed that all the attributes are functionally dependent on each other. Hence, RBDEFP is expressed as:

$$RBDEFP = IRBC \text{ X } TCF \text{ function points} \qquad (2)$$

4.2 Requirement Based Software Size Estimation (RBSSE)

Size estimation for the proposed software is language dependent. High level languages require fewer lines of code to implement per function point than that of low level languages. Analyzing the functional requirements and estimating function points on the basis of functionality is more precise than describing the source lines of code (SLOC) [5]. Geoffrey and Kemmerer [12] consider various levels of languages for software development and also maps the languages with function points on the basis of SLOC required to implement per function point, this becomes the size of language. Hence for precise estimation of software development effort, RBSSE is dependent on RBDEFP and size of language. This is mathematically expressed as

$$RBSSE = (RBDEFP \text{ X Size of language})/ 1000 \text{ KLOC} \qquad (3)$$

Further, in order to estimate the productivity of software developer, IEEE standard 1045, software productivity measurement [14] describes the software productivity in terms of effort combined with counts of lines of code or function points. It is assumed that: (a) more complex system is harder to maintain, and (b) that system suffers from entropy and becomes more chaotic and complex as it goes on [19]. The productivity [12] of the proposed software in reference to the software complexity is expressed as:

$$Productivity = 5.52+0.346 \text{ X KLOC} \qquad (4)$$

4.3 Requirement Based Development Effort Estimation (RBDEE)

In order to compute RBDEE for the proposed software, it is necessary to consider various derived contributing factors related to SDE estimation like RBSSE, productivity, and environmental

complexity factors (ECF). Firstly, the initial requirement based software development effort ($RBDEE_i$) is calculated on the basis of RBSSE and productivity of the proposed software and later, the final requirement based software development effort ($RBDEE_f$), is calculated on the basis of $RBDEE_i$ and applicable environmental complexity factors (ECF) [7]. In order to clearly understand the entire computation procedure for the estimation of RBDEE, figure 3 describes an algorithm for final computation of RBDEE on the basis of IRBC of the proposed software. Since RBSSE is computed from RBDEFP and development language [5] as shown in equation 2, also the productivity of the software is calculated from equation 3. These measures are taken in into account to derive the $RBDEE_i$ in man-months for the proposed software that is expressed as:

$$RBDEE_i = RBSSE / productivity \quad Person\text{-}months \tag{5}$$

On substituting the value of productivity in initial effort equation, we get:

$$RBDEE_i = KLOC / (5.52+0.346 \text{ X } KLOC) \quad Person\ months \tag{6}$$

[Procedure for computation of RBDEE]

1. If (SDLC = NULL)then return
2. SET IRBC := ((PC * PCA) +DCI+IFC+SFC)*SDLC
3. Repeat step 4 for i= 1,2,….,14)
4. TCF = 0.65 + 0.01 * F_i [End of step 3 loop]
5. SET RBDEFP := IRBC * TCF
6. SET SOL := Const_value
7. SET RBSSE := (RBDEFP * SOL)/1000
8. SET Productivity = 5.52 + 0.346 * KLOC
9. $RBDEE_i$ = RBSSE/5.52 + 0.346 * KLOC
10. Repeat step 11 for j= 1, 2 ,…,9)
11. SET ECF: = 1.4 + (-0.03 * ECF)[End of step 10 loop]
12. $RBDEE_f$ = $RBDEE_i$ * ECF

Figure: 3 Algorithm showing computation of RBDEE from IRBC

Further, to make the development effort estimation more precise and accurate, it is necessary to consider environmental complexity factors (ECF) as available in [7] that are derived for the relaxation and quantification of development team. The development team identifies the impact of each factor on the basis of its perception [7]. The calculated factors are then summed to produce the value of ECF that is expressed as:

$$ECF = 1.4 + (-0.03*Environmental\ factor) \tag{7}$$

Every individual factor of ECF has an impact and dependency on both RBSSE and productivity of the proposed software. Therefore, to compute final effort estimate the $RBDEE_i$ is multiplied with ECF, this is mathematically expressed as:

$$RBDEE_f = RBDEE_i \text{ X } ECF \quad Person\ months \tag{8}$$

5. Results and Validation

This section compares the proposed RBDEE measure with various other established prevalent practices for software development effort estimation proposed in the past. In order to analyze the validity of the proposed measure, and for comparison purpose fifteen SRS's of various problem statements have been considered along with the source code of all these problems. The fifteen problems considered for applying the development effort estimation measures are – bit stuffing, matrix addition, matrix multiplication, first come fist serve algorithm implementation, knapsack problem, LZW algorithm, round robin scheduling, cyclic redundancy check computation, semaphore implementation, prime number computation, bubble sort, palindrome, calculator, RSA algorithm and linear search. The comparison is strictly based on various categories of software development effort estimation like – use case based, algorithmic model based and code based. Figure 4 shows the comparison of RBDEE with various algorithmic model based effort estimation measures. It can be seen from the plot that the proposed RBDEE measure has a close relation with

other established measures and the values obtained are very well aligned with each other. Figure 5 shows comparison of RBDEFP with use case and function point measures and observes an approximate mean value with both requirement based measures. Figure 6 shows the comparison with code based measure, which computes the development effort on the basis of operators and operands, but IRBC and RBDEE still aligns with it. Finally, Figure 7 exhibits the dependency of software development effort on its own contributing parameters- RBDEFP, IRBC and RBDEE and finally, it also establishes that higher complexity requires higher development effort.

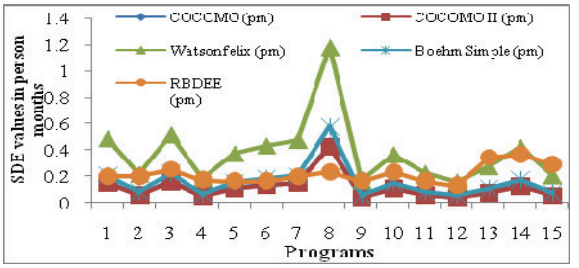

Figure 4: Plot between constant based measures v/s RBDEE

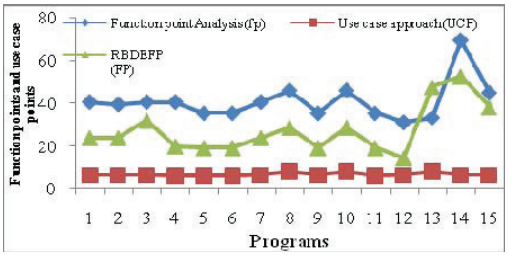

Figure 5: Plot between UCP, FPA and RBFP

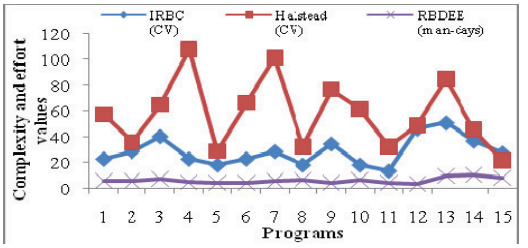

Figure 6: Plot between code based measures v/s proposed measures

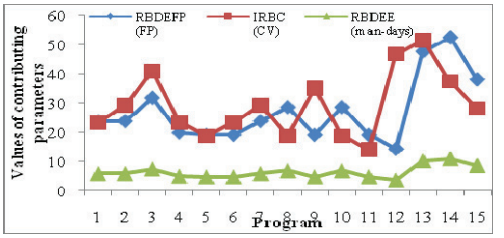

Figure 7: Plot between contributing RBDEE measures

6. Conclusion

The work presented in the paper computes the requirement based software development effort estimation from IRBC of the proposed software to make the effort estimation systematic, reliable and precise. Also, the robustness of the proposed RBDEE measure is established by comparing it with various other established measures for the estimation of software development effort. Although, there have been various proposals, but many of the important contributing factors such as input, output, interfaces, data storage, functional requirement decompositions, most importantly non-functional requirements and requirement based complexity of the proposed software have not been taken into consideration, which otherwise plays very significant and decisive role in estimation of software development effort. Results validates that the effort estimation values obtained from RBDEE aligns well with all the other established measures in a comprehensive fashion. The proposed measure can be drawn up at a very early software development phase that is based on SRS and not on any software analyst or any expert judgment. However, other established requirement based measures like use case depend on the prudence of the analyst and function point based measures requires expert judgment to predict the software development effort for the proposed software, that may induce unintended errors during analysis and modeling of the software requirements, that in turn lead to improper estimates.

References

1. B.W. Boehm, Software Engineering Economics, Englewood Cliffs, NJ: Prentice Hall, 1981
2. B.Boehm, E Horowitz, R.Madachy, B.Clark, C.Westland, R.Selby, Cost models for future software lifecycle process: COCOMO 2.0, IEEE computer society, pp 1-31, 1987
3. Yinhuan Zheng et.al. Estimation of the software project effort based on function point,IEEE-ICCSE-2009
4. Stephen Mac Donell, Comparative review of functional complexity assessment methods for effort estimation, Software Engineering Journal pp. 107-116, 1994.

5. Alan J Albrecht and John E Gaffiney, Software functions, source lines of code and development effort prediction: A software science validation, IEEE Transactions on Software Engineering, Vol SE-9, No.6, pp-639-648, November 1983
6. Matson, J.E.; Barrett, B.E.; Mellichamp, J.M.; Software Development Cost Estimation using function points, IEEE Transaction of Software Engineering, Vol. 20, Issue 4, pp-275 – 287, 1994
7. Clemmons. R.K., Project estimation with Use Case Points, The Journal of Defense Software Engineering. pp. 18-22, February, 2006
8. Magne Jorgensen, Dag IK Solberg, Impact of effort estimate on software project work, Elsevier, Info & Soft. Tech. pp. 939-948, 2006.
9. Bernhard Peschi et. al., Recommending effort estimate method for software project management, IEEE-MC-ACM, pp. 77-80, 2001.
10. Noureldin AZ Adem, Zarinah M Kasirun, Automating function point analysis based on the functional and the non-functional requirement text, pp. 664-669, IEEE Conf. 2010.
11. IEEE Computer Society, IEEE recommended practice for software requirement specification, IEEE std 830-1998
12. Geoffrey K. Gill, Chris F. Kemmerer, Cyclometric complexity density and software maintenance productivity, IEEE transactions on software engineering vol.17,no.12, pp. 1284-1288, December 1991.
13. Charles R Symons. Function point: Difficulties and Improvements, IEEE Transactions on Software Engineering, Vol.14, No.1, pp. 2-11, January 1988
14. IEEE standard for software productivity metrics, IEEE standard-1045-1992, pp. 1-38, 1993.
15. Sharma Ashish, Kushwaha D.S., NLP/ POS tagger based requirement analysis and its complexity analysis, ACM SigSoft , Vol.36, Issue 2, pp. 1-14, January 2011
16. Sharma Ashish, Kushwaha D.S., Complexity measure based on requirement engineering document and its validation, IEEE Conf. on Computer and Communication Technology , pp. 608-615, ICCCT 2010
17. M.H. Halstead, Elements of Software Science, New-York: Elsevier, 1977
18. DS Kushwaha, AK Misra, Software Test Effort Estimation, ACM Sig Soft, Software Engineering Notes, Vol. 33 No. pp. 1-6, 3, May 2008
19. The Standish group research for staggering bugs and effort, http://standishgroup.com