APPLYING REQUIREMENT BASED COMPLEXITY FOR THE ESTIMATION OF SOFTWARE DEVELOPMENT AND TESTING EFFORT

Ashish Sharma[1] and Dharmender Singh Kushwaha[2]

[1]Department of Computer Engineering & App., GLA University, Mathura, India
ashish.sharma@gla.ac.in
[2]Department of Computer Sc. & Engineering., MNNIT, Allahabad, India
dsk@mnnit.ac.in

ABSTRACT

The need of computing the software complexity in requirement analysis phase of software development life cycle (SDLC) would be an enormous benefit for estimating the required development and testing effort for yet to be developed software. Also, a relationship between source code and difficulty in developing a source code are also attempted in order to estimate the complexity of the proposed software for cost estimation, man power build up, code and developer's evaluation. Therefore, this paper presents a systematic and an integrated approach for the estimation of software development and testing effort on the basis of improved requirement based complexity (IRBC) of the proposed software. The IRBC measure serves as the basis for estimation of these software development activities to enable the developers and practitioners to predict the critical information about the software development intricacies and obtained from software requirement specification (SRS) of proposed software. Hence, this paper presents an integrated approach, for the prediction of software development and testing effort using IRBC. For validation purpose, the proposed measures are categorically compared with various established and prevalent practices proposed in the past. Finally, the results obtained, validates the claim, for the approaches discussed in this paper, for estimation of software development and testing effort, in the early phases of SDLC appears to be robust, comprehensive, early alarming and compares well with other measures proposed in the past.

KEYWORDS

IRBC, Requirement based development effort function point (RBDEFP), IEEE-830:1998 - SRS, RBDEE, Software Requirement Specification (SRS), SDLC, requirement based test effort estimation (RBTEE)

1. INTRODUCTION

Software development effort estimation is the process of calculating the most realistic measure of effort required to develop or maintain the software on the basis of inputs like software requirements, function points, size of proposed software, use case points etc. Further, the effort estimates are used as an input to project plans, iteration plans, budgets, investment analysis,

Natarajan Meghanathan, et al. (Eds): ITCS, SIP, JSE-2012, CS & IT 04, pp. 393–415, 2012.
 DOI : 10.5121/csit.2012.2137

pricing processes and bidding rounds. Software researchers and practitioners have been addressing the problems of effort estimation for software development projects since early 1960s. Most of the research has been focused on either construction of formal software effort estimation models or considers use case or function points for the estimation of size or effort, but does not consider software complexity for the computation of software development and testing effort. There are numerous researchers that have established a strong relationship between the complexity of the software and its impact on effort, schedule and maintenance. Therefore, there is a vital need to find a method through which the software development and testing effort can be estimated on the basis of improved requirement based complexity (IRBC) of the proposed software [15, 16] in requirement analysis phase of software development. The overall ideology of the work presented in this paper is to compute IRBC of the proposed software first and later empirically estimate the software development and testing effort for proposed software. Various prevalent methodologies that have been proposed in the past consider constants, size, scaling, use cases etc. for the estimation of software development effort and expert judgment, test specification, use case point, scenario, test execution complexity etc. for the computation of software testing effort. But, the effort prediction can be made more realistic and practically possible if we first compute the IRBC of the proposed software that has its basis on software requirements. Defining the user requirements is arguably one of the most difficult and challenging task for the development of complex systems [19]. Hence, the estimation of software development and testing effort on the basis of IRBC will be an early warning, systematic, less complex, faster and comprehensive one. The proposed measures are also validated, proved and compared with various established effort estimation practices proposed in past. For more rigorous and strict comparison the existing development effort and testing effort measures are classified in various categories like function point & use case point based effort estimation, code based effort estimation and algorithmic model & constant based effort estimation measures and the proposed measures are individually compared with these practices.

2. RELATED WORKS

Research shows that, development effort estimation is carried out on the basis of software requirements, function points, use case points, size and empirically determined constants. Various models, method, tools & techniques have been developed in the past for the estimation of the software development effort. This section presents a survey of some of the leading papers describing the work carried out so far in the area of software development and testing effort estimation.

2.1. Literature review for Software Development Effort

Barry Boehm [1] aims at estimating the development effort from COCOMO - I, for small to medium sized software projects. Three modes of software development are considered in this model: organic, semi detached and embedded. The value of development effort depends on size of software and empirically determined constants. Further, Barry Boehm et. al. [2] discusses the revised version of COCOMO-I as COCOMO-II. It uses some fixed values for one constant and the other constant depends on the scaling factors. The model also considers effort multipliers for the computation of effort in person month. For more accurate estimation, object points are also considered. Yinhuang Zheng et. al. [3] computes development effort for programming measurement and estimation. Estimation is carried out by considering 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, which captures environmental factors and various differences among the given projects. It also considers constant factor i.e. 5.5 to arrive at final effort. Albrecht and Gaffney [5] discusses about the unit of measurement to express the amount of software

functionality. Functional point analysis (FPA) is another popularly used method of measuring the size of software. The five functional units are ranked according to their complexity level and perspective standards in order to calculate the size of the proposed software. Matson et. al. [6] uses function point for the software cost estimation and also uses 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 which provides valuable insight for the effort and size requirement for design and implementation of the project. The method uses use case point, which is derived from the requirements of the software. Magne Jorgenson [8] attempts to provide six criteria for evaluation of the methods like - automation, comprehensive assessment, objectivity, specification, testing and validity for the software to be developed. It also provides comparative information of methods for the estimation development effort by considering the functional assessment and estimation option. Bernard et. al. [9] is a result of two case studies and two experiments to show the impact of effort estimates on software project. It works with two hypotheses: first pre-planning for effort estimation and secondly that too low effort estimates lead to less effort and more errors compared with more realistic effort estimates. This provides method to determine magnitude of relative error in development effort estimation. Noureldin AZ Adam 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 standard IEEE 830:1998. Geoffrey K Gill and Chris F. Kemerer [12] discusses about complexity density ratio that is a useful predictor of software maintenance productivity on the basis of cyclometric complexity. The paper also considers the measurement of software productivity in kilo non commented source lines of code (KNSLOC). Charles R Symon [13] discusses about function points analysis (FPA) and compare 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] discusses the improved requirement based complexity (IRBC) measure based on SRS of the proposed software and also proposes an object based semi-automated model for the tagging and categorization of 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 length, vocabulary, volume, potential volume, estimated program length, difficulty and finally the estimation of 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). Further the CICM is compared with cyclometric number. The measure is able to demonstrate that the cyclometric number and the test effort increase with increase in software complexity. Paper [19] discuss about importance of defining the user requirements and their impact on software development process.

2.2. Literature review for Software Testing Effort

During the last few decades, various models, methods and techniques have been developed to estimate the test effort for software to be developed. This section presents a survey of prevalent testing practicies which are categorized into code based, requirement based and complexity based methods for the estimation of software testing effort. Kuo Chang Tai [20] propose the exploration of testing complexity for several class of programs, based on testing path that is obtained on the basis of test data. Muthu Ramchandran [21] proposes a model for test process and investigates the possibility of deriving the test cases from system models and requirement analysis techniques. Johannes Ryser et. al. [22] describes validation and classification of software requirements based on heuristic and solution based strategies. Suresh Nageshwaran [23], discuss a use-case based approach for the estimation of test effort based on use case weight, use case points and complexity factors. Borris Veysburg et. al. [24] discusses an approach for the reduction of

requirement based test suites using Extended Finite State Machine (EFSM) dependence analysis. The technique supports test case generation from EFSM system models. Ian Holden and Dalton [25] uses Cumulative Test Analysis (CTA) for test selection and produces an objective measure upon identifying and assigning impact of risk for test effectiveness. Antonio Bertilino [26] discusses testing roadmap for the achievements, challenges in software testing and discusses four dreams as efficacy maximized test engineering, 100% automatic testing, test based modelling and universal test theory for the testing of any software. Aranha and Borba P [27] uses a tool to convert test specification into natural language for the estimation of the test effort and also finds test size and test execution complexity measure. Ajitha Rajan et. al. [28] proposed an approach to automate the generation of requirements based tests for the model validation, in order to formalize the requirement using linear temporal logic (LTL) properties for the test case generation. Aranha et.al. [29] discusses test execution and a test automation effort estimation model for the test case selection on the basis a controlled natural language and uses a manual coverage and automated test case generation technique for effort estimation. Harry Sneed [30] uses a test strategy & automatically performs requirement analysis by identifying keywords in requirement analysis phase and generates the test cases. Uuistallo et. al. [31] provides a set of practices that can be applied to link the requirements with testing based on interdependencies and linking the people with requirement documentation. Zhu Xiachun et. al. [32] presents an experience based approach for the estimation of software test suite size. Zho and Xiachun [33] presents an empirical study on early test execution effort estimation based on test case number prediction from use case and estimation the test effort using test execution complexity. Tibor Ripasi [34] models the development process using V & W models for the automated test case generation and test execution. Deniel et. al. [35] considers effort estimation model based on data analysis, hypothesis formulation, evaluation, accumulated efficiency and finally models the test effort. Erica R. et. al. [36] describes a method for test effort based on the information contained in use case. It also considers the various parameters like actor ranking and technical environment factor in order to finally arrive at test effort estimation. Veenendal E Dekkers [37] provides a method called test point analysis (TPA) that uses function points for the estimation of final result.

3. COMPUTATION OF REQUIREMENT BASED COMPLEXITY FROM SRS

It has been established that the complexity of the software has a direct bearing on the required effort of that software [18]. For systematic, planned and accurate estimation of software Since software complexity plays an extremely important role in identifying the degree of difficulty associated with the software, and, has an extremely high pay off for investment. Hence, this section calculates IRBC measure, which is derived on the basis of software requirements written as per the recommendations of IEEE: 830:1998 for SRS document. In order to make precise and perfect estimation of software complexity for the proposed software, figure 1 shows a procedure for the computation of IRBC for yet to be developed software. IRBC [15, 16] is obtained on combining all the contributing complexities on the basis of their significance and relative contributions towards computation of overall complexity for any physical system or yet to be developed software.

The Next section discusses about the application of IRBC for the estimation of software development effort .

4. 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. For the estimation of SDE, traditional methods are either dependent on lines of code that is available only at later stage in the software development life cycle (SDLC) or

dependent on function points or use case points that is based on prudence of analyst or requires expert judgment in order to estimate development effort for the proposed software. However, the proposed measure predicts the software development effort on the basis of complexity analysis of yet to be developed software.

A framework for the estimation of proposed requirement based software development effort (RBDEE) is shown in figure 2, wherein the entire estimation is carried out on the basis of IRBC [15, 16] that has been derived from elicited customer's requirements and documented as per IEEE 830:1998 standard [11] for the generation of SRS for the proposed software.

Steps to compute IRBC

1. [Computation of Input Output Complexity]
   $IOC = IC + OC + SC$
2. [Computation of Functional Requirement based Complexity (FR)]
   $$FR = \sum_{i=1}^{n} (Functionality)i \sum_{j=1}^{3} (SPF)ij$$
3. [Computation of Non-Functional Requirement based Complexity]
   $$NFR = \begin{cases} \sum_{i=1}^{6} (Attribute)_i \ \sum_{j=1}^{m} X\ (Subattribute)_{ij} \\ 1 \ otherwise\ for\ inbuilt\ quality\ attributes \end{cases}$$
4. [Computation of Requirement Complexity (RC)]
   RC = FR + NFR
5. [Computation of Product Complexity (PC)]
   PC = IOC * RC
6. [Computation of Personal Complexity Attributes (PCA)]
   $$PCA = \prod_{i=j=1}^{5} (AttributeValue)ij$$
7. [Computation of Design Constraints Imposed (DCI)]
   $$DCI = \sum_{i=0}^{n} (Constra\,\text{int}\_type)i \sum_{j=0}^{6} (Number)ij$$
8. [Computation of Interface Complexity (IFC)]
   $$IFC = \sum_{i=0}^{n} (ExternalInterfaces)i \sum_{j=0}^{m} (Number)ij$$
9. [Computation of Software deployment location Complexity]
   $$SDLC = \sum_{i=1}^{n} (LC)i \sum_{j=0}^{m} (UCC)ij$$
10. [Computation of Software Feature Complexity (SFC)]
    $$SFC = \sum_{i=1}^{n} (Feature)i$$
11. [Computation of IRBC]
    IRBC = ((PC X PCA) + DCI + IFC + SFC) X SDLC

Figure 1: Procedure for the computation of IRBC

The method of computing SDE also employs fourteen technical complexity factors (TCF) [13] as general application characteristics for the proposed software on the basis of degree of influence to quantify non functional requirements also. The obtained IRBC and TCF serves as basis for the estimation of requirement based development effort function points (RBDEFP). Further the obtained RBDEFP serves as the basis to derive size, project types, productivity and finally requirement based software development effort (RBDEE) for the proposed software.

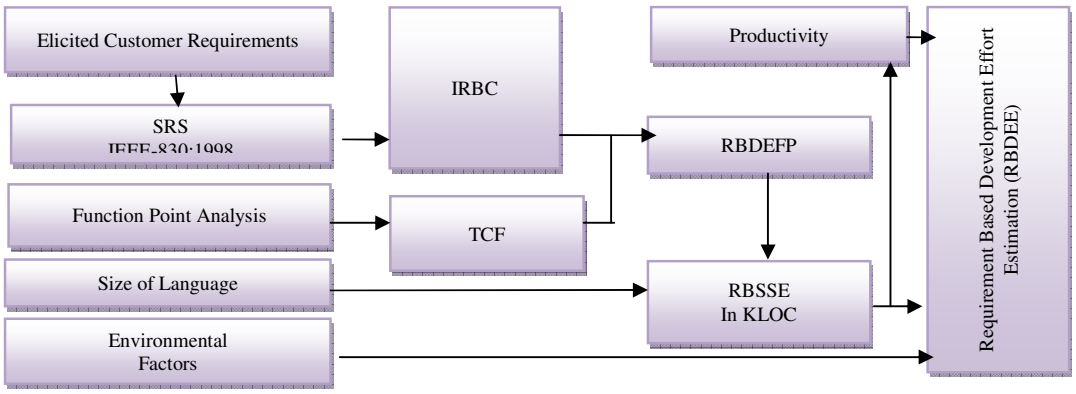


Figure 2: Process flow showing the computation of proposed software development effort

4.1 COMPUTATION PROCEDURE

4.1.1 Requirement based Development Effort Function points (RBDEFP)

IRBC is obtained on the basis of software requirements written as per the recommendations of IEEE-830:1998 document. It encompasses an exhaustive set of parameters for the evaluation of complexity, of the proposed software, immediately after freezing the software requirements in requirement analysis phase of SDLC. However, attributes needed for the computation of FPA are subset of IRBC parameters. Further, in order to obtain function points and make the software development effort estimation precise and practically possible, technical complexity factors (TCF) [13] are also considered as represented in table 1, TCF consist of fourteen components that provide general application characteristics for the software projects. These characteristics have an impact on software productivity relating to various technical issues for 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).

Table 1: Technical Complexity factors

Characteristics	**DI**	**Characteristics**	**DI**
Data Communication	--	On line update	--
Distributed Function	--	Complex processing	--
Performance	--	Re-usability	--
Heavily used	--	Installation ease	--
Transaction rule	--	Operational ease	--
On line data entry	--	Multiple sites	--
End user efficiency	--	Facilitate change	--

Table 2: DI Values

Influences	**Degree**
Not present influence	0
Insignificant influence	1
Moderate influence	2
Average influence	3
Significant influence	4
Strong influence	5

Table 2 illustrates the various DI values. However, 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:

$TCF = 0.65 + 0.01 \times \sum Fi$

where Fi shows required and applicable factors for the proposed software.

Finally, the obtained IRBC and TCF serve as the 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].

Table 3 shows the dependency and relationship between the attributes of IRBC and TCF. It is observed that all the attributes are functionally dependent on each other. Hence, RBDEFP is described as a product of IRBC and TCF and is expressed as:

RBDEFP = IRBC X TCF function points

The obtained RBDEFP serves as the basis for computing out the optimal number of function point that is further required to estimate the size of the proposed software. The following section discusses about the estimation of software size from RBDEFP.

Table 3: Dependency and Relationship between TCF and IRBC

	General Application Characteristics	IRBC					
		PC	PCA	DCI	IFC	SFC	SDLC
Technical Complexity Factors (TCF)	Data Communication	√	√	√	√	√	√
	Distributed Function	√	√	√	√	√	√
	Performance	√	√	√	√	√	√
	Heavily used	√	√	√	√	√	√
	Transaction rule	√	√	√	√	√	√
	On line data entry	√	√	√	√	√	√
	End user efficiency	√	√	√	√	√	√
	On line update	√	√	√	√	√	√
	Complex processing	√	√	√	√	√	√
	Re-usability	√	√	√	√	√	√
	Installation ease	√	√	√	√	√	√
	Operational ease	√	√	√	√	√	√
	Multiple sites	√	√	√	√	√	√
	Facilitate change	√	√	√	√	√	√

4.1.2 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. The amount of functionality to be performed by the proposed software can be estimated on the basis of functional requirements and on analysis of sub-processes associated to the functionality. 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]. Therefore, for precise effort estimation, it is absolutely imperative that the size of the language should be considered on per function point basis [12]. Hence, RBSSE in terms of KLOC is mathematically expressed as:

RBSSE = (RBDEFP X Size of language)/ 1000 KLOC

Though, function point is proportionate to the size of language [5], higher function points will result in higher source lines of code (SLOC) irrespective of the level of language, Therefore, RBDEFP is multiplied by the size of language [5].

In order to further 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 complexity is expressed as:

Productivity = 5.52+0.346 X KLOC

4.1.3 Requirement Based Software Development Effort Estimation (RBDEE)

As discussed in earlier section, in order to compute software development effort for the proposed software, it is necessary to consider various contributing factors related to SDE estimation like RBSSE, productivity, and environmental complexity factors (ECF). Firstly, the initial requirement based software development effort (RBDEEi) is calculated on the basis of RBSSE and productivity of the proposed software and later the final requirement based software development effort (RBDEEf), is calculated on the basis of RBDEEi and applicable environmental complexity factors. In order to clearly understand the entire computation procedure for the estimation of RBDEE, figure 3 describes an algorithm for computation of proposed RBDEE measure on the basis of IRBC of the proposed software.

[Procedure for computation of RBDEE]

1. [Calculate Software deployment sites]
   If (SDLC = NULL)then return
2. [Compute IRBC of proposed software]
   SET IRBC := ((PC * PCA) +DCI+IFC+SFC)*SDLC
3. [Computation of Technical Complexity Factors]
   Repeat step 4 for i= 1,2,….,14)
4. [Computation of Technical Complexity Factors]
   TCF = 0.65 + 0.01 * F_i
   [End of step 3 loop]
5. [Compute Requirement based development effort function points]
   SET RBDEFP := IRBC * TCF
6. [Identification of language of development]
   SET SOL := Const_value
7. [Size Estimation in KLOC]
   SET RBSSE := (RBDEFP * SOL)/1000
8. [Computation of Software Productivity]
   SET Productivity = 5.52 + 0.346 * KLOC
9. [Initial Effort Estimation]
   $RBDEE_i$ = RBSSE/5.52 + 0.346 * KLOC
10. Repeat step 11 for j= 1, 2 ,….,9)
    SET ECF: = 1.4 + (-0.03 * EF_j)
    [End of step 10 loop]
11. $RBDEE_f$ = $RBDEE_i$ * ECF
    Return

Figure: 3 Algorithm for computation of RBDEE from IRBC

Since RBSSE is computed from RBDEFP and software development language [5]. However, the productivity of the software is calculated on the basis of [12]. These measures are taken in into account in order to derive the initial requirement based software development effort estimate (RBDEEi) equation in man-months for the proposed software, this is expressed as:

RBDEEi = RBSSE / productivity Person-months

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

RBDEEi = KLOC / (5.52+0.346 X KLOC) Person months

Software developer may be sound with the syntax of the language used for software development but several other related skills as shown in table 4 also have an impact on the productivity of yet to be developed software. Since RBSSE is computed from RBDEFP and software development language [5]. However, the productivity of the software is calculated on the basis of [12]. These measures are taken in into account in order to derive the initial requirement based software development effort estimate (RBDEEi) equation in man-months for the proposed software, this is expressed as:

RBDEEi = RBSSE / productivity Person-months

On substituting the value of productivity in initial effort equation, we get:
RBDEEi = KLOC / (5.52+0.346 X KLOC) Person months
Software developer may be sound with the syntax of the language used for software development but several other related skills as shown in table 4 also have an impact on the productivity of yet to be developed software.

Table 4: Environmental Complexity Factors

Factor	Description	Weight
E1	Familiarity with UML	1.5
E2	Part time workers	-1
E3	Analyst capability	0.5
E4	Application experience	0.5
E5	Object oriented	1
E6	Motivation	1
E7	Difficult programming	-1
E8	Stable requirements	2

Table 5: Dependency and interconnection between Ei and ECF

ECF	RBDEEi	
	RBSSE	Productivity
E1	↓	↑
E2	↑	↓
E3	↓	↑
E4	↓	↑
E5	↓	↑
E6	↓	↑
E7	↑	↓
E8	↓	↑

Thus, to make the development effort estimation more precise and accurate, it is necessary to consider environmental complexity factors (ECF) [7] that are derived on the basis of software requirements, for the relaxation and quantification of development team(s). Further, the development team identifies the impact of each factor on the basis of its perception [7]. Table 4 represents eight ECF's along with their respective weights. A value of 1 indicates that the factor has lower impact on the project; value of 3 has an average impact and 5 means it has a strong positive impact. A value of zero has no impact. The negative sign associated with couple of factors shows that, if they exist, then it is detrimental for the software development. Hence, each

factor's weight is multiplied by its perceived impact. The calculated factors are then summed to produce the value of ECF that is expressed as:

ECF = 1.4 + (-0.03 X Environmental factor)

Table 5 shows the dependency and interconnection between various attributes that contribute towards estimation of Ei with various factors of ECF. Where "↑" shows increment and "↓" shows decrement.

Since every individual factor of ECF has an impact and dependency on both RBSSE and productivity of the proposed software. Therefore, in order to obtain final requirement based development effort (RBDEEf), the RBDEEi is multiplied by ECF, this is mathematically expressed as:

RBDEEf = RBDEEi X ECF Person months

Early estimation of software development effort using software requirements shall save tremendous amount of time, cost and man power for yet to be developed software and shall provide a great help for precise planning and execution of software process as well as software projects.

4.2 Software Documentation & Comparison

4.2.1 Documentation using IEEE-830:1998

This section carries out a case study of FCFS Scheduling algorithm in order to illustrate the proposed metric and its comparison with other existing measures.

Example SRS: FCFS Scheduling
Introduction
The CPU is one of the primary computer resources. First come first serve (FCFS) scheduling algorithm is based on the concept of assigning the CPU to the process that requests the CPU first.

2. Purpose

The main purpose of FCFS scheduling algorithm is to increase the CPU utilization & throughput and reduce waiting time and response time. With this algorithm one can achieve fairness in allocating the processes to CPU based on the order of their arrival.

3.Scope

The major scope of FCFS scheduling algorithm lies in batch systems, where the waiting time can be large if short request waits behind the longer ones. Therefore, FCFS is used in the case where the burst time of the processes are comparatively less and in ascending order. It is not suitable for time sharing systems but is used in multilevel feedback queues.

4. Definitions

FCFS is a scheme in which the process that requests the CPU first is assigned the CPU first. Throughput can be defined as the number of processes that are to be completed per unit time. Turnaround time is the interval between the submission time of the processes to the completion

time. Waiting time can be defined as the sum of periods spent in waiting in the ready queue. Response time is the amount of time that describes when the processor starts responding.

5. References

Galvin, Abraham Silberschatz, Introduction to Operating System Concepts, 7th Edition, Wiley publication.

6. Overview

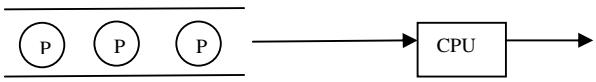
This algorithm executes the requests on the basis of their arrival. The average waiting time under this policy is quite long; hence, the FCFS policy is non-preemptive.

7. Overall Problem Description:

Process that requests the CPU first will be allocated the CPU first. When a process enters the ready queue, its PCB is linked onto the tail of the queue. When the CPU is free it is allocated to the process at the head of the queue. The running process is then removed from the queue.

8. Product Perspective:

The perspective of the case study for FCFS scheduling can be described with a block diagram shown below, where the number of process in ready queue will be assigned on first come and first serve basis to the processor.



READY QUEUE

9. Product Functions:

9.1 Inputs: Process arrival time, number of processes, burst time, waiting time process, turnaround time.

9.2 Outputs: Display of waiting Time, Display of turnaround Time.

10. User Characteristics:

User type is end user only for providing the input and visualizing the result on output screen. The computation to be performed is on single machine without any client server environment.

11. Constraints

FCFS is non-preemptive in nature, and FCFS is particularly troublesome for time-sharing system. Also, no process should be allowed to keep the CPU for an extended period.

4.2.2 Illustration and comparison of RBDEE with other SDE measures

The commonly used measures by the practitioners and software developers such as COCOMO-I, COCOMO-II, Watson Felix Model, Bailey Basili Model, Boehm model, Function point analysis, Matson and Barnett method and Use Case based approaches are used to compute the

development effort estimate in person-months and these results are later compared by the proposed RBDEE measure.

Program: First Come First Serve Algorithm (FCFS) Scheduling Algorithm.

Method 1: COCOMO

Size=25 LOC, Size=0.025 KLOC
Effort=a*(size)b =2.4(.025)1.05= 0.0498 PM

Method 2: COCOMO II

Size=0.025 KLOC, Effort =A*(Size)B * (product of effort multipliers) =2.5(0.025)1.09 *(1) = .0448 PM

Method 3: Watson Felix Model

Effort= 5.2*(KLOC)0.91 , 5.2*(.025)0.91 = 0.1811PM

Method 4: Bailey Basili Model

Effort=5.5+0.73*(KLOC)1.16 , 5.5+0.73*(.025)1.16 = 5.5101PM

Method 5: Boehm

Effort=3.2*(KLOC)1.05 , 3.2*(.025)1.05 = 0.0665PM

Method 6: Function point analysis

Number of input=2; Number of output=2; Number of files=2
UFP=2*4+2*5+2*10=38
FP=UFP*CAF=38*1.07=40.66

Method 7: Matson, Barnett and Mellichamp method

Effort = 585.7+15.12(FP)
585.7+15.12(40.66) =1200.479

Method 8: Use Case Point Approach

Unadjusted Actor Weight:

Actor	No. of Use Cases	Factor	UAW
User	1	2	2

Total=2 Unadjusted use case Weight (UUCW):

Use Case	Type	Factor
Input	Simple	5
User Creation	Simple	5
Resources	Simple	5

Total=5+5+5=15; Unadjusted Use Case Point=2+15=17

Technical Factor Description:

Facto	Description	Per.	Cal.
T1	Distributed system	0	0
T2	Performance	3	3
T3	End user efficiency	0	0
T4	Complex internal processing	0	0
T5	Reusability	0	0
T6	Easy to install	0	0
T7	Easy to use	3	1.5
T8	Portability	3	6
T9	Easy to change	3	3

T10	Concurrency	0	0
T11	Special security features	0	0
T12	Provides direct access for third	3	3
T13	Special user training facilities	0	0

Total=16.5; Environmental total factor

Factor	Description	Wt.	Per.	Calc.
E1	Familiarity with UML	1.5	5	7.5
E2	Part time workers	-1	0	0
E3	Analyst capability	0.5	5	2.5
E4	Application experience	0.5	5	2.5
E5	Object oriented experience	1	5	5
E6	Motivation	1	5	5
E7	Difficult programming	-1	0	0
E8	Stable requirements	2	5	10
Environmental total factor				32.5

UCP Calculation:

UCP=UUCP*[0.65+.01*16.5]*[1.4+(0.03*32.5)]=17*[0.65+.01*16.5]*0.425 =5.888 UCP

Method 9: Proposed RBDEE Measure

Calculation of Input Output Complexity:

a. Calculation of Input Complexity:
   i. Number of Input : 2
   ii. Type of Input : Integer
   iii. Source of Input : Keyboard = 2*1*1 =2

b. Calculation of Output Complexity:
   i. Number of Output : 2
   ii. Type of Output : Integer
   iii. Source of Output : Screen = 2*1*1 =2

c. Calculation of Storage Complexity: 1

IOC = Input Complexity + Output Complexity + Data Storage Complexity = 5

2. Calculation of Functional Requirements:
   i. Functionality to be performed: FCFS
   ii. Decomposed Sub Processes: Input of Execution time, Computation, Display

FR = 1*3=3

3. Calculation of Non-Functional Requirements: based on ISO-9126 model

i. No. of Attributes to be considered: Functionality, Usability

ii. No. of Sub-attributes to be considered:Accuracy, Operability

NFR = 1*1+1*1 =2

4. Requirement Complexity = FR +NFR = 3+2=5

5. Product Complexity = IOC X RC = 5*5 =25

6. Personal Complexity Attributes: = 1.17

7. Design Constraints Imposed: DCI = 0

8. Interface Complexity: IFC = 0

9. User Class Complexity: UCC = 1; No. of User Class considered: Casual End User

10. System Feature Complexity: SFC = 0

11. SDLC= 1*1 =1

12. IRBC = 29.25
13. RBDEFP=IRBC*CAF=29.25*0.68=19.89
14. RBSSE= (19.89*128)/1000=2.545 KLOC
15. Productivity=(5.52+0.346*KLOC)
16. RBDEEf = (KLOC/(5.52+0.346*KLOC))*ECF
    (2.545/(5.52+0.346*2.545))*0.425=0.1690 Person Month

Having seen the result of various existing proposals and the proposed RBDEE metric for the computation of software development effort estimation, we evaluate, these for a variety of programs in order to ascertain and establish the proposed metric as illustrated in table 7.

4.3 RESULTS AND VALIDATION

This section categorically compares the proposed RBDEE measure with various established measures for software development effort estimation proposed in the past. In order to analyze the validity of the proposed measure, fifteen SRS's of various problem statements have been considered and for evaluation and comparison sake the source code of all fifteen problems are also developed. Table 7 illustrates a tabular comparison between RBDEE and other established measures. 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 between proposed development effort measure and established algorithmic model based effort estimation measures. It is seen from the plot that the proposed measure has a close relation with other established measures and the values obtained from proposed as well as other measures are very well aligned. Higher value for Watson Felix [3] measure is due to the multiplication of constant 5.2 with the size of the software to estimate final development effort. Also very high value of program #8 is due to higher SLOC.

Figure 5 shows comparison of RBDEFP with use case points and function point measures. All the measures takes software requirements of the proposed software into consideration, however, use case based measures considers actor weight, unadjusted use case points (UUCP) and various other factors for the estimation of software development effort but does not takes the non functional requirements of the proposed software into account. Function point measure merely considers only five different attributes from software requirements of the proposed software.
It provides a basic estimate of function point count and requires intervention of human experts for the calculation of function points. However the proposed measure uses complexity of proposed software in order to calculate software development effort that does not use any arbitrary value.

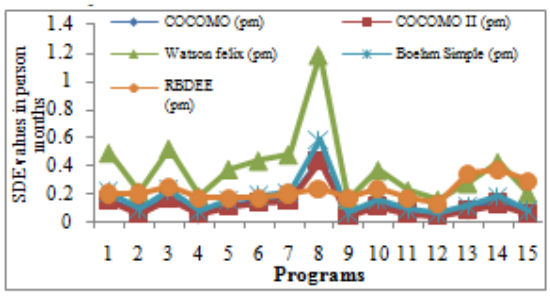


Figure 4: Plot between constant based software development effort measures v/s RBDEE

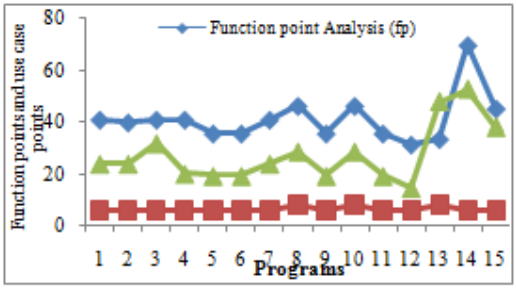


Figure 5: Plot between UCP, FPA and RBFP

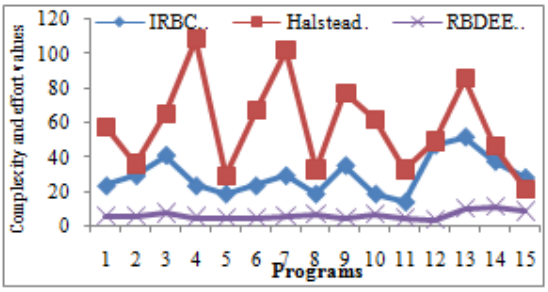


Figure 6: Plot between codes based development effort estimation measures v/s proposed measures

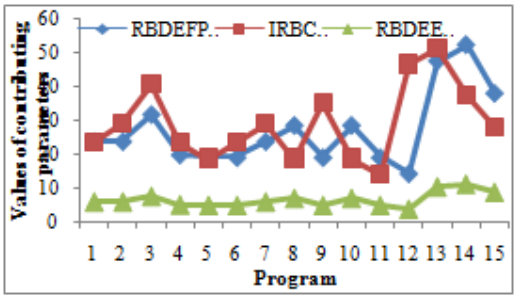


Figure 7: Plot between IRBC, requirement based function points and RBDEE

Figure 6 shows the comparison between code based measure i.e. Halstead [17], IRBC and proposed RBDEE measure. Halstead measure computes the development effort on the basis of code of the proposed software. It later calculates number of operators, operands, occurrences of operators and operands, program length, vocabulary, language level, difficulty etc. to compute the final effort in terms of mental discriminations and on the basis of a constant. The measure is computation intensive and the amount of re-work also gets increased. Higher values of results obtained for program no. 3, 7, 9 and 13 are due to the number of source lines of code, but IRBC still aligns with it.

The metrics based on use case includes the parameters like actor weight, use case weight; constant conversion factors etc for the estimation of development effort, but the important contributing factors such as input, output, interfaces, storage, functional requirement and most importantly non functional requirements are not taken into consideration in the existing and established development effort measures. The code based methods consider the size of the software in terms of lines of code and calculates operators, operand, difficulty, language level to finally arrive at the development effort. Further, the algorithmic model based measures consider the size of software in SLOC and later estimate the development effort on the basis of empirically determined constants that varies with the model available. However, the effort estimation obtained from the proposed metric is more realistic because it is systematically derived from IRBC that in turn is obtained from confirmed and documented software requirements on the basis of IEEE: 830:1998 SRS documents. This methodology is not used by any other established measures for the estimation of development effort. The proposed measure when compared with three categories of effort estimation, illustrates that, the proposed measure is a comprehensive one, results are well aligned and comparable with other established measure.

Finally, Figure 7 shows the dependency of software development effort on the IRBC and RBDEFP of the proposed software. It can be deduced that the development effort estimation is dependent on the complexity of the software. Though the measuring units of parameters are different, still the dependency is observed. The unit for IRBC is complexity value; RBDEFP is measured in function points, however development effort is measured in person months. The purpose is to show the actual relationship that exists between all three parameters, that can be described as, higher complexity requires higher development effort.

5. ESTIMATION OF SOFTWARE TESTING EFFORT

This section discusses about the application of IRBC for the estimation of software testing effort for the proposed software in order to evade systematic testing. Since, every type of testing

technique demands adequate test case generation, modelling and documentation. Though many software testing measures have been proposed in the past research, but still it is far from being matured. The following paragraphs discuss about derivation of proposed test effort estimation measure from obtained IRBC.

5.1 Computation Procedure

5.1.1 Requirement Based Test Function Points (RBTFP)

IRBC comprises of all the attributes that are sufficient to compute the function point analysis (FPA) [13] for any software. Function point is a unit of measurement to express the amount of functionality that software provides to a user. The function point measure includes five parameters i.e. external input, external output, interfaces, file and enquiry. These parameters are the basis for the estimation of required function points and the size of the proposed software. In addition to the five parameters used by FPA, there are also certain other parameters, that are extracted from software requirements and IRBC make use of these and an exhaustive set of extracted parameters for its computation, which in turn will help in computing the requirement based test function point (RBTFP) measure. In order to fine grain the estimate of RBTFP, it is necessary to consider weighted technical and environmental factors (TEF) as available [23, 36] pertaining to the testing activity. The need and applicability of these factors are determined on the basis of degree of influence (DI), ranges from zero (harmless) to four (essential). Hence, TEF can be computed by summing the score of nine different factors as:

$$TEF = 0.65 + 0.01 \text{ X } \sum Fi$$

Since, IRBC for its computation, has a strong bearing on two basic parameters i.e. functionality to be performed and input(s) for the system, and these parameters are sufficient to decide and generate the test case in both black box and white box scenarios. Also, we have nine TEF, specifically defined for the purpose of software testing and differ from TCF. Hence, requirement based test function point (RBTFP) can be described as a product of IRBC of the proposed software and weighted sum of TEF. Table 6 shows the dependency and relationship between the various attributes of TEF and IRBC. Since all the attributes are functionally dependent on each other, hence, IRBC is multiplied by TEF that is expressed as:

$$RBTFP = IRBC \text{ X } TEF$$

Table 6. Dependency between TEF and IRBC

		Technical and Environmental Factors (TEF)								
		Test Tool	Doc. Input	Dev. Env.	Test Env.	Test Reuse	Dist. System	Performance	Security	Interface
IRBC	PC	√	√	√	√	√	√	√	√	√
	PCA	√	√	√	√	√	√	√	√	√
	DCI	√	√	√	√	√	√	√	√	√
	IFC	√	√	√	√	√	√	√	√	√
	SFC	√	√	√	√	√	√	√	√	√
	SDLC	√	√	√	√	√	√	√	√	√

Also, RBTFP serves as the basis for determining the optimal number of test cases that is essential to evade exhaustive testing for the proposed software.

5.1.2 Number of Requirement Based Test Cases (NRBTC)

Test case is set of conditions under which tester determines the correctness of the proposed software functionality on the basis of requirements. Estimation of number of requirement based

test case (NRBTC) is a function of requirement based test function point (RBTFP), because numbers of function point dictate the number of test cases to be designed [33]. Like function points, acceptance test cases should be independent of technology and implementation techniques. Hence, this is expressed as:

$$NRBTC = (RBTFP)^{1.2}$$

Numbers of test cases are closely related to the amount of required testing effort. Hence, NRBTC plays a very significant role in the estimation of required test effort in man hours for the proposed software.

5.1.3 Requirement Based Test Team Productivity (RBTTP)

Productivity is defined as accomplishment of objective in a given unit of time. Hence, test team productivity depends on the number of staff and personnel (talent) available to test the software. In order to estimate test team productivity, we consider rank and proficiency of tester. Therefore, a model [32] proposes estimation of tester rank on the basis of two dimensions i.e. experience in testing and knowledge of target application as represented in figure 8.

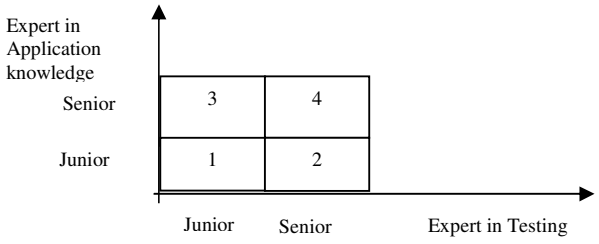


Figure 8. Tester Rank Model

The tester rank helps in understanding tester behavior for test execution because higher the rank of test team, lower the need of number of tester. Therefore, in order to derive the requirement based test team productivity (RBTPP), the number of testers and their relative rank is considered that is expressed as:

$$RBTTP = \sum_{i=1}^{n} T_i \; X \sum_{j=1}^{4} Rij$$

Where T shows testers and R is the respective rank of the tester from tester rank model. Having obtained the value of number of test cases (NRBTC) and test team productivity (RBTTP), in the following section we compute the requirement based test effort estimation in man-hours for the proposed software.

5.1.4 Requirement Based Test Effort Estimation (RBTEE)

In order to compute software testing effort for the proposed software, it is necessary to have knowledge of two significant parameters, first, the prior estimation of number of test case requirement and second, productivity of the test team. Hence in this direction we have already derived the contributing measures i.e. NRBTC, for the computation of number of test case and RBTTP, for the estimation of test team productivity for the proposed software. It is necessary to consider the combination of these measures for the estimation of proposed RBTEE because of the following reasons:

There is a strong relationship between number of test cases and the number of test team productivity because in order to carry out successful testing, the knowledge in terms of expertise and experience of tester in respective domain plays a very significant role.

Also, to deliver tight interaction, a software test suite must have a common shared notion of a team project. Further, an actual implementation of successful test is managed with persistent data, in order to make automation, relationship and communication possible.

Lastly, the complicacy of the test case increases when there is a hierarchy of requirements. A high level business requirement is broken down into several lower level functional and technical requirements. Therefore, the test plan must include the details about the type and level of requirements for which the test cases are to be generated and depends on the proficiency of test team. Hence, these measures are multiplied in order to obtain final requirement based test effort estimate (RBTEE) in man-hours for the proposed software that is expressed as:

$$RBTEE = NRBTC \text{ X } RBTTP \quad \text{man-hours}$$

Early estimation of software testing effort using IRBC will save tremendous amount of time, cost and man power for the proposed software. The next section carries out a case study in order to elaborate the proposed approach and its comparison with various other prevalent approaches given in the past.

5.2 Results and Validation

This section categorically compares the proposed RBTEE measure with various other established test effort estimation measures proposed in past. To analyze the validity of the proposed measure, fifteen SRS's of various problem statements have been developed and compared with various prevalent testing practices as shown in table 8. The comparison strictly considers various categories of test effort estimation like use case based, complexity value based and code & execution points based.

Figure 9 shows the comparison of proposed RBTEE measure with other requirement based approaches like use case point and scenario.

It is seen from the plot, that, the value of use case point based approach [23] is on higher side because of multiplication of constant 20 with adjusted use case point. However, lower values of scenario based measures [36] are due to consideration of lower conversion factor i.e. constant 3. The co-relation observed with use case based approaches and the proposed measure illustrates that: Use case point based approach is close to five times of the proposed measure, and, scenario based approach is close to one-third of the proposed measure.

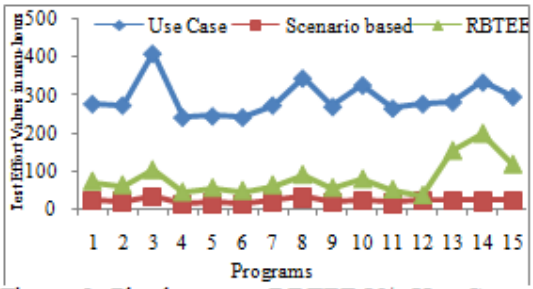


Figure 9. Plot between RBTEE V/s Use Case based measures

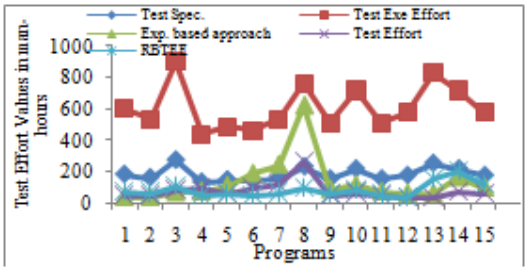


Figure 10. Plot between RBTEE V/s Test Execution points based measures

It is seen from the plot, that, the value of use case point based approach [23] is on higher side because of multiplication of constant 20 with adjusted use case point. However, lower values of scenario based measures [36] are due to consideration of lower conversion factor i.e. constant 3. The co-relation observed with use case based approaches and the proposed measure illustrates that:

- Use case point based approach is close to five times of the proposed measure, and,
- Scenario based approach is close to one-third of the proposed measure.

Further, Program # 3 and 14 shows higher values of test effort because of increased number of use case points, extended value of complexity factors with corresponding decompositions and varying value of normal and exceptional scenarios.

Figure 10 shows the values obtained from test execution point and test specification based approaches that are purely code dependent. The code & execution point based approaches carry higher values of test effort in man-hours than the proposed measure because of the use of execution points, that in turn depends on number of variables used in the program. Higher values for 3, 10 and 13 are due to higher execution points, screen items that in turn increase the execution complexity and required test effort. However, rest of the programs follows normal trend. The proposed measure is drawn from IRBC that uses an exhaustive set of attributes; hence the values obtained from proposed measure are lower. Though, other categories of test effort estimation observes similar trend with the proposed RBTEE measure in both cumulative and individual fashion.

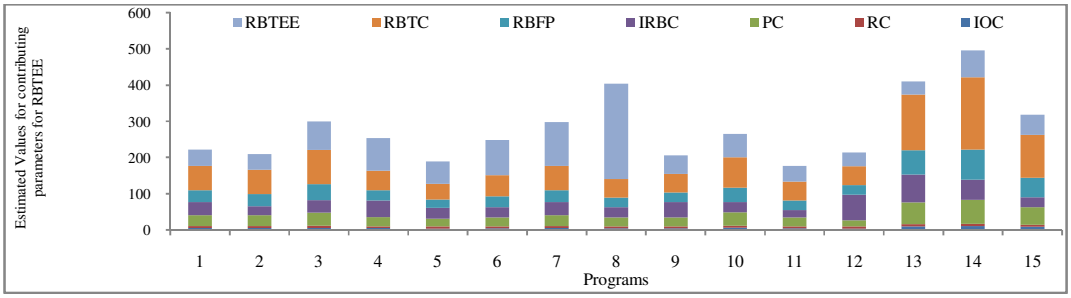


Figure 11. Comparison between IRBC and RBTEE

Finally, Figure 11 shows the comparison between various contributing parameters of RBTEE to show the dependency and relative contribution among self. From the plot, it can be deduced that, test effort estimation is dependent on requirement based complexity as, higher complexity requires higher test effort.

6. CONCLUSIONS

The work presented in this paper attempts to estimate the most sensitive and critical software development activities from the IRBC of proposed software. Also, to accomplish the objectives, an attempt has been made to address the following issues:

- Early estimation of IRBC on the basis of SRS of the proposed software,
- Further, estimation of software development effort using IRBC of proposed software, and,
- Finally, a metric for early estimation of software testing effort from IRBC.

At the onset, this paper uses an improved requirement based complexity that acts as basis for the estimation of software development and testing effort. Since, IRBC is capable of computing the

complexity of proposed software at a very early stage of the software development; hence it outperforms other approaches for the prediction of software development and test effort. Later, the paper presents requirement based software development effort estimation measure (RBDEE) on the basis of IRBC. The RBDEE measure is systematically derived in order to get a close approximation with existing prevalent practices for effort estimation. Also, the proposed RBDEE measure is validated against various established development effort measures and results obtained validates the claim that the measure is robust, comprehensive and compares well with various categories of development effort estimation. Finally, the paper proposes a test effort estimation (RBTEE) metric based on IRBC of proposed software and, on the basis of result and validation it is observed that the proposed test effort measure follows the trend of all the other established measures in a comprehensive fashion. Also, the values obtained through RBTEE have an approximate mean with use case based measures. This will provide an aid to the developer and practitioner in reducing rework by delivering maximum coverage with minimum number of test cases for improving the test effectiveness. The approaches presented in the paper are also validated with realistic results, to ascertain validity of the proposed measures with the conventional code based approaches. This will enable the developers and analysts to carry out effective, planned and systematic software development in requirement analysis phase of software development life cycle.

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.
3. Yinhuan Zheng et.al. Estimation of the software project effort based on function point, IEEE-ICCSE-2009, pp. 71-85.
4. Stephen Mac Donell , Comparative review of functional complexity assessment methods for effort estimation, Software Engineering Journal,1994,pp. 107-116.
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, November 1983, pp. 639-648
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, 1994, pp-275 - 287
7. Clemmons, R.K., Project estimation with Use Case Points, The Journal of Defense Software Engineering, pp. 18-22.
8. Magne Jorgensen, Dag IK Sjoberg, Impact of effort estimate on software project work, Elsevier, Info & Soft.Tech.2006, pp. 939-948.
9. Bernhard Peschi et.al. Recommending effort estimate method for software project management, IEEE-MC-ACM,2001, pp. 77-80.
10. Noureldin AZ Adem, Zarinah M Kasirun, Automating function point analysis based on the functional and the non-functional requirement text, IEEE Conf. 2010, pp. 664-669.
11. IEEE Computer Society, IEEE recommended practice for software requirement specification, IEEE std 830-1998
12. Geoffrey K. Gill, Chris F. Kemerer, Cyclometric complexity density and software maintenance productivity, IEEE transactions on software engineering vol.17,no.12,december 1991, pp. 1284-1288.
13. Charles R Symons. Function point: Difficulties and Improvements, IEEE Transactions on Software Engineering, Vol.14, No.1, (Jan. 1988), pp. 2-11.
14. IEEE Standard for Software Productivity Metrics," IEEE Std 1045-1992 , pp. 1-38, 1993
15. Sharma Ashish, Kushwaha D.S., NLP/ POS tagger based requirement analysis and its complexity analysis, ACM SigSoft, Jan 2011, Vol.36, No. 1, pp. 1-14.
16. Sharma Ashish, Kushwaha D.S., Complexity measure based on requirement engineering document and its validation, IEEE Conf. on Computer and Communication Technology, ICCCT 2010, pp. 608-615.
17. M.H. Halstead, Elements of Software Science, New-York: Elsevier, 1977

18. DS Kushwaha, AK Misra, Software Test Effort Estimation, ACM SigSoft, Software Engineering Notes, Vol. 33 No. 3, May 2008, pp. 1-6.
19. The Standish group research for staggering bugs and effort, http://standishgroup.com
20. Kuo Chung Tai, Program Testing Complexity & Test Criteria, IEEE Transactions on Software Engineering Vol. SE-6, No.6 , November 1980 pp. 531-538
21. Muthu Ramachandran, Requirements-Driven Software Test: A Process Oriented Approach, ACM Sigsoft, Software Engineering Notes Vol. 21, No. 4, pp. 66-70, July 1996
22. Johannes Ryser,Stefan Bernaer, Martin Glinz On The State Of Art In Requirements- Based Validation And Test of Software, University of Zurich , 1999, pp. 1-16
23. Suresh Nageshwaran, Test Effort Estimation Using USE CASE Points, Quality Week 2001, San Francisco, California USA, 2001, pp. 1-6.
24. Boris Vaysburg , Luay H. Tahat, Bogdan Korel, Dependence Analysis In Reduction Of Requirement Based Test Suites, ACM Journal, 2002, pp. 107-111
25. Ian Holden, Dave Dalton, Improving Test Efficiency Using Cumulative Test Analysis, Proceedings of the Testing: Academic and Industrial Conference – Practice and Research Techniques (TAIC-PART'06), 2006, pp. 152-158.
26. Antonio Bertilino, Software Testing Research: Achievements, Challenges and dreams, IEEE -Future of software engineering-FOSE, 2007, pp. 85-103.
27. Eduardo Aranha, Filipe de Almeida, Thiago Diniz, Vitor Fontes, Paulo Borba, Automated Test Execution Effort Estimation Based On Functional Test Specification, Proceedings of Testing : Academic and Industrial Conference Practice and Research Techniques, MUTATION 07, 2007, pp. 67-71.
28. Ajitha Rajan, Michael W Whalen, Mats P. E. Heimdahl Model Validation Using Automatically Generated Requirements-Based Test", 10th IEEE- High Assurance Systems Engineering Symposium, 2007, pp. 95-104.
29. Aranha E, Borba P, Test Effort Estimation Model Based On Test Specifications, Testing : Academic and Industrial Conference- Practice and Research Techniques, IEEE Computer Society, 2007, pp-67-71
30. Harry M Sneed, Testing Against Natural Lang. Requirements, 7th International Conference on Quality Software (QSIC- 2007), pp. 380-387
31. Eero J Uusitalo, Marko Komssi, Marjo Kauppinen, Alan M. Davis, Linking Requirement And Testing In Practice, 16th IEEE International Requirement Engineering Conference, IEEE-Computer Society, 2008, pp-265-270
32. ZHU Xiaochun, ZHOU Bo, WANG Fan, QU Yi CHEN Lu, Estimate Test Execution Effort at an Early Stage: An Empirical Study, International Conference on Cyber World , IEEE Computer Society, 2008, pp- 195-200
33. Qu Yi Zhou Bo, Zhu Xiaochun, Early Estimate the Size of Test Suites from Use Cases, 15th Asia-Pacific Software Engineering Conference, IEEE Computer Society, 2008, pp-487-492
34. Tibor Repasi, Software Testing- State Of The Art And Current Research Challenges, 5th IEEE - International Symposium on applied Computational intelligence and Informatics, May 28-29, 2009, Romania, 2009, pp-47-50
35. Deniel Guerreiro e Silva, Bruno T. de Abreu, Mario Jino, A Simple Approach For Estimation of Execution of Function Test Case, IEEE-International Conference on Software Testing Verification and Validation, 2009.pp 289-298
36. Erika R. C De Almeida, Bruno T. de Abreu, Regina Moraes, An Alternative Approach to Test Effort Estimation Based on Use Case, IEEE-International Conference on Software Testing Verification and Validation, 2009, pp-279-288
37. Veenendal. E e Deckkers. T. Test point analysis: a method for test estimation in project control for software quality", edited by Rob Kusters Arian Cowderoy, Fred Heemstra e Erik van, Shaker publishing, pp-1-16.

Table 7: Comparison between proposed Development effort measure v/s other established effort estimation

	Name of Program	COCOMO (pm)	COCOMO II (pm)	Watson Felix (pm)	Bailey Basili (pm)	Boehm Simple (pm)	FPA (FP)	Use case (UCP)	RBDEE (pm)
1	Bit Stuffing	0.1559	0.1463	0.4864	5.5356	0.2079	40.66	6.1773	0.19737
2	Matrix Addition	0.0604	0.0547	0.2138	5.5124	0.0805	39.59	6.1773	0.19737
3	Matrix Add. & Mul.	0.167	0.1571	0.5162	5.5384	0.2226	40.66	6.1773	0.2498
4	FCFS	0.0498	0.0448	0.1811	5.5101	0.0665	40.66	5.888	0.169
5	Knapsack	0.1141	0.1059	0.3713	5.5252	0.1522	35.31	5.888	0.1631
6	LZW	0.136	0.127	0.4322	5.5306	0.1814	35.31	5.888	0.1631
7	Round Robin	0.1514	0.142	0.4744	5.5345	0.2019	40.66	6.1773	0.19737
8	CRC	0.4335	0.4231	1.1802	5.6102	0.5781	46.01	7.966	0.2295
9	Prime Number	0.0436	0.039	0.1612	5.5087	0.0581	35.31	6.1773	0.1631
10	Semaphore	0.112	0.1038	0.3651	5.5247	0.1493	46.01	7.966	0.2295
11	Bubble Sort	0.0625	0.0567	0.2203	5.5129	0.0833	35.31	5.888	0.1631
12	Palindrome	0.0415	0.0371	0.1546	5.5082	0.0553	31.03	6.1773	0.1265
13	Calculator	0.0795	0.0728	0.2715	5.5169	0.1061	33.17	7.966	0.3401
14	RSA	0.1316	0.1228	0.4201	5.5295	0.1755	69.55	6.1773	0.3641
15	Linear Search	0.0562	0.0507	0.2008	5.5115	0.0749	44.94	6.1773	0.288

Table 8: Comparison between proposed Test effort estimation v/s other established test effort estimation

Sl.	Program	Use Case (Man-hr)	Test Spec (Man-hr)	Scenario based (Man- hr)	Test Exe. (Man-hr)	Exp. based approach	CICM (CU)	Test Effort (Man-hr)	RBTEE (Man-hr)
1	Bit Stuffing	277.4	182	24.80	598	36.38	329.034	44.99	73.22
2	Matrix Addition	273.6	161	22.2	529	35.31	437.14	43.689	63.57
3	Matrix Add. & Mul.	410.4	273	33.66	897	67.41	624.925	78.119	103.42
4	FCFS	241.4	133	15.98	437	72.76	17.11	90.04	45.01
5	Knapsack	244.8	147	18.20	483	110.09	256.75	62.23	55.90
6	LZW	241.4	140	15.98	460	186.74	1092.92	96.66	48.64
7	Round Robin	273.6	161	22.64	529	243.28	232.266	120.495	63.57
8	CRC	345	231	31.82	759	622.13	1899.43	263.5	91.14
9	Prime Number	269.8	154	19.18	506	88.011	79.45	51.64	56.02
10	Semaphore	326.6	217	25.75	713	115.74	44.375	64.882	79.12
11	Bubble Sort	265.2	154	18.25	506	72.02	182.33	43.67	51.44
12	Palindrome	277.4	175	23.53	575	61.17	17.91	38.38	36.42
13	Calculator	281.8	252	25.58	828	60.81	65.93	35.425	154.48
14	RSA	334.3	217	23.31	713	162.46	94.80	74.279	199.77
15	Linear Search	296.4	175	25.75	575	96.916	41.99	55.59	118.19

Authors

Ashish Sharma received his M.Tech. Degree in Computer Science & Engineering. from UP Technical University, Lucknow, India in the year 2006. He is presently pursuing his Ph.D. in Computer Science & Engineering from Motilal Nehru National Institute of Technology, Allahabad, India under supervision of Dr. D.S. Kushwaha. He is presently associated with the GLA Univ ersity, Mathura as a Reader in the Department of Computer Engineering & Applications. He is having experience of 13 years. His research interest area includes Software Engineering, Requirement Engineering, Software Complexity and Software Testing.

Dr. D.S. Kushwaha received his Doctorate Degree in Computer Science & Engineering from Motilal Nehru National Institute of Technology, Allahabad, India in the year 2007. He is presently working with the same Institution as an Associate Professor in the Department of Computer Science & Engineering. He is having over two decades of academic and research experience. His research interests include areas in Software Engineering, Distributed Systems, Service Oriented Architecture, Web Services and Data Structures. He has over 60 International publications in various Conferences & Journals.