EFFORT ESTIMATION OF COMPONENT BASED SOFTWARE DEVELOPMENT (CBSD) LIFECYCLE USING FUZZY LOGIC

By

Jahanzaib Khan

2010-NUST-MS PhD-CSE (E)-08

Submitted to the Department of Computer Engineering
In partial fulfillment of the requirements for the degree of

Master of Science
in
Computer Software Engineering

Advisor:
Dr. Aasia Khanum

College of Electrical & Mechanical Engineering
National University of Science and Technology
2013

DECLARATION

I hereby declare that I have developed this thesis entirely on the basis of personal efforts under the auspices, sincere guidance and supervision of Dr. Aasia Khanum. All the sources used in this thesis have been cited. No portion of the work presented in this thesis has been submitted in support of any application for any other degree of qualification to this or any other university or institute of learning.

Jahanzaib Khan

ACKNOWLEDGEMENTS

First of all thanks to Almighty Allah, The most Merciful and The most Beneficent. It was impossible to complete this work without His help.

Then, I would like to express my most sincere appreciation to my supervisor Dr. Aasia Khanum, for his continuous support, intellectual guidance and critical remarks during the whole period of this work. I also thank to all the committee members for their acceptance to become the members of guidance and evaluation committee of this thesis and sparing their precious time for reviewing the manuscript.

My special thanks are for the survey participants who spent their time and shared their views and experience. I would like to thank to Mr. Aslam Jarwar for his assistance in Java, for implementation of the CBSD effort Estimation Model.

I would like to thank to my family for their continuous patience, understanding and emotional support during these years.

ABSTRACT

In this era, no one denies the importance of software reuse because software systems are growing and becoming complex with every passing day. Component Based Software Development (CBSD) emerged as a software creation approach with the concept of reusability. In this approach, Software Components which are common among different software applications are reused rather than being written from scratch for every application. CBSD technique is of keen interest to researchers and practitioners as they hold promise to support the timely and cost effective development of large-scale complex systems. It is becoming imperative that effort involved in CBSD may be accurately estimated to attain maximum benefits of the approach.

Effort estimation is one of the major tasks in software project management. The literature shows several efforts estimation models of CBSD but each model does have their own pros and cons. Furthermore, different effort estimation models primarily focuses on the efforts involved in component's integration activities. Moreover, all phases of CBSD lifecycle are unaddressed by existing effort estimation models. Thus, the need to estimate effort involved in CBSD lifecycle is an ongoing challenge.

In this research focus is on the effort estimation of CBSD lifecycle with the help of Fuzzy Logic approach. For the purpose, it was necessary to have a comprehensive CBSD lifecycle model which can be made the basis of effort estimation in CBSD. Thus, first in this study a Circular Process Model (CPM) for CBSD lifecycle is proposed. CPM contains the strengths and weaknesses of the existing CBSD lifecycle models with the focus on rejuvenation of one phase in subsequent phases of the lifecycle. CPM is also validated using the Process Quality Measurement Model (PQMM) [19] and by comparing with the existing CBSD lifecycle process model of Hazleen Iris et al [13]. Then, effort estimation model for CBSD lifecycle is proposed on the basis of CPM. The proposed effort estimation models is also implemented and validated with the help of a case study. Fuzzy logic is used in the implementation as it is more appropriate when the systems are not suitable for analysis by conventional approach or when the available data is uncertain, inaccurate or vague.

Table of Contents

DECLARATION II

ACKNOWLEDGEMENTS III

ABSTRACT IV

List of Figures VIII

List of Table IX

CHAPTER 1 1

INTRODUCTION 1

1.1. Component Based Software Development 1

1.2. Significance of Effort Estimation 1

1.3. Problem Statement 2

1.4. Problem Decomposition 2

1.5. Proposed Models 2

1.5.1. CBSD Lifecycle Model 2

1.5.2. CBSD Lifecycle Effort Estimation Model 3

1.6. Thesis Outline 3

CHAPTER 2 5

LITERATURE REVIEW 5

2.1 Literature related to CBSD Lifecycle Model 5

2.1.1. EPIC, Cecilia Albert et al 5

2.1.2. Qureshi and Hussain 5

2.1.3. Sommerville 5

2.1.4. W Model, Kung-Kiu Lau et al 5

2.1.5. Jason H. Sharp et al 7

2.1.6. Classification Model, Gerald Kotonya et al 7

2.1.7. M. Morisio et al. 7

2.1.8. Component-Based Software Development Process, EhsanKouroshfar et al. 7

2.1.9. MyCL Process Model, Hazleen Iris et al 7

2.1.10. AnasBassam AL-Badareen et al 8

2.1.11. Knot Model, Rajender Singh Chhillar et al 8

2.1.12. Umbrella Model, Anurag Dixit et al 8
2.1.13. Y Model, Luiz Fernando Capretz 8
2.1.14. V Model, IvicaCrnkovic et al. 8
2.1.15. Elite Model, LataNautiyal et al. 9
2.1.16. X Model, Gill N. S. et al 9
2.2. Literature related to CBSD Effort Estimation Model. 9
2.2.1. SAIC Model 9
2.2.2. Stutuzke's Model 9
2.2.3. Ellis's Model 10
2.2.4. Aoyama's Model 10
2.2.5. ABB Model 10
2.2.6. COCOTS Model 10
CHAPTER 3 : PROPOSED MODELS 12
3.1 Proposed Lifecycle Model 12
3.1.1. Domain Engineering 13
3.1.2. Requirement Analysis 13
3.1.3. Component Analysis 14
3.1.4. Component Assurance 14
3.1.5. Architectural Design 14
3.1.6. Component Integration 15
3.1.7. Deployment 15
3.1.8. Maintenance 15
3.2. Proposed Effort Estimation Model 17
3.2.1. Effort Parameters 18
3.2.2. Use of Fuzzy Logic 20
3.2.3. Application Development 26
CHAPTER 4 28
VALIDATION 28
4.1 Validation of the Proposed Lifecycle Model 28
4.1.1. By Comparing with Existing Process Model 28
4.1.2. Using Process Quality Measurement Model (PQMM) 29
4.2. Validation of the Proposed CBSD Lifecycle Effort Estimation Model 31
4.2.1. Conducting a Survey/ Case Study 31

4.2.2. By Measuring Specificity and Sensitivity 33
CHAPTER 5 35
CONCLUSION AND FUTURE WORK 35
5.1. Conclusion 35
5.2. Future Work 36
Reference 37
APPENDIX – A: Survey 40
APPENDIX – B: Linguistic Variables 46
APPENDIX – C: Specificity and Sensitivity Calculations 48
APPENDIX – D: CPM Validation Matrix using PQMM. 49
APPENDIX – E: CPM Validation Calculations using PQMM. 52

List of Figures

Figure 1 : Idealized Circular Process Model 12
Figure 2 : Proposed Circular Process Model 13
Figure 3 : Timeline of Phases in CPM 15
Figure 4 : Chart of Membership Function for Testing Methodology Effort Parameter 23
Figure 5 : Chart of Membership Function for Success Criteria Effort Parameter 24
Figure 6 : Chart of Membership Function for Effort 24
Figure 7 : Front-end of the application 27
Figure 8 : PQMM Chart for CPM 29
Figure 9 : Lifecycle Activities Accuracy Graph of Industrial Survey 32
Figure 10 : Lifecycle Phases Accuracy Graph of Industrial Survey 33
Figure 11 : Specificity and Sensitivity Graphs 34
Figure 12 : Membership Function for Effort Parameters with Three Linguistic Terms .... **Error! Bookmark not defined.**
Figure 13 : Membership Function for Effort Parameters with Two Linguistic Terms **Error! Bookmark not defined.**

List of Table

Table 1 : Strengths and Weaknesses of CBSD Process Models 7
Table 2 : Comparison of Existing CBSD Effort Estimation Models 11
Table 3: Proposed Circular Process Model (CPM) Phases 17
Table 4 : Identified Effort Parameters 19
Table 5 : Function Block (Sample of Unit Testing Activity of Components Provision Phase) 21
Table 6 : PQMM [19] Quality Attributes Values for CPM 30
Table 7: Activity-wise Accuracy Results of Industrial Survey 31
Table 8 : Phase-wise Accuracy Results of Industrial Survey 32
Table 9 : Specificity and Sensitivity Results 34

CHAPTER 1

INTRODUCTION

1.1. Component Based Software Development

Traditionally, software products are built from the scratch, which requires tedious effort, ample resources and plenty of time. As a result, products arrive late into market. This approach works fine when the software products are small and simple. Today, software products have become very large and complex which demands innovation in software development field too. Accordingly, Component Based Software Development (CBSD) emerged with the concept of software reuse and it is gaining high importance day by day among software development organizations.

Software developers believe that many identical component(s) may be found or required by the different software systems. Component Based Software Development (CBSD) emphasizes the reuse of those identical components by avoiding the development again and again from the scratch for every new system. CBSD offers several advantages over traditional software development approaches; including flexibility in development, fast time-to-market, better quality of software, parallel development and cheaper cost of the product.

1.2. Significance of Effort Estimation

Effort is the that specific time period, which consumed working on a project from its inception to completion.

In addition, Effort Estimation is the process of measuring or assessing the effort required for the project. Effort Estimation is the most difficult and important activity in project management. Without good effort estimate, it is almost impossible to devise an effective planning for the software project.

Not only this, a proper effort estimation method is a requirement for Software Project Planning Key Processing Area of CMM level 2. “Good estimation methods are available for projects” is the requirement of “Integrated Software Management” Key Process Areas of

CMM level 3. Use of past estimation data for future projects is the requirement for "Quantitative Process Management" KPA of CMM level 4.

1.3. Problem Statement

Effort estimation is an important job in management of a project. Not only this, effort estimates are the input of every economic decision of the project carried out by the project manager. Thus, accurate estimation of effort is very crucial for the successful completion of project. If improper or unrealistic estimates were made the basis of a project then either project will be challenged in one of the three aspects i.e. time, schedule and scope or it will leads to failure.

By keeping in view the importance of effort estimation it is necessary that *a comprehensive rule based model is developed which estimates lifecycle effort in CBSD at acceptable accuracy level.*

1.4. Problem Decomposition

To devise an effective solution, problem statement is decomposed in following tasks:

- To develop an Enhanced CBSD Lifecycle Model.
- To Enhance Effort Parameters.
- Preparation of a Rule Based Model that incorporates approximate/uncertain input parameters with high accuracy.
- Implementation of Model.
- Testing and validation

1.5. Proposed Models

1.5.1. CBSD Lifecycle Model

CBSD not only differs from traditional software development approaches in terms of advantages and disadvantages, but also with respect to its lifecycle process. *Lifecycle process is the course of activities that produces a new product, and continues through its maintenance.* Software lifecycle is a vague concept [8] and in the case of CBSD there is no universally agreed upon lifecycle process that can be carried out. Several attempts have been made to define an effective process model for CBSD, and all the proposed approaches have their own tradeoffs. Even the IEEE Std. 1517 [20] which deals with software reuse process does not enforce single lifecycle to follow, rather it just tells a minimum set of requirements a software lifecycle must have.

In general however, rejuvenation of one phase of the process in subsequent phases, which is inevitable in CBSD, still needs to be addressed. In this study, we proposed a Circular Process Model (CPM) for CBSD lifecycle whose main focus is to incorporate the rejuvenation of one lifecycle phase in later phases of the lifecycle. Efforts were also made for the validation the proposed CBSD CPM Lifecycle using the PQMM [19] and by comparing with existing CBSD lifecycle process model of Hazleen Iris et al [13].

1.5.2. CBSD Lifecycle Effort Estimation Model

An estimation model defines precisely which values are needed & how these values can be used to compute the effort. Component Based Software development effort estimation requires integration activities to also be considered as opposed to traditional software development which focuses only on development activities. Literature shows that several efforts have been made to estimate the CBSD process effort [23], which are discussed in chapter-2. Despite, no attempt is made towards the effort estimation of complete lifecycle of CBSD. [23].

In this study we also proposed a complete lifecycle effort estimation model for CBSD using Fuzzy Logic. This model is developed with enriched effort parameter/ effort drivers for each activity/phase of the proposed Circular Process Model (CPM) of CBSD lifecycle. The effort parameters are fuzzified using Fuzzy Logic. Comprehensive fuzzy rule base is prepared to produce a crisp effort value of the lifecycle. The application for the proposed effort estimation is prepared in Java Language. Fuzzy Logic is implemented using Fuzzy Control Language [22].

1.6. Thesis Outline

The rest of the thesis is structured as follows. Chapter 2 presents a literature review for CBSD lifecycle Process Models and CBSD Effort Estimation Models. Initially, CBSD and its well known lifecycle Process Models with their strength and weaknesses are discussed. Secondly, the efforts carried out by different researchers in estimating the CBSD effort are discussed. At last, the need for Effort Estimation model of complete lifecycle of CBSD is discussed.

Chapter 3 discussed the proposed models and their implementation. Both models i.e. proposed Lifecycle Circular Process Model (CPM) of CBSD and proposed effort estimation model of CBSD, are discussed separately in detail. Activities/Phases of proposed CPM are discussed with their execution timeline in the process. Output and rejuvenation of each phase also shown with the help of figures and tables. For effort estimation model of CBSD, identified effort parameters with their fuzzy membership functions are explained. Rule formation of Fuzzy Rule Base also highlighted. Categorization of effort parameters in each activity of proposed CPM is also shown.

Chapter 4 includes validation of the both proposed models. CPM lifecycle process model is validated in two ways: First, by comparing with existing lifecycle process model of Hazleen et al[13]. Second, using Process Quality Measurement Model (PQMM) [19]. Proposed Effort Estimation Model is also validated in two ways: First, by a survey research conducted to refine and validate the model. It describes a brief justification for the research method and details about case study design with research questions, data collection and analysis methods.

Chapter 5 gives a short summary of the study and emphasizes the contributions of the model. It further states limitations of the model such as needs of additional quality attribute definitions and deficiencies of some present quality attributes. The propositions for overcoming the limitations and the development of a tool for making the measurement easier are given as future study.

CHAPTER 2

LITERATURE REVIEW

2.1 Literature related to CBSD Lifecycle Model

Despite CBSD novelty in comparison to traditional software development approach, lot of work has been done on the process of CBSD. Brief description of literature reviewed is given below:

2.1.1. EPIC, Cecilia Albert et al

Evolutionary Process for Integrating COTS-Based Systems (EPIC) approach is adapted from Rational Unified Process (RUP) [9]. It rewrites managerial, engineering and acquisition activities to control COTS market in better way [1]. It is a risk-based spiral approach whose phases are same as those of RUP.

2.1.2. Qureshi and Hussain

Process model of Qureshi and Hussain [2] is inclined towards Object Oriented Software development lifecycle. Component Repository is the main contribution of this model but there are no guidelines regarding the addition of components in the repository. Furthermore, when components will be added in the repository is also unclear.

2.1.3. Sommerville

Sommerville proposed sequential lifecycle process model [3], [9] in which components are searched before design; and then modification of requirements will be carried out. In this fashion, design and requirements are based on the components in hand.

2.1.4. W Model, Kung-Kiu Lau et al

Kung-Kiu Lau et al propose W-Model [12] which is mainly focused on Verification and Validation Software Development. They argue that V&V is necessary in both lifecycles i.e. Component development lifecycle and Component Based in Component Development and Component Based Software Development Lifecycle. In this model repository and maintenance phases are not included.

Model Name/ Authors	Strengths/Main Focus	Weaknesses
EPIC, Cecilia Albert et al.[1]	• Risk-based • Disciplined • Spiral-engineering • Facilitate organizations to make and maintain COTS solutions	• Across the life of a large or complex project, many solutions– often overlapping–are created and retired in response to new technology, new components, and new operational needs.
Qureshi and Hussain[2]	• Central Repository	• Not revealed when and how components will be added into repository
Sommerville[3]	• Component Searching before design • Reusability	• Phases like Domain Engineering and Maintenance are missing
W Model, Kung-Kiu Lau et al [12]	• Verification and Validation for both lifecycles i.e. Component lifecycle and CBSD lifecycle.	• Maintenance and Deployment phases are missing. • Repository Missing
Jason H. Sharp et al[18]	• Dual Life cycle Model • Design Science based recommendations • Reusability	• Maintenance and Deployment phases are missing. • Repository Missing
Classification Model, Gerald Kotonya et al[4]	• CBSEnet Knowledgebase. • Management • Short term and long term objectives	• Only Short term objectives are focused.
M. Morisio et al [14]	• Vendor involvement throughout lifecycle • Bi-directional information flow.	• Covers only development (i.e. No maintenance). • Unit Testing in reduced activities
Component-Based Software Development Process(CBSDP), EhsanKouroshfar et al[17]	• Comprehensive stages and task process patterns	• Generic • Not all stage process patterns are mandatory.
MyCL Process Model, Hazleen Iris et al[13]	• Simplicity	• No unit testing • Several included processes not described
AnasBassam AL-Badareen et al [16]	• Reusability • Central Repository • Empirical Validation • Systematic Framework • Discuss Dual Lifecycle	• Maintenance discussed separately and not in development-with-reuse lifecycle. • Only deals with in-house development
Knot Model, Rajender Singh Chhillar et al [5]	• Reusability[5] • Easy Planning [5] • Requirements clear [5] • No complexity of software applications[5] • Reduces risk and development time[5] • Reduces cost[5] • Applicable on larger & complex systems[5]	• Selecting a right component is difficult[5] • Reservoir may be huge or difficult to manage[5]
Umbrella Model, Anurag Dixit et al[8]	• Verification or Testing	• Costly and time consuming due to testing or verification in each phase
Y Model, Luiz Fernando Capretz [7]	• Reusability[5] • Solving by analogy[5] • Follows both top down and bottom up approach[5]	• Iteration and overlapping during process[5] • Does not define a component model
V Model, IvicaCrnkovic et al[10]	• Verification and Validation • Supports Unit Test and System Test. • Central Repository	• No Domain Engineering • No System Deployment.

Elite Model, LataNautiyal et al [11]	• Reusability • Testing or Verification	• Unit Testing is missing • Design/Architecture Phase is missing
X Model, Gill N. S. et al [6]	• Reusability[5] • Clear requirements[5] • Suitable for large systems[5]	• Increases complexity[5] • No risk analysis[5] • Increase cost[5]

Table 1 : Strengths and Weaknesses of CBSD Process Models

2.1.5. Jason H. Sharp et al

Jason H. Sharp et al [18] proposed lifecycle model with design science based recommendations. They discussed phases of component development and system development separately. They did not include the domain analysis phase in system development lifecycle.

2.1.6. Classification Model, Gerald Kotonya et al

Gerald Kotonya et al proposed Classification lifecycle Model for CBSD [4], [2] whose center of attention is CBSEnet knowledge Base. In this model both short and long term objectives are discussed but it addresses only short term objectives.

2.1.7. M. Morisio et al

M. Morisio et al [14] proposed COTS lifecycle model in which emphasis is put on the involvement of vendor throughout the lifecycle. In this study new activities and roles are identified related to vendor. Limitations of the model are that it only focuses on development phase. Maintenance phase is also missing.

2.1.8. Component-Based Software Development Process, EhsanKouroshfar et al.

Ehsan Kouroshfar et al [17] proposed Component Basedd Software Development Process (CBSDP). It is a generic process derived by reviewing seven CBSD based methodologies like FORM, RUP and CORBA etc. One limitation in the process is that all the activities are not mandatory due to its generic nature; thus, difficult to implement.

2.1.9. MyCL Process Model, Hazleen Iris et al

MyCL Process Model was proposed by Hazleen Iris et al [13]. It is an attempt to make the lifecycle process very simple, but in doing so several phases or process have lost necessary detail. Furthermore, requirements and architecture become fixed before component selection. Unit testing is also eliminated in this model.

2.1.10. AnasBassam AL-Badareen et al

AnasBassam Al-Badareen et al [16] in their research focused on reusability and proposed two lifecycle processes i.e. build-for-reuse and build- by-reuse. They discuss in detail the transfer of build-for-reuse process to build-by-reuse process. Central repository is also focused in this study. This model treats maintenance process separately, which should be part of the lifecycle. One limitation is that this model only deals in-house development.

2.1.11. Knot Model, Rajender Singh Chhillar et al

Knot Model [5] was proposed by Rajender Singh Chhillar et al. In each phase of this model risk analysis and feedback is focused which ultimately improves the quality of the system. Reusability and estimation is also used in each phase to reduce the cost. In addition, the developed Component Based Software System (CBSS) is also present in pool for utilization. Limitations of this model are huge repository size and difficulty in selecting the right component.

2.1.12. Umbrella Model, Anurag Dixit et al

Umbrella Model [8] was proposed by Anurag Dixit et al. This model mainly revolves around testing or verification. Authors argue that testing or verification must be included as an ongoing process throughout lifecycle. In this model testing or verification phase overlaps and repeats in every phase.

2.1.13. Y Model, Luiz Fernando Capretz

Y Model [7] was proposed by Luiz Fernando Capretz. This model supports iteration and overlapping, if required. Furthermore, it permits both top-down and bottom-up approach of software development. However, definition of component model is overlooked by this model.

2.1.14. V Model, IvicaCrnkovic et al.

V Model for CBSD [10] was proposed by Ivica Crnkovic et al. This model is an adaptation of V Model which is widely used in the industry for traditional software developments. This model also focuses on verification and validation. However, steps like Domain Engineering and system deployment are missing.

2.1.15. Elite Model, LataNautiyal et al.

Elite Model [11] proposed by Lata Nautiyal et al. also mainly concentrate on testing or verification as continuous activities. During development and maintenance, this model promotes software reusability.

2.1.16. X Model, Gill N. S. et al

X Model [6, 11, 5] is proposed by Gill N. S. et al. Focus of this model is also software reusability. This model is best for large software developments it is quite complex and has overlapping activities. This model ignores feedback and risk analysis.

2.2. Literature related to CBSD Effort Estimation Model.

Literature shows that despite CBSD approach novelty several effort estimation models have been proposed. A great work has been conducted regarding the consolidation of literature on CBSD Effort Estimation models in [23]. In this work effort models are divided into three categories on the basis of their modeling techniques. Following effort estimation models are discussed in [23].

2.2.1. SAIC Model

It is developed in the early 1990s at the Science Applications International Corporation (SAIC), California [23, 26]. Focus of this model is the end-user cost of adopting a particular component into a larger system.

$$\text{Estimated Cost} = \text{Licensing Cost} \times \text{No. of Required License} + \text{Training Cost} + \text{Glue Code Cost}$$

The weakness of this model is that it does not consider the component searching and selecting efforts. Some of the important cost factors covered by SAIC model are licensing and training costs. This model also not provides details of determining the effort of glue code development [23, 27].

2.2.2. Stutuzke's Model

This model concentrates on the volatility cost which is one of the major factors in cost of using software component [23, 27, 28]. The rate of component's version release by its vendor is called component volatility.

EAC = Component Volatility × Architectural Coupling × Interface Size (Cost of Screening + Change Cost). This model only focuses on volatility and ignores other important cost drivers. Furthermore, this model has not been implemented [23,29].

2.2.3. Ellis's Model

This model mainly focuses on component integration phase and used 17 cost drivers to calculate effort. This model is implemented and calibrated but calibrated data set is not publically available.

2.2.4. Aoyama's Model

This model is based on some suppositions. For example, Aoyama completely neglects unit testing and consider effort of CBSD system testing tantamount to traditional software development system testing. However, in reality CBSD testing demands extra effort and time than traditional software testing demands [23, 30, 31]. Similarly, Unit testing may simply not be neglected in CBSD.

2.2.5. ABB Model

This model is based on GQM (goal-question-metrics) approach. This model may be used to decide that whether or not the CBSD approach followed because it provides the economic analysis of CBSD [23].

2.2.6. COCOTS Model

It is the most inclusive effort estimation model of CBSD. It is modeled as an extension of COCOMO-II. This model is basically divided in three steps: First it calculates the assessment effort, then tailoring effort and finally integration effort. All three are combined to calculate the total effort involved. The focus of this model is the integration activities. This model is based on two things: the source code of the COTS is not available to developer and the future evolution of the COTS is not under the control of develop [23].

Model Name	Focus Point	Weak Points
SIAC Model	End-user cost of adopting Licensing and training costs	•Does not consider the component searching and selecting efforts •Do not provide details of determining the effort of glue code development[27]
Stutzke's Model	Volatility Cost	•Ignores other important cost drivers •Not been implemented.[29]
Ellis's Model	-Component integration phase and 17 cost drivers	•Calibrated data set is not publically available
Aoyama's Model	Economic model	•Neglects unit testing
ABB Model	Economic model	-Ignores other important cost drivers
COCOTS	Integration activities	-Ignores other important cost drivers.

Table 2 : Comparison of Existing CBSD Effort Estimation Models

CHAPTER 3 : PROPOSED MODELS

3.1 Proposed Lifecycle Model

The CPM model is derived by embracing the strengths of the reviewed process models and eliminating their weaknesses. The main focus of this model is to address the rejuvenation of earlier phase(s) during the execution of subsequent phase(s), which is certain in CBSD. CPM comprises eight phases which are further divided into seventeen activities as shown in Table 3.

In an idealized CBSD process one phase follows another, as shown in Figure 1. Phases start from Domain Engineering and continue till Maintenance, in clockwise direction. In Idealized CBSD process no phase repeat itself as all phases execute sequentially. But this is the case which one can only dream of. For instance, what happens when required components are not available in Component Assurance phase? Does the development team not change the requirement(s)? If this is the case then we are admitting that requirement analysis step will be revisit after component assurance. This is mainly focused in our proposed CPM.

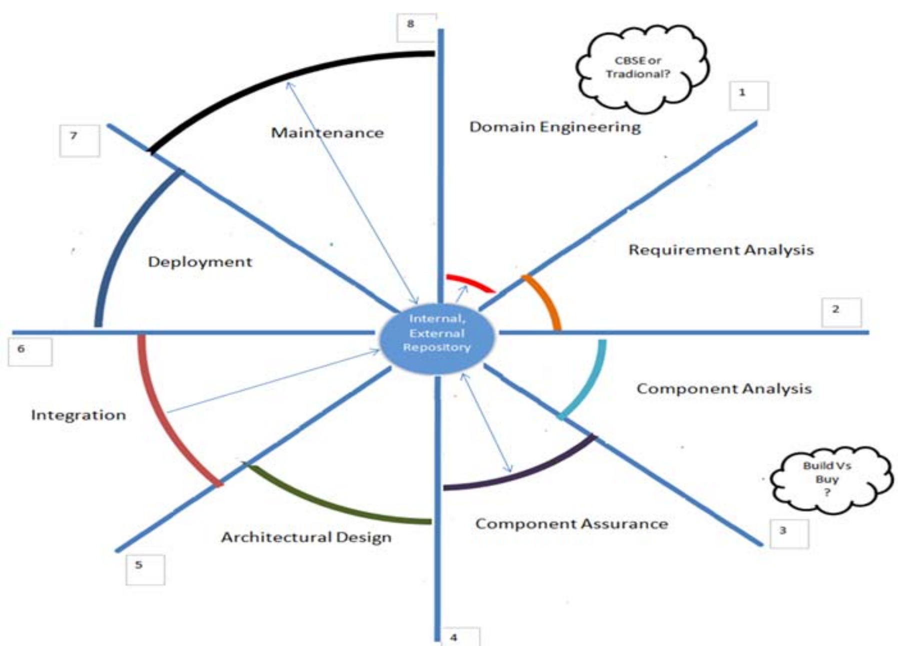

Figure 1 : Idealized Circular Process Model

In CPM, as shown in Figure 2, phases are represented with circles. The inner most circle represents the Domain Engineering phase and the outer most represents the maintenance phase. Phases in the proposed model are executed in clockwise direction from Domain Engineering to Maintenance. Outermost circle in each phase represents the currently executing phase while inner circles in a phase express that they may re-occur during the executing phase.

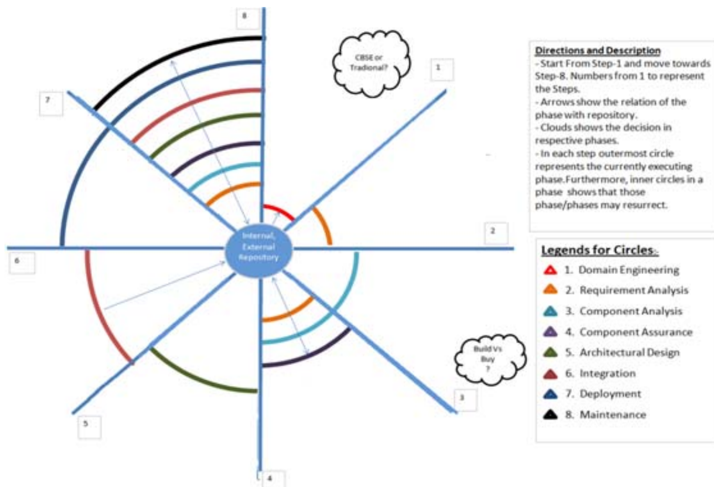

Figure 2 : Proposed Circular Process Model

3.1.1. Domain Engineering

In Domain Engineering identical areas across different applications in a domain are recognized as having common understanding on the basis of application domain analysis [7]. Domain Engineering is the also an important phase of IEEE Std. 1517 which specifies cross project processes. Cross project processes facilitate software reuse in CBSD.

At the end of this phase expert judgment is required for the decision that whether the specified requirements can be accomplished using CBSD approach? If not then it would be wise to adapt traditional approach. It is fact that this decision is very daunting and only an expert may decide it.

3.1.2. Requirement Analysis

In Requirement Analysis, software requirements are first elicited and then specified. The final outcome of this phase is requirement specification document. This phase is not one-time activity, especially in CBSD where it may untill the successful completion of the component assurance phase (See Figure 2).

3.1.3. Component Analysis

Component Analysis phase encompasses the process of identification of components from the specified requirements and then specification of the identified components. In this phase, requirement specification document is reviewed for component identification and specification. Outcome of this phase is requisites component specification document.
At the end of this phase another decision is required and another test of expert's abilities is demanded. Here, expert decided on the basis of his experience and identified components whether component development from scratch is better or use of COTS would be beneficial? This decision is necessary because if we plunge directly into the next phase, .i.e. Component Assurance, then it would be very difficult to meet the schedule. It is so because Component Assurance is a time consuming activity and if Components are unavailable then all the exercise of this phase will be futile.

3.1.4. Component Assurance

This phase is an important and distinct phase of CBSD lifecycle. It is distinct because major activities of this phase are not the part of traditional software development approach. In this phase requisite components are searched from the repository. If one fails in finding the requisite component then Requirement Analysis phase is re-executed that in turn re-calls Component Analysis phase. This phenomenon is shown in Figure 2 and Figure 3. Component assurance phase continues till all required components become available. At the end of this phase, the development team has all the requisite components in hand.

3.1.5. Architectural Design

At this stage, final requirements and requisite components are in developers' hands so it is the right time to design architecture of the application. In this phase, component interactions are analyzed to shape the software architecture. Output of this phase is System Architecture description.

3.1.6. Component Integration

In Component Integration phase components are integrated one by one into the system. After integration of each component, system is tested to ensure the smooth functioning. To accomplish the task of component integration new code is required, which works as interface between the component and the system under development. This new code is called Glue Code [15].

3.1.7. Deployment

Deployment is the process of transferring the system to the customer in a fashion that customer feels comfortable with the product; and may be able to enjoy the maximum benefits from it. To ensure successful deployment, training and documentation must be a provided to customer [7].

3.1.8. Maintenance

Maintenance is a system support activity which ensures smooth running of the system and increases product's lifetime. As far as CBSD is concerned, maintenance may be required because of two reasons. First, change in requirement and second, component up-gradation. Change in requirements is also very common cause of maintenance in traditional software but maintenance due to component up-gradation is specific to Component Based Software Systems. It may occur due to the availability of new version of the utilized components in market which need to be replaced.

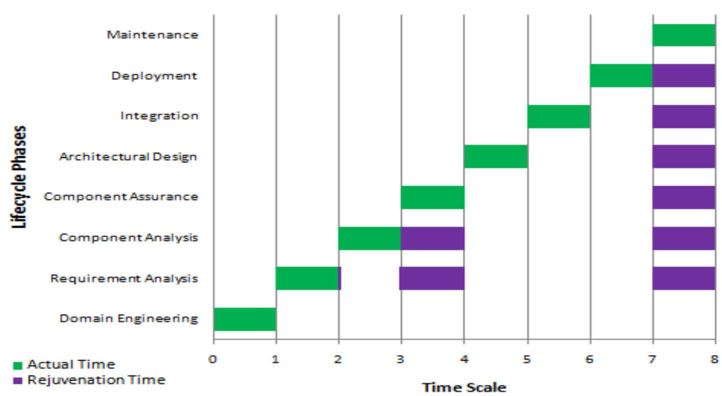

Figure 3 : Timeline of Phases in CPM

Phases	Activities	Description	Output
Domain Engineering	Domain Engineering	It is a process which provides understanding regarding the application domain and help in taking the decision of following CBSD or Traditional approach.	Common Processes of Application Domain

1st Decision: CBSD or Traditional			
Requirement Analysis	Requirement Assessment	It is a comprehensive activity which deals with finalization of requirements with consultation of end-user and domain experts, and refinement of requirements for specification.	System Requirements Specification (SRS) Document
	Requirement Specification	It is the process of preparing requirement specification document from the requirements finalized in requirement assessment activity.	
Component Analysis	Component Identification	It deals with determining required components, by analyzing the requirement specification document.	Requisite Components Specification Document
	Component Specification	In this identified components are completely specified (i.e. interfaces, member functions etc.) to have clear idea of needed components	
2nd Decision: Build Vs. Buy			
Component Assurance	Component Searching	Needed components are searched first in organization's internal repository then from external vendor's repository (if not found in internal repository).	Requisite Components (COTS)
	Component Selection	Best components are selected from the components found (if more than one) in search activity.	
	Component Acquisition	Process of acquiring selected components from the vendor, if not present in organization's internal repository.	
	Tailoring	To set component for apply irrespective of integrated system [15].	
	Unit Test	Ensure component functioning in isolation after component tailoring.	
Architectural Design	Component Architectural Comprehension	Each component's architecture is realized in detail to ensure best possible architecture.	System Architecture
	Application Design	System Architecture is finalized on the basis of available components.	
Integration	Component Adaptation	Each component is adapted for integration into the system by writing glue code.	Component Based Software System(CBSS)
	Integration Test	Ensure that system works well after integration of each component.	
Deployment	Deployment Kit Preparation	User manual, training guide or other relevant material is prepared to ensure user understandability of the product alongwith preparation of executable copy of the product.	User Manual, Training Guide
Maintenance	Substitution	Required if new version of COTS is available.	Component Based Software System(CBSS)
	Evolution	Required when new/change requirements are demanded.	

Table 3: Proposed Circular Process Model (CPM) Phases

3.2. Proposed Effort Estimation Model

A software estimation model defines precisely which values are needed & how these values can be used to compute the effort. In the proposed CBSD lifecycle Effort Estimation model we used proposed CBSD Lifecycle Circular Process Model (CPM). This model has 17 activities and 08 phases. Effort parameters/drivers are identified for each activity from Domain Engineering to Maintenance. Bottom-up approach of effort estimation is used. Effort for each activity is estimated on the bases of identified effort parameters using Fuzzy Logic. Then, combined effort of all activities is calculated to obtain the Lifecycle effort.

It is pertinent to mention here that crisp value is achieved for all activities and total lifecycle effort. Unit of effort may have different meanings for different organizations. For example, an 'ideal hour' for an organization may be the time spent on development activities while for other organizations it may be the time of development activities plus other parallel activities like meetings, presentations, internet surfing etc related to project. Let's discuss the implementation of the proposed model in detail.

3.2.1. Effort Parameters

Effort Parameters / Effort drivers are those factors which are related with any aspect of the project and affect the Effort in any respect, till project completion.

These effort parameters are actually the basic units which help in estimating the effort. Different effort estimation models have utilized different number of effort parameters for their effort estimation. For example, COCOMO-II has 17 effort/cost drivers with five scale factors [24]. Similarly, COCOTS, an extension of COCOMO-II model, [15] have different number of effort parameters.

In this study, 64 effort parameters are used, which are categorized under activities, phases and lifecycle. Out of these 64 effort parameters, 07 parameters are taken from COCOTS model [15], 03 parameters are taken from scale factors of COCOMO-II [15], 02 Parameters are taken from [25]. COCOTS parameters are used under the activities of Component Acquiring and Component Tailoring. Complete list of Effort parameters under activities/phases is shown in table 4.

Table 4 : Identified Effort Parameters

CPM Phases	CPM Activities	Effort Parameters		
		Activity Level	**Phase Level**	**Lifecycle Level**
Domain Engineering	Domain Engineering	NOADA - No. of available Domain Applications		• OC- Organization Culture [25] • PM-Process Maturity[24] • LS- Leadership Skills[25] • TC- Team Cohesion [24] • SC- Stakeholder Cohesion • TSK- Team Skills • TE- Team Experience • TSZ- Team Size • TC- Team Consistency • PS-Project Size • PC-Project Complexity • PP-Project Precedence [24] • UOST-Use of Standard Tools • RW- Rework
Requirement Analysis	Assess	NORS - No. of Requirement Sources	LOEUI- Level of End-User Interest	
		OD - Organizational Diversity		
		UD - User Diversity		
	Specify	NOFR - No. of FRs		
		NONFR - No. of NFRs		
		NOC - No. of Constraints		
		RC - Requirement Clarity		
Component Analysis	Identification	NOFR - No. of FRs	RT – Reuse Type	
		NONFR - No. of NFRs		
	Specification	NOIC - No. of Identified Components		
		NOII - No. of Identified Interfaces		
		NOIMF - No. of Identified Member functions		
		COH - Cohesion		
Component Provision	Search	RS - Repository Size	NOIC- No of Identified Components	
		SS - Search Strategy		
	Select	NOFR - No. of FRs		
		NONFR - No. of NFRs		
		NOADA - No. of available domain applications		
	Acquire	ACPTD - COTS Supplier Provided Training and Documentation[15]		
		ACSEW - COTS Supplier Product Extension Willingness[15]		
		ACPPS - COTS Supplier Product Support[15]		
	Tailoring	NOPTBS - No. of Parameters to be Specified[15]		
		IGS - Input/GUI screen[15]		
		ORL - Output report layout[15]		
		SPS - Security protocols set-up[15]		
	Unit Test	TM - Testing Methodology		
		SC - Success Criteria		
Architectural Design	Component Interaction	NOCF - No. of Components Fashioned		
		CAM - Components Architectural mismatch		
		NOIAMF - No of Interfaces and Membership Functions		
		IC - Interface Complexity		
		Cou –Coupling		
	Application	RF - Requirements Flexibility		

	Design	SF - Schedule Flexibility		
		RA - Resources Availability		
Integration	Adaptation	FP - Function Points		
		NOIAMF - No of Interfaces and Membership Functions		
		AC - Architectural Constraints		
	Integration Testing	TM - Testing Methodology		
		SC - Success criteria		
Deployment	Document. / User Training	NOSTBD - No of Sites to be Deployed		
		TE - Targeted End-user		
		UMDC - User Manual/ Documentation Comprehensiveness		
Maintenance	Substitution	NOCTBR - No. of Components to be substitute.		
	Evolution	SOC - Size of Change		

3.2.2. Use of Fuzzy Logic

For the implementation of the effort estimation we used Fuzzy Logic because it is based on intuition and judgment and does not require any mathematical model. Furthermore, Fuzzy Logic provides smooth transition between members and nonmembers. Fuzzy Logic is also comparatively simple, fast and adaptive. Moreover, it is less sensitive to system fluctuation.

In implementing the proposed model we used and open source Fuzzy Logic Library jFuzzyLogic 2.1. It uses Fuzzy Control Language (FCL). The theory of Fuzzy Logic in the application of control is named Fuzzy Control. The Fuzzy Control is emerging as a technology that can enhance the capabilities of industrial automation. [22]. Fuzzy Control Language ***FCL*** is defined by IEC 1331 part 7 [21].

3.2.2.1. Function Blocks

A Function Block is a FCL program which is used to keep the Fuzzy Control Logic. Function Block specifies I/O parameters, declarations and fuzzy rule base. Function Blocks defined in Fuzzy Control Language FCL can be used in Programs and Function Blocks written in any of the languages [22].

In this study, for implementing the model we define a Function Block for each activity of the proposed CPM lifecycle model. Function Block for the Unit Testing activity of Component Provision phase is shown below:

Table 5 : Function Block (Sample of Unit Testing Activity of Components Provision Phase)

```
FUNCTION_BLOCK componentProvisionUnitTesting

VAR_INPUT
      tm : REAL;    // Effort Parameter Testing Methodology.
      sc: REAL;     // Effort Parameter Success Criteria
      noic : REAL;  // Effort Parameter No. of Identified Components.
END_VAR

VAR_OUTPUT
      effort : REAL;    // Estimated Effort Variable
END_VAR

FUZZIFY tm
      TERM whitebox := (0, 1) (4, 0) ;
      TERM glassbox := (1, 0) (4,1) (6,1) (9,0);
      TERM blackbox := (6, 0) (9, 1);
END_FUZZIFY

FUZZIFY sc
      TERM acceptableerrors := (0, 1) (1, 1) (3,0) ;
      TERM errorfree := (7,0) (9,1);
END_FUZZIFY

FUZZIFY noic
      TERM few := (0, 1) (4, 0) ;
      TERM average := (1, 0) (4,1) (6,1) (9,0);
      TERM many := (6, 0) (9, 1);
END_FUZZIFY

DEFUZZIFY effort
      TERM low := (0,0) (5,1) (10,0);
      TERM medium := (10,0) (15,1) (20,0);
      TERM high := (20,0) (25,1) (30,0);
      METHOD : COG;
      DEFAULT := 0;
END_DEFUZZIFY

RULEBLOCK No1
      AND : MIN;
      ACT : MIN;
      ACCU : MAX;

RULE 1 :
IF tm IS whitebox AND sc IS acceptableerrors AND noic IS few THEN
effort IS medium;
RULE 2 :
IF tm IS whitebox AND sc IS acceptableerrors AND noic IS average THEN
effort IS medium;
RULE 3 :
IF tm IS whitebox AND sc IS acceptableerrors AND noic IS many THEN
effort IS high;
RULE 4 :
```

```
IF tm IS whitebox AND sc IS errorfree AND noic IS few THEN effort IS
high;
RULE 5 :
IF tm IS whitebox AND sc IS errorfree AND noic IS average THEN effort
IS high;
RULE 6 :
IF tm IS whitebox AND sc IS errorfree AND noic IS many THEN effort IS
high;
RULE 7 :
IF tm IS glassbox AND sc IS acceptableerrors AND noic IS few THEN
effort IS low;
RULE 8 :
IF tm IS glassbox AND sc IS acceptableerrors AND noic IS average THEN
effort IS medium;
RULE 9 :
IF tm IS glassbox AND sc IS acceptableerrors AND noic IS many THEN
effort IS medium;
RULE 10 :
IF tm IS glassbox AND sc IS errorfree AND noic IS few THEN effort IS
medium;
RULE 11 :
IF tm IS glassbox AND sc IS errorfree AND noic IS average THEN effort
IS medium;
RULE 12 :
IF tm IS glassbox AND sc IS errorfree AND noic IS many THEN effort IS
high;
RULE 13 :
IF tm IS blackbox AND sc IS acceptableerrors AND noic IS few THEN
effort IS low;
RULE 14 :
IF tm IS blackbox AND sc IS acceptableerrors AND noic IS average THEN
effort IS low;
RULE 15 :
IF tm IS blackbox AND sc IS acceptableerrors AND noic IS many THEN
effort IS medium;
RULE 16 :
IF tm IS blackbox AND sc IS errorfree AND noic IS few THEN effort IS
medium;
RULE 17 :
IF tm IS blackbox AND sc IS errorfree AND noic IS average THEN effort
IS medium;
RULE 18 :
IF tm IS blackbox AND sc IS errorfree AND noic IS many THEN effort IS
high;
END_RULEBLOCK
END_FUNCTION_BLOCK
```

- Definition of the *FUNCTION BLOCK*

```
FUNCTION_BLOCK componentProvisionUnitTesting
```

- Definition of Input and output variables (only ***REAL*** is implemented yet in FCL)

```
VAR_INPUT
      tm : REAL;    // Effort Parameter Testing Methodology.
      sc: REAL;     // Effort Parameter Success Criteria
```

```
      noic : REAL;  // Effort Parameter No. of Identified Components.
END_VAR

VAR_OUTPUT
      effort : REAL;    // Estimated Effort Variable
END_VAR
```

- Fuzzification of input variables. Each input variable is defined in ***FUZZIFY*** block. In each block Linguistic Terms of that input variable is defines along with membership function. Each term is composed by a name and a membership function. E.g.:

```
FUZZIFY tm
      TERM whitebox := (0, 1) (4, 0) ;
      TERM glassbox := (1, 0) (4,1) (6,1) (9,0);
      TERM blackbox := (6, 0) (9, 1);
END_FUZZIFY
```

Three linguistic terms are used to define the Testing Methodology(tm) input variable. For instance term ***whitebox*** uses a piece-wise linear membership function defined by points x_0 = 0, y_0 = 1 and x_1 = 4, y_1 = 0. Same membership functions are chosen for No. of Identified Components (noic) input variable

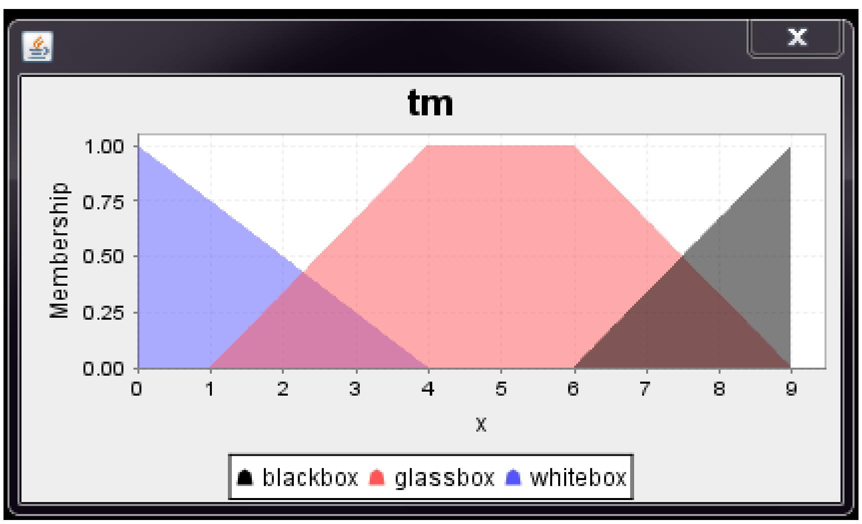

Figure 4 : Chart of Membership Function for Testing Methodology Effort Parameter

Similarly, ***Success Criteria*** variable fuzzify block is define:

```
FUZZIFY sc
      TERM acceptableerrors := (0, 1) (1, 1) (3,0) ;
      TERM errorfree := (7,0) (9,1);
END_FUZZIFY
```

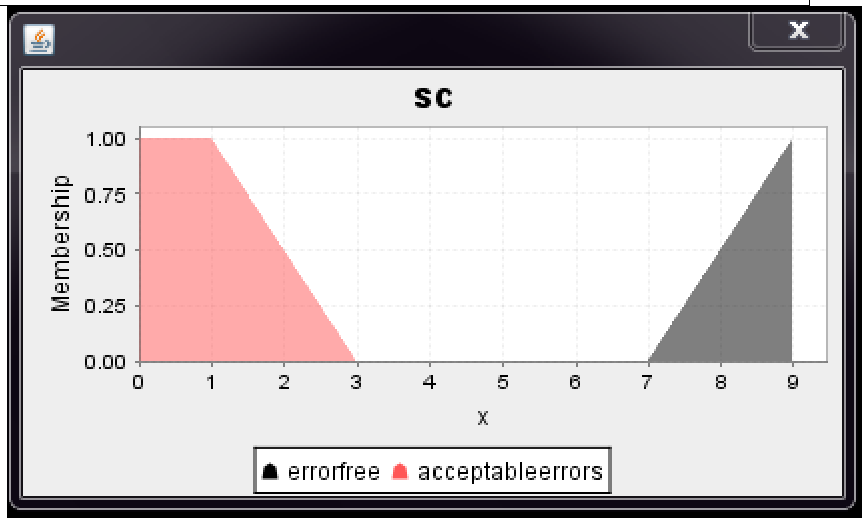

Figure 5 : Chart of Membership Function for Success Criteria Effort Parameter

- Defuzzification of output variable. Output variable are defined in **DEFUZZIFY** block. We have only one output variable in the proposed model that is ***Effort.*** Defuzzification is show below:

```
DEFUZZIFY effort
      TERM low := (0,0) (5,1) (10,0);
      TERM medium := (10,0) (15,1) (20,0);
      TERM high := (20,0) (25,1) (30,0);
      METHOD : COG;
      DEFAULT := 0;
END_DEFUZZIFY
```

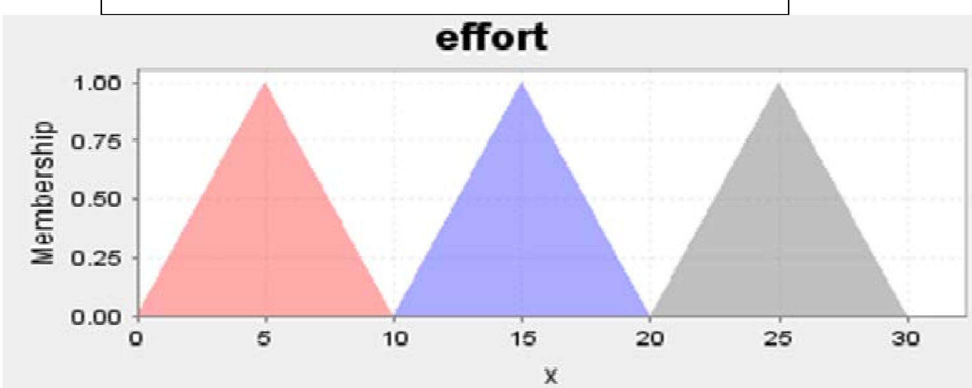

Figure 6 : Chart of Membership Function for Effort

Parameters METHOD in DEFUZZIFY block denotes defuzzification method. In the proposed model 'Center of gravity' is opted for defuzzification and set DEFAULT value to '0' if no rule executes:

```
METHOD : COG;
DEFAULT := 0;
```

- Define Rules using a ***RULEBLOCK***. First some parameters are defined. For the proposed model minimum is used for **AND**. Used Activation (**ACT**) method is also minimum while used Accumulation (**ACCU**) method is maximum :

```
RULEBLOCK No1
AND : MIN;
ACT : MIN;
ACCU : MAX;
```

Then following 18 rules are defined in this **RULEBLOCK.** The **Cartesian product** of input variable's membership function in each activity is adapted, to prepare rules, for maximum coverage of inputs and better estimation.

```
RULE 1 :
IF tm IS whitebox AND sc IS acceptableerrors AND noic IS few THEN
effort IS medium;
RULE 2 :
IF tm IS whitebox AND sc IS acceptableerrors AND noic IS average THEN
effort IS medium;
RULE 3 :
IF tm IS whitebox AND sc IS acceptableerrors AND noic IS many THEN
effort IS high;
RULE 4 :
IF tm IS whitebox AND sc IS errorfree AND noic IS few THEN effort IS
high;
RULE 5 :
IF tm IS whitebox AND sc IS errorfree AND noic IS average THEN effort
IS high;
RULE 6 :
IF tm IS whitebox AND sc IS errorfree AND noic IS many THEN effort IS
high;
RULE 7 :
IF tm IS glassbox AND sc IS acceptableerrors AND noic IS few THEN
effort IS low;
RULE 8 :
IF tm IS glassbox AND sc IS acceptableerrors AND noic IS average THEN
effort IS medium;
RULE 9 :
```

```
IF tm IS glassbox AND sc IS acceptableerrors AND noic IS many THEN
effort IS medium;
RULE 10 :
IF tm IS glassbox AND sc IS errorfree AND noic IS few THEN effort IS
medium;
RULE 11 :
IF tm IS glassbox AND sc IS errorfree AND noic IS average THEN effort
IS medium;
RULE 12 :
IF tm IS glassbox AND sc IS errorfree AND noic IS many THEN effort IS
high;
RULE 13 :
IF tm IS blackbox AND sc IS acceptableerrors AND noic IS few THEN
effort IS low;
RULE 14 :
IF tm IS blackbox AND sc IS acceptableerrors AND noic IS average THEN
effort IS low;
RULE 15 :
IF tm IS blackbox AND sc IS acceptableerrors AND noic IS many THEN
effort IS medium;
RULE 16 :
IF tm IS blackbox AND sc IS errorfree AND noic IS few THEN effort IS
medium;
RULE 17 :
IF tm IS blackbox AND sc IS errorfree AND noic IS average THEN effort
IS medium;
RULE 18 :
IF tm IS blackbox AND sc IS errorfree AND noic IS many THEN effort IS
high;
END_RULEBLOCK
```

3.2.3. Application Development

As discussed in previous section that for Fuzzy Logic implementation Fuzzy Control Language is used. Similarly, for the development of application front-end Java language is used. The IDE used for the application development is Eclipse Helios.

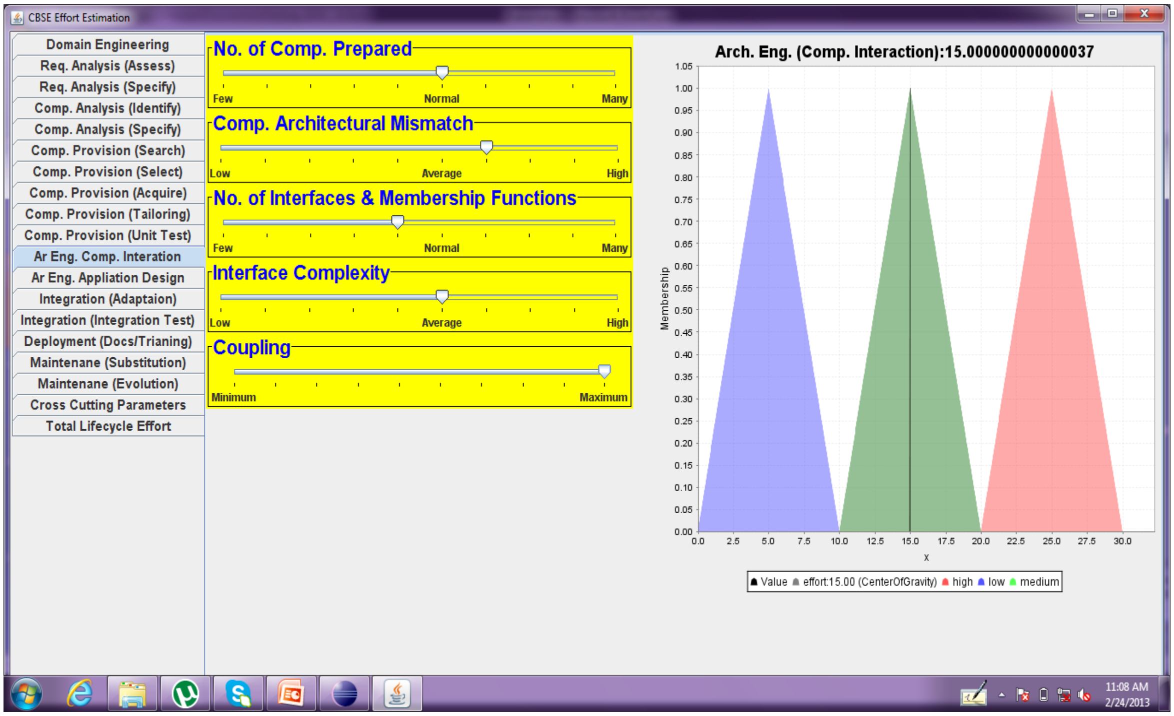

Figure 7 : Front-end of the application

In application each lifecycle activity is shown separately with its specific effort parameters as input variables. Slider Control is used to adjust the inputs. Separated Estimated effort graph are also shown for each activity. For example, in Domain Engineering Phase only one effort parameter ‘No. of Available Domain Applications’ is identified.

CHAPTER 4

VALIDATION

4.1 Validation of the Proposed Lifecycle Model

4.1.1.By Comparing with Existing Process Model

Without comparison it is difficult to say that one thing is better than the other. We chose a state-of-art model, the MyCL Process Model [13] for comparison as this model is also based on integrating the strengths and removing the weaknesses of the existing models.

In MyCL Process Model requirements are finalized at Requirement Analysis phase, as in Waterfall Model, and in component development phase, components are adapted or engineered to comply with requirements. There is no recourse to requirement analysis phase if the requisite component did not found. Only provided thing is developing component from scratch which is not the essence of CBSD. This is not the case in the proposed circular lifecycle model. In circular lifecycle model you can build new component, or you can modify your requirements, as desired.

Architectural Design phase is placed before Component Selection phase, which does not suits CBSD because when you don't have selected component in hands how you can have a frozen architecture? Second there is also no recourse to architectural design phase if the components assumed in architecture did not satisfy the architecture. This problem is resolved by circular model in which architectural design phase is placed after component Assurance phase.

Again, in MyCL process Unit test is removed from the lifecycle by stating *"removing unit testing from the development lifecycle. This removal is obvious, as the system is no longer built from scratch, but from composed components."[13]* In Circular Lifecycle

Model Unit test is included because component tailoring is required which is to set the component to be used irrespective of the integrating system [15]. Thus, unit test is necessary.

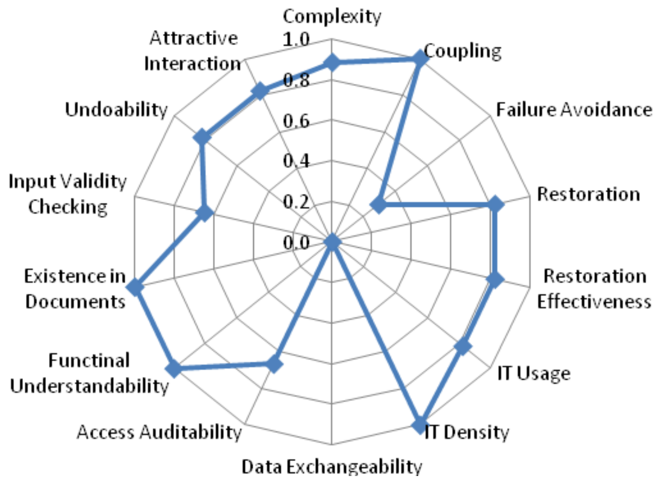

Figure 8 : PQMM Chart for CPM

4.1.2. Using Process Quality Measurement Model (PQMM)

We have validated the proposed circular lifecycle model using Process Quality Measurement Model (PQMM) of Guceglioglu et al [19]. The PQMM provides a set of quality metrics that can be used to evaluate static quality of a software development process. Each of these metrics lies in value between 0 and 1. We have used a subset of these metrics for process evaluation, using only those metrics that were relevant to the process and could be calculated from the process definitions. Table3 shows the metrics (with definitions re-phrased or adapted from [19]).

It can be seen that only failure avoidance attribute of the process requires improvement. Overall validation, however, shows that the model efficiently fulfills PQMM characteristics, implying that the model is very much maintainable, reliable, functional and usable. Model assessment according to PQMM is illustrated in Figure 8.

Table 6 : PQMM [19] Quality Attributes Values for CPM[1]

Quality Characteristics	Quality Sub-Attribute	Metric	Explanation	Value
Maintainability	Analyzability	Complexity	Obtained by subtracting the normalized number of decision points in the process from 1, such that higher the obtained value, lesser the complexity and thereby better the analyzability.	0.9
		Coupling	Examines interactions between process flow and other organizational processes. Obtained by subtracting the number of interactions from 1 so that higher the obtained value, lesser the complexity and thereby better the analyzability.	1.0
Reliability	Fault Tolerance	Failure Avoidance	Here term failure means user-based mistakes which can be avoided using techniques like reviews, inspections and checkpoints	0.3
	Recoverability	Restoration	Activities restoration is required when an abnormal event occurs. It investigates activities and their status of recorded and unrecorded.	0.8
		Restoration Effectiveness	It examines efficiency of restoring recorded activities.	0.8
Functionality	IT Based Functionality	IT Usage	Use of IT applications in the process activities is examined.	0.8
		IT Density	It is the ratio between documents in which IT applications are used with the total no of documents in the process.	1.0
	Interoperability	Data Exchange ability	This investigates the usage of data received from the interacted process.	No Interaction
	Security	Access Auditability	This attributes identify the person who have access to data source for audit purpose.	0.7
Usability	Understandability	Functional Understandability	In this level of staff's understanding of process activities is assessed.	1.0
	Learnability	Existence in Document	This attributes checks the presence of process activities in documents.	1.0
	Operability	Input Validity Checking	It is the examination of implementation of input parameter checking in process activities	0.6
		Undoability	In this undoability of the recorded process activities is examined after they are completed.	0.8
	Attractiveness	Attractive Interaction	Utilization of prepared documents in the process activities is examined.	0.8

4.2. Validation of the Proposed CBSD Lifecycle Effort Estimation Model

4.2.1. Conducting a Survey/ Case Study

The proposed CBSD Lifecycle Effort Estimation Model is validated by conducting an industrial survey (Attached at Appendix-A). Survey is designed on the basis of Effort Parameters/Drivers used in the proposed model. Around 48 questions were asked by the participants. Questions were arranged in CPM lifecycle phases and activities. Participants were asked to answer on the basis of their experience.

Twelve (12) participants from different organizations (public and private sectors) participated in the survey. The answers provided by the experts are then analyzed and combined percentage of accurate answer is calculated which is shown in following table:

Table 7: Activity-wise Accuracy Results of Industrial Survey

Activities	Accuracy %
Domain Engineering	85.83
Assess	80.56
Specify	87.50
Identification	86.67
Specification	81.04
Search	77.50
Select	80.28
Acquire	86.11
Tailoring	88.13
Unit Test	84.58
Component Interaction	83.33
Application Design	88.89
Adaptation	83.33
Integration Testing	76.25
Documentation / Training	90.28
Substitution	91.67
Evolution	91.67

Survey results analysis shows that the proposed model have the average accuracy between 80% - 90%. On the basis of these results we may say that the proposed model is able to estimate the effort with 80% - 90% accuracy. Better results can be achieved by repeatedly using the model during the project lifecycle because as we proceed into the project more accurate estimate is available.

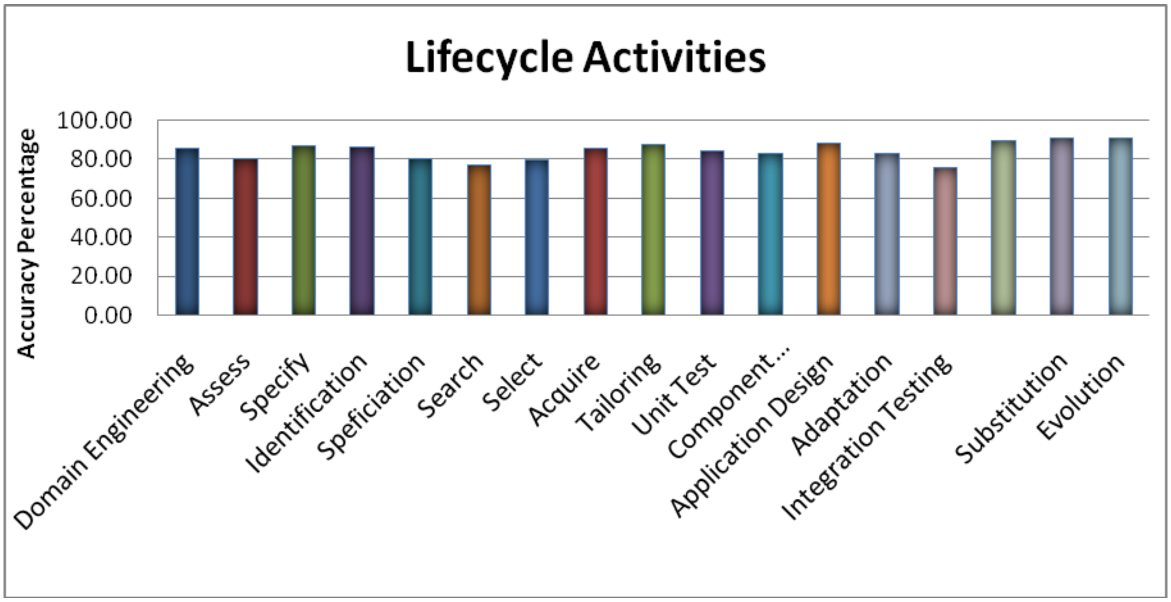

Figure 9 : Lifecycle Activities Accuracy Graph of Industrial Survey

It revealed while conducting survey that some questions asked in survey might not be interpreted as author desires by the participants; otherwise results may be more accurate. It was likely because the author was not present with participants to attain the purposeful results.

Table 8 : Phase-wise Accuracy Results of Industrial Survey

Phase	Accuracy %
Domain Engineering	85.83
Requirement Analysis	84.52
Component Analysis	82.92
Component Provision	83.99
Architectural Design	85.42
Integration	80.50
Deployment	90.28
Maintenance	91.67

For instance, Question No. 1: what effort (low, medium, high) would be required if No. of available domain applications would be (many, normal, few)? Some experts may consider that 'No. of available domain applications' effort parameters is in-directly proportional to Effort because if many domain applications are available then availability of the component will be high thus effort required will be very low.

On the other hand, some experts may be of the view that 'No. of available domain application' effort parameter is directly proportional to the effort because if many domain applications are available then effort required in Domain Engineering phase would be high. Author modeled, second view in the proposed model because the question was asked specific to the Domain Engineering Phase/activity.

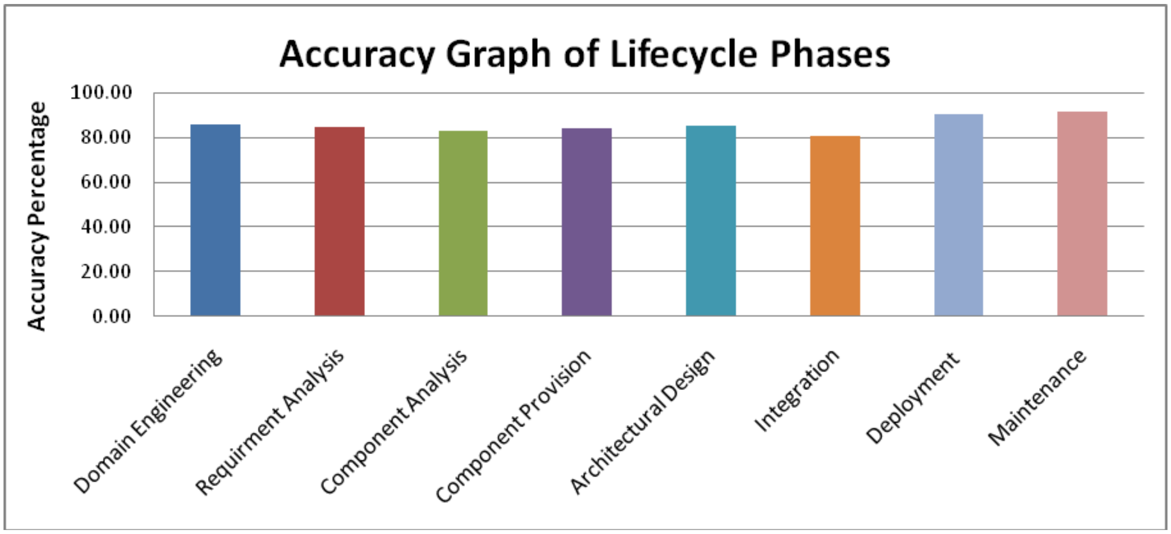

Figure 10 : Lifecycle Phases Accuracy Graph of Industrial Survey

4.2.2. By Measuring Specificity and Sensitivity

Measurement of the survey answer is subject to random variation. Because when same question answered multiple times by multiple participants the answer may vary. This variation might be due to variation in the question understanding or in the participants. Therefore, it is necessary to measure the surveys answers as precisely as possible in order to validate the proposed effort estimation model. Random variation is indirectly proportion to the precision of the measurement. It means that if random variation decreases then precision of the measurement will increase. Thus, Specificity and Sensitivity measurement is used to decrease the variation in survey answers.

Table 9 : Specificity and Sensitivity Results

Phase	Specificity	Sensitivity
Domain Engineering	#DIV/0!	1
Requirement Analysis	0.5	0
Component Analysis	0.5	#DIV/0!
Component Provision	0.6666667	0
Architectural Design	1	0
Integration	0.5	0
Deployment	1	#DIV/0!
Maintenance	1	#DIV/0!
Complete Lifecycle	0.962963	0.6315789

Specificity and Sensitivity was measured by analyzing the right/wrong answers and clear/ambiguous questions. Here right/wrong answers means the accuracy of answers of the survey participants, while clear/ ambiguous means the question which may be interpreted as clear or ambiguous by the participants. Example of Clear/ambiguous question is given in section 2.4.1.

Detail Calculation of the Specificity and Sensitivity is shown in Appendix- C while results and result graph are shown in Table 9 and Figure 11, respectively.

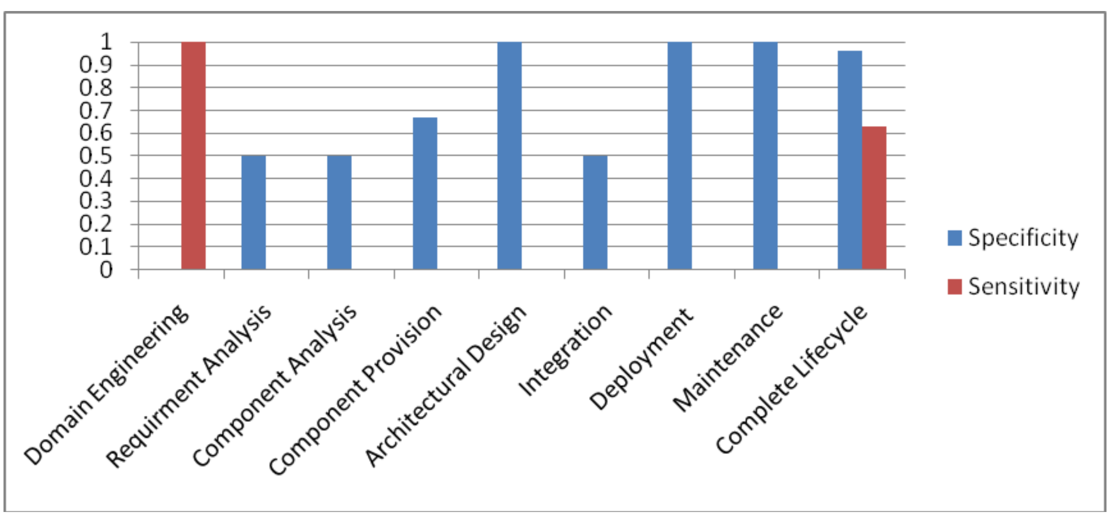

Figure 11 : Specificity and Sensitivity Graphs

CHAPTER 5

CONCLUSION AND FUTURE WORK

This chapter comprises of two sections: the first section discusses the concluding notes of the presented work while the second section discusses the recommendations for future work.

5.1. Conclusion

Effort Estimation is considered a very crucial and difficult activity of the Project Management. Poor estimates may lead to project failure or undesirable results. Like traditional software development approach, Effort Estimation for Component Based Software Development is also a challenging activity. Literature reviewed in thesis shows that focus of the researcher in the field of CBSD effort estimation remained towards integration centric activities, while other phase(s) of lifecycle remained unaddressed.

The work presented in this thesis is the first step towards estimating complete lifecycle effort of Component Based Software Development. For the purpose, Fuzzy Logic approach is used. It was ensured that each aspect of CBSD lifecycle must be covered thus Circular Lifecycle Process Model of CBSD is proposed. This model is also validated to ensure that accurate estimates can be achieved. For each activity of the proposed Circular Process Model, effort parameters were identified. These effort parameters are the factors which directly or indirectly affect the effort. Each effort parameters is fuzzified using membership function. An enriched Fuzzy rule base was prepared to provide maximum input coverage and precise estimation. This effort estimation model is also validated by conducting an industrial survey and then by measuring specificity and sensitivity of the survey results.

In this thesis following objectives were achieved:

- A comprehensive CBSD lifecycle process model is proposed.
- The CPM model is validated using Process Quality Measurement Model (PQMM) [19] and by comparing with process model of Hazleen Aris et al [13].

- A complete lifecycle effort estimation model for CBSD is also proposed which is a first step towards estimating CBSD lifecycle effort.
- Proposed Estimation model is also validated by conducting a industrial survey.

5.2. Future Work

Avenues towards perfection remains always open. Following are a few suggestions to extend or improve this work:

- Proposed CBSD Life-Cycle Effort Estimation model presently has 64 effort parameters which may be enriched to achieve more specific results.
- This model is formulated using Fuzzy Logic; which can be optimized for more accurate results.

REFERENCES

[1] Cecilia Albert and Lisa Brownsword, Evolutionary Process for Integrating COTS-Based Systems (EPIC): An overview, Technical Report CMU/SEI-2002-TR-009 ESC-TR-2002-009, July, 2002.

[2] K. Kaur and H. Singh. Candidate process models for component based software development. Journal of Software Engineering, 4(1):16–29, 2010.

[3] Ian Sommervilee, Software Engineering, 7th Edition, Pearson Education.

[4] G. Kotonya, I. Sommerville, and S. Hall. Towards a classification model for component-based software engineering research. In Proc. 29th EUROMICRO Conference, pages 43–52. IEEE Computer Society, 2003.

[5] Rajender Singh Chhillar, ParveenKajla, A New Knot Model for Component Based Software Development, International Journal of Computer Science Issues Vol: 8 Issue: 3 Pp.: 480-484, 2011.

[6] Gill N. S. and Tomar P., “X Model: A New Component- Based Model”, MR International Journal of Engineering and Technology, Vol. 1, No. 1 & 2, pp. 1-9, , 2008

[7] Luiz Fernando Capretz, " Y: A new Component-Based Software Life Cycle Model ", Journals of Computer Science1 (1) : pp.76-82, 2005.

[8] Anurag Dixit and P.C. Sexna, "Umbrella: A new Component- Based Software Development Model", International Conference on Computer Engineering and Applications IPCSIT, Singapore, vol.2, 2011.

[9] Kuljit Kaur et al, "Towards a suitable and systematic approach for Component Based Software Engineering", World Academy of Science, Engineering and Technology, 27, 2007.

[10] Ivica Crnkovic, component Based Development Process and Component Life Cycle, 27th International Conference on I.T Interfaces, IEEE, Caretat, Croatia, 2005

[11] Lata Nautiyal et al, "Elite: A New Component-Based Software Development Model", Int. J. Computer Technology & Applications, Vol 3 (1), 119-124, JAN-FEB, 2012.

[12] Kung-Kiu Lau et al, “The W Model for Component-based Software Development”, EUROMICRO-SEAA 2011: 47-50.

[13] Hazleen Aris and Siti Salwah Salim, “The Development of a Simplified Process Model for CBSD”, The International Arab Journal of Information Technology, Vol. 4, No. 2, April 2007.

[14] M. Morisio et al, ”COTS-based software development: Processes and open issues” The Journal of Systems and Software 61, 189–199, 2002.

[15] Chris Abts, M.S. et al, “COCOTS: A COTS Software Integration Lifecycle Cost Model - Model Overview and Preliminary Data Collection Findings”, USC Center for Software Engineering, 2000.

[16] AnasBassam AL-Badareen,et al, “Reusable Software Component Life Cycle”, International Journal of Computers, Issue 2, Volume 5, 2011.

[17] Ehsan Kouroshfar et al, “Process Patterns for Component-Based Software Development”, G.A. Lewis, I. Poernomo, and C. Hofmeister (Eds.): CBSE 2009, LNCS 5582, pp. 54–68, 2009.

[18] Jason H. Sharp and Sherry D. Ryan, “Component-Based Software Development: Life Cycles and Design Science-Based Recommendations”, Proc CONISAR, v2 (Washington DC), 2009

[19] A.Selcuk Guceglioglu et al, “The Application of a New Process Quality Measurement Model for Software Process Improvement Initiatives ", IEEE 11th International Conference on Quality Software, 2011.

[20] IEEE 1517, Standard for Information Technology—Software Life Cycle Processes—Reuse Processes, IEEE, Piscataway, N.J., 1999.

[21] http:// www.sourceforge.net.

[22] International Electro-technical Commission (IEC), Technical Committee No. 65: Industrial Process Measurement and Control, Sub-Committee 65 B: Devices, IEC 1131 – Programmable Controllers, Part 7 - Fuzzy Control Programming, Committee Draft CD 1.0 (Rel. 19 Jan 97)

[23] T. Wijayasiriwardhane, R. Lai, K.C. Kang, “Effort estimation of component-based software development – a survey”, The Institution of Engineering and Technology, IET Software, 2011, Vol. 5, Iss. 2, pp. 216–228

[24] T.N.Sharma, “Analysis of Software Cost Estimation using COCOMO II”, International Journal of Scientific & Engineering Research Volume 2, Issue 6, ISSN 2229-551, June-2011.

[25] Khaled Hamdan et al, “The Influence of Culture and Leadership on Cost Estimation”, UAE University, Al Ain, UAE and University of Sunderland, Sunderland, UK.

[26] Karpowich, M., Sanders, T., Verge, R.: ‘An economic analysis model for determining the custom versus commercial software tradeoffs,’ in Gulledge, T.R., Hutzler, W.P. (Eds):

‘Analytical methods in software engineering economics’, (Springer-Verlag, 1993), pp. 237–252.

[27] Abts, C., Boehm, B.W.: ‘COTS software integration cost modeling study’ (Centre for Systems and Software Engineering, University of Southern California), http://sunset.usc.edu/csse/TECHRPTS/1998/usccse98-520/usccse98-520.pdf, accessed August 2008

[28] Abts, C., Boehm, B.W., Clark, E.B.: ‘COCOTS: a COTS software integration lifecycle cost model – model overview and preliminary data collection findings’. Proc. 11th European Software Control and Metrics Conf. and Third Software Certification Programme in Europe, (ESCOM – SCOPE 2000), Munich, Germany, 2000, pp. 325–333

[29] Abts, C.: ‘Extending the COCOMO II software cost model to estimate effort and schedule for software systems using commercial-off-theshelf (COTS) software components: the COCOTS model’. PhD thesis, University of Southern California, 2004

[30] Minkiewicz, A.F.: ‘Are software COTS solutions an affordable alternative’. Proc. Aerospace Conf., Piscataway, NJ, March 2004, pp. 4073–4082

[31] Mahmood, S., Lai, R., Kim, Y.S., Kim, J.H., Park, S.C., Oh, H.S.: ‘A survey of component based system quality assurance and assessment’, Inf. Softw. Technol., 2005, 47, (10), pp. 693–707.

APPENDIX – A: Survey

Survey–EFFORT ESTIMATION IN COMPONENT BASED SOFTWARE DEVELOPMENT

Introduction:

This survey is being carried out to acquire the expert opinion, regarding the effort estimation in Component Based Software Development (CBSD). The information gathered in this survey will help in validation of the CBSD Effort Estimation model. We'd like to know participant's experience regarding effort estimation in CBSD. Privacy and confidentiality of the participant will be respected and taken seriously. It would take about 30 minutes.

Guidelines for filling the Survey:

- Questions are categorized in CBSD lifecycle phases/activities.
- Question must be answered in context to their phase/activity, mentioned in the survey.
- There is no right or wrong answer. Just answer on the basis of experience.
- One question may have multiple answers.
- To answer, tick (√) the appropriate box.

If you have any questions or concerns, please feel free to contact:-
Jahanzaib Khan,
NUST College of E&ME, Rawalpindi.
+ 092 – 0314 – 2096 931 or JzebKhanzada@yahoo.com

Participants Information

Personal Information	
Name:	
Designation:	
Qualification:	
Experience(in years)	
Contact No:	
Organization Information	
Organization Name	
No. of Employees	
Type(Public/Private)	

Organization Age					
S#	Questions		Effort Required		
			High	Medium	Low
Phase-I: **Domain Engineering**					
1	What Effort would be required if **Number Of Available Domain Applications** would be:				
		Many			
		Normal			
		Few			
Phase-II: **Requirement Analysis (Sub – *Activity-I : Requirement Assessment*)**					
2	What Effort would be required if number of **Requirement Sources** would be:				
		Single			
		Multiple			
3	What Effort would be required if **Organizational Diversity(**functional, hierarchical etc) would be:				
		High			
		Medium			
		Low			
4	What Effort would be required if **End-User Diversity** would be:				
		High			
		Medium			
		Low			
Phase-II: **Requirement Analysis (Sub – *Activity-II : Requirement Specification*)**					
5	What Effort would be required if System's N**umber of Functional Requirements** would be:				
		Too Many			
		Average			
		Too Few			
6	What Effort would be required if System's **Number of Non- Functional Requirement** would be:				
		Too Many			
		Average			
		Too Few			
7	What Effort would be required if System's **Number of Constraints** would be:				
		Many			
		Average			
		Few			
8	What Effort would be required if the System Requirements are:				
		Lucid(Clear)			
		Obscure(Un-Clear)			
Phase-III: **Component Analysis (Sub – *Activity-I : Component Identification*)**					
9	What Effort would be required if System's **Number of Functional Requirement** would be:				
		Too Many			
		Average			
		Too Few			
10	What would be the Effort required if System's **Number of Non- Functional Requirement** would be:				
		Too Many			
		Average			
		Too Few			
Phase-III: **Component Analysis (Sub – *Activity- II : Component Specification*)**					
11	What would be the Effort required if **Number of Identified Components** from requirements are:				
		Many			
		Average			

	Few			
12	What Effort would be required if **Number of Identified Interfaces** of identified components would be:			
	Many			
	Average			
	Few			
13	What would be the Effort required if **Number of Identified Membership Functions** would be:			
	Many			
	Average			
	Few			
14	What would be the Effort required if Identified **Component's Cohesion** is:			
	Minimum			
	Maximum			
Phase-IV: **Component Provision (Sub – *Activity- I : Component Search)***				
15	What would be the Effort required if **Size of the repository** used for component searching is :			
	Large			
	Medium			
	Small			
16	What would be the Effort required if **Strategy used for the component search** is :			
	Top-Down			
	Bottom-Up			
Phase-IV: **Component Provision (Sub – *Activity- II : Component Select)***				
17	What would be the Effort required if Component's **number of Functional Requirement** would be:			
	Too Many			
	Average			
	Too Few			
18	What would be the Effort required if Component's **number of Non- Functional Requirement** would be:			
	Too Many			
	Average			
	Too Few			
19	What would be the Effort required if **number of available domain applications** would be:			
	Many			
	Normal			
	Few			
Phase-IV: **Component Provision (Sub – *Activity- III : Component Acquire) (Optional)***				
20	What would be the Effort required if **Training/Documentation** provided by Component's Supplier is::			
	Satisfactory			
	Unsatisfactory			
21	What would be the Effort required if **Product Extension Willingness** of Component Supplier is:			
	High			
	Moderate			
	Low			
22	What would be the Effort required if **Support** of component's Supplier is:			
	Available			
	Un-Available			
Phase-IV: **Component Provision (Sub – *Activity- IV : Component Tailoring)***				
23	What would be the Effort required if component's **number of parameter to be specified** are:			
	High			
	Normal			
	Low			

24	What would be the Effort required if **number of scripts** required for the components are:			
	High			
	Normal			
	Low			
25	What would be the Effort required if required **number of reports/GUI screen** for the components are:			
	High			
	Normal			
	Low			
26	What Effort would be required if **number of security levels/user profiles** needed for components are:			
	High			
	Normal			
	Low			
Phase-IV: **Component Provision (Sub – *Activity- V : Unit Testing)***				
27	What would be the Effort required if **Testing Methodology** used for component's unit testing is:			
	White Box			
	Grey Box			
	Black Box			
28	What would be the Effort required if **Success Criteria** of the testing is:			
	Error_free			
	With Acceptable_Errors			
Phase-V: **Architectural Engineering (*Sub – Activity- I : Component Interaction)***				
29	What would be the Effort required if **number of Components prepared** in phase-IV are:			
	Many			
	Normal			
	Few			
30	What would be the Effort required if **architectural mismatch** among prepared components is :			
	High			
	Average			
	Low			
31	What Effort would be required if Component's **number of interfaces or membership functions** are :			
	Many			
	Normal			
	Few			
32	What would be the Effort required if Component's **Interface Complexity** is :			
	High			
	Average			
	Low			
33	What would be the Effort required if **Coupling** among components is :			
	Minimum			
	Maximum			
Phase-V: **Architectural Engineering (*Sub – Activity- II : Application Design)***				
34	What would be the Effort required if **Requirement Flexibility** in the project is :			
	Allowed			
	Not Allowed			
35	What would be the Effort required if **Schedule Flexibility** for the project is :			
	Allowed			
	Not Allowed			

36	What would be the Effort required if **Resource Availability** for the project is :			
	Ample			
	Adequate			
	Scanty			
Phase-VI: **Integration (*Sub – Activity- I : Adaptation)***				
37	What would be the Effort required if **number of Function Points** are :			
	Many			
	Normal			
	Few			
38	What would be the Effort required if Components **number of interfaces** or membership functions are :			
	Many			
	Normal			
	Few			
39	What would be the Effort required if **number of architectural constraints of the component** are :			
	Many			
	Normal			
	Few			
Phase-VI: **Integration (*Sub – Activity- II : Integration Testing)***				
40	What would be the Effort required if **Testing Methodology** used for integration testing is:			
	White Box			
	Grey Box			
	Black Box			
41	What would be the Effort required if **Success Criteria** of integration testing is:			
	Error_free			
	With Acceptable_Errors			
Phase-VII: **Deployment (*Sub – Activity- I : Documentation / User Training)***				
42	What would be the Effort required if required **User Manual/ Documentation** is :			
	Concise			
	Comprehensive			
43	What would be the Effort required if **number of sites** , the system to be deployed are:			
	Many			
	Average			
	Few			
44	What would be the Effort required if **targeted End-use**r is:			
	Technical			
	Non-Technical			
Phase-VIII: **Maintenance (*Sub – Activity- I : Substitution)***				
45	What would be the Effort required if **number of components to be replaced** are:			
	Much			
	Average			
	Few			
Phase-VIII: **Maintenance (*Sub – Activity-I I :Evolution)***				
46	What would be the Effort required if **size of change** is are:			
	High			
	Medium			
	Low			

Just two questions phase specific.

Requirement Analysis				
47	What would be the Effort required interest of End-User in requirement analysis phase is:			
	High			
	Medium			
	Low			
Component Analysis				
48	What would be the Effort required if **Reuse Type** considered is:			
	WhiteBox			
	GreyBox			
	BlackBox			

Thanks for your time and sharing your opinion.

APPENDIX – B: Linguistic Variables

S#	Effort Parameters	Effort		
		Low	Medium	High
Domain Engineering				
1.	NOADA - No. of available Domain Applications	Few	Normal	Many
Requirement Analysis				
2.	NORS - No. of Requirement Sources	Single		Multiple
3.	OD - Organizational Diversity	Low	Medium	High
4.	UD - User Diversity	Low	Medium	High
5.	NOFR - No. of FRs	Too Few	Average	Too Many
6.	NONFR - No. of NFRs	Too Few	Average	Too Many
7.	NOC - No. of Constraints	Few	Average	Many
8.	RC - Requirement Clarity	Lucid	-	Obscure
9.	**LOEUI - Level of End-User Interest**	Willing	-	Un-Willing
Component Analysis				
10.	NOFR - No. of FRs	Too Few	Average	Too Many
11.	NONFR - No. of NFRs	Too Few	Average	Too Many
12.	NOIC - No. of Identified Components	Few	Average	Many
13.	NOII - No. of Identified Interfaces	Few	Average	Many
14.	NOIMF - No. of Identified Member functions	Few	Average	Many
15.	COH - Cohesion	Maximum	-	Minimum
16.	**RT - Reuse Type**	Blackbox	Greybox	Whitebox
Component Provision				
17.	RS - Repository Size	Small	Medium	Large
18.	SS - Search Strategy	Top Down	-	Bottom Up
19.	NOFR - No. of FRs	Too Few	Average	Too Many
20.	NONFR - No. of NFRs	Too Few	Average	Too Many
21.	NOADA - No. of available domain applications	Few	Normal	Many
22.	ACPTD - COTS Supplier Provided Training and Documentation[15]	Satisfactory	-	Unsatisfactory
23.	ACSEW - COTS Supplier Product Extension Willingness [15]	High	Moderate	Low
24.	ACPPS - COTS Supplier Product Support [15]	Available	-	Unavailable
25.	NOPTBS - No. of Parameters to be Specified [15]	Low	Normal	High
26.	IGS - input/GUI screen [15]	Low	Normal	High
27.	ORL - output report layout [15]	Low	Normal	High
28.	SPS - security protocols set-up [15]	Low	Normal	High
29.	TM - Testing Methodology	Blackbox	Greybox	Whitebox
30.	SC - Success Criteria	Acceptable Errors	-	Error Free

31.	**NOIC - No. of Identified Components**	Few	Average	Many
Architectural Design				
32.	NOCF - No. of Components Fashioned	Few	Normal	Many
33.	CAM - Components Architectural mismatch	Low	Average	High
34.	NOIAMF - No of Interfaces and Membership Functions	Few	Normal	Many
35.	IC - Interface Complexity	Low	Average	High
36.	Cou –Coupling	Minimum	-	Maximum
37.	RF - Requirements Flexibility	Allowed	-	Not-Allowed
38.	SF - Schedule Flexibility	Allowed	-	Not-Allowed
39.	RA - Resources Availability	Ample	Adequate	Scanty
Integration				
40.	FP - Function Points	Few	Normal	Many
41.	NOIAMF - No of Interfaces and Membership Functions	Few	Normal	Many
42.	AC - Architectural Constraints	Few	Normal	Many
43.	TM - Testing Methodology	Blackbox	Greybox	Whitebox
44.	SC - Success criteria	Acceptable Errors	-	Error Free
45.	**NOCF - No. of Components Fashioned**	Few	Average	Many
Deployment				
46.	NOSTBD - No of Sites to be Deployed	Few	Average	Many
47.	TE - Targeted End-user	Technical	-	Non-Technical
48.	UMDC - User Manual/ Documentation Comprehensiveness	Concise	-	Comprehensive
Maintenance				
49.	NOCTBR - No. of Components to be replaced	Few	Average	Medium
50.	SOC - Size of Change	Low	Medium	High
Cross-Cutting Parameters				
51.	OC - Organization Culture [25]	Good	So So	Bad
52.	PM - Process Maturity[24]	Mature	-	Immature
53.	LS - Leadership Skills[25]	Adroit	Intermediate	Novice
54.	TC - Team Cohesion [24]	High	Medium	Low
55.	SC - Stakeholder Cohesion	High	Medium	Low
56.	TSK - Team Skills	Adroit	Intermediate	Novice
57.	TE - Team Experience	Vast	Sufficient	Beginner
58.	TSZ - Team Size	Large	Medium	Small
59.	TC - Team Consistency	Low	Medium	High
60.	PS - Project Size	Large	Medium	Small
61.	PC - Project Complexity	Much	Average	Less
62.	PP - Project Precedence [24]	High	Medium	Low
63.	UOST - Use of Standard Tools	Yes	-	No
64.	RW – Rework	Extensive	-	Slight

APPENDIX – C: Specificity and Sensitivity Calculations

Phase	Clear Question	Ambiguous Question	Total	Specificity	Sensitivity
Domain Engineering					
Right Answer	0	1	1	#DIV/0!	1
Wrong Answer	0	0	0		
Total	0	1	1		
Requirement Analysis					
Right Answer	3	0	3	0.5	0
Wrong Answer	3	1	4		
Total	6	1	7		
Component Analysis					
Right Answer	3	0	3	0.5	#DIV/0!
Wrong Answer	3	0	3		
Total	6	0	6		
Component Provision					
Right Answer	8	0	8	0.6666667	0
Wrong Answer	4	2	6		
Total	12	2	14		
Architectural Design					
Right Answer	5	0	5	1	0
Wrong Answer	0	3	3		
Total	5	3	8		
Integration					
Right Answer	2	0	2	0.5	0
Wrong Answer	2	1	3		
Total	4	1	5		
Deployment					
Right Answer	3	0	3	1	#DIV/0!
Wrong Answer	0	0	0		
Total	3	0	3		
Maintenance					
Right Answer	2	0	2	1	#DIV/0!
Wrong Answer	0	0	0		
Total	2	0	2		
Complete Lifecycle					
Right Answer	26	12	38	0.9629629	0.631578947
Wrong Answer	1	7	8		
Total	27	19	46		

APPENDIX – D: CPM Validation Matrix using PQMM.

	Domain Engineering	Requirement Assessment	Requirement Specification	Component Identification	Component Specification	Component Search	Component Selection	Component Acquire	Tailoring	Unit Test	Comp. Arch. Comprehension	Application Design	Adaptation	Integration Test	Deployment	Substitution	Evolution
Complexity	No Decision	No Decision	CBSE or Not?	No Decision	Build VS Buy?	No Decision	No Decision	No Decision	No Decision	No Decision	No Decision	No Decision	No Decision	No Decision	No Decision	No Decision	No Decision
Coupling	No Interaction	No Interaction	No Interaction	No Interaction	No Interaction	No Interaction	No Interaction	No Interaction	No Interaction	No Interaction	No Interaction	No Interaction	No Interaction	No Interaction	No Interaction	No Interaction	No Interaction
Failure Avoidance*	No review, inspection, checkpoint or similar techniques	No review, inspection, checkpoint or similar techniques	review	No review, inspection, checkpoint or similar techniques	review	No review, inspection, checkpoint or similar techniques	No review, inspection, checkpoint or similar techniques	No review, inspection, checkpoint or similar techniques	No review, inspection, checkpoint or similar techniques	testing	No review, inspection, checkpoint or similar techniques	review	No review, inspection, checkpoint or similar techniques	testing	No review, inspection, checkpoint or similar techniques	No review, inspection, checkpoint or similar techniques	No review, inspection, checkpoint or similar techniques
Restoration	Not Recorded	Not Recorded	Recorded in SRS	Not Recorded	Recorded in RCS	Recorded in RCS	Recorded in RCS	Recorded in RCS	Recorded in RCS	Recorded in TD	Recorded in RCS	Recorded in ADD	Recorded in Implementation document	Recorded in TD	Recorded in User Manual	Recorded in ID,RCS,TD, ADD	Recorded in SRS,ID, RCS,TD ,ADD
Restoration Effectiveness	No Restoration	No Restoration	Restored	No Restoration	Restored	Restored	Restored	Restored	Restored	Restored	Restored	Restored	Restored	Restored	Restored	Restored	Restored
Functional Adequacy	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

Functional Completeness	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
IT Usage	No IT Usage	No IT Usage	Application Software Usage	No IT Usage	Application Software Usage	Application Software Usage	Application Software Usage	Application Software Usage	Application Software Usage	Application Software Usage	Application Software Usage	Application Software Usage	Application Software Usage	Application Software Usage	Application Software Usage	Application Software Usage	Application Software Usage
IT Density	No Document	No Document	SRS	No Document	RCS	RCS	RCS	RCS	RCS	TD	RCS	ADD	Implementation Document	TD	User Manual	RCS,TD, ADD	SRS,RCS, TD,ADD
Computational Accuracy	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Data Exchange ability	No Interaction	No Interaction	No Interaction	No Interaction	No Interaction	No Interaction	No Interaction	No Interaction	No Interaction	No Interaction	No Interaction	No Interaction	No Interaction	No Interaction	No Interaction	No Interaction	No Interaction
Access Auditability	Domain Expert	No Interaction with Data Source	No Interaction with Data Source	No Interaction with Data Source	No Interaction with Data Source	Developer	No Interaction with Data Source	No Interaction with Data Source	No Interaction with Data Source	Developer	No Interaction with Data Source	No Interaction with Data Source	No Interaction with Data Source	Developer	No Interaction with Data Source	Yes but Actor cannot be identified	Yes but Actor cannot be identified
Functional Understandability	No difficulties or misunderstandings	No difficulties or misunderstandings	No difficulties or misunderstandings	No difficulties or misunderstandings	No difficulties or misunderstandings	No difficulties or misunderstandings	No difficulties or misunderstandings	No difficulties or misunderstandings	No difficulties or misunderstandings	No difficulties or misunderstandings	No difficulties or misunderstandings	No difficulties or misunderstandings	No difficulties or misunderstandings	No difficulties or misunderstandings	No difficulties or misunderstandings	No difficulties or misunderstandings	No difficulties or misunderstandings
Existence in Documents	Described	Described	Described	Described	Described	Described	Described	Described	Described	Described	Described	Described	Described	Described	Described	Described	Described
Input Validity Checking	No	No	Yes	No	Yes	Yes	Yes	No	Yes	Yes	No	Yes	Yes	Yes	No	Yes	Yes

Undoability	Not Recorded	Not Recorded	Undoability of SRS	Not Recorded	Undoability of RCS	Undoability of RCS	Undoability of RCS	Undoability of RCS	Undoability of RCS	Undoability of TD	Undoability of RCS	Undoability of ADD	Undoability of Implementation document	Undoability of TD	Undoability of User Manual	Undoability of RCS,TD,ADD	Undoability of SRS,RCS, TD,ADD
Attractive Interaction	No Attractive Interaction	No Attractive Interaction	Attractive Interaction	No Attractive Interaction	Attractive Interaction	Attractive Interaction	Attractive Interaction	Attractive Interaction	Attractive Interaction	Attractive Interaction	Attractive Interaction	Attractive Interaction	Attractive Interaction	Attractive Interaction	Attractive Interaction	Attractive Interaction	Attractive Interaction

APPENDIX – E: CPM Validation Calculations using PQMM.

	Results	A	B	Formula	Description [19]
Complexity	0.9	2	17	X=1-A/B	A = Number of decisions B = Number of activities
Coupling	1.0	0	17	X=1-A/B	A = Number of interactions B = Number of activities
Failure Avoidance	0.3	5	17	X=A/B	A = Number of activities in which review, inspection, checkpoint or similar techniques are applied B = Number of activities
Restoration	0.8	14	17	X=A/B	A = Number of activities which are recorded on paper or computerized environment B = Number of activities
Restoration Effectiveness	0.8	14	17	X=A/B	A = Number of activities which can be restored B = Number of total activities
IT Usage	0.8	14	17	X=A/B	A = Number of activities in which IT applications are used for preparation, deletion, updating or searching purposes B = Number of activities
IT Density	1.0	6	6	X=A/B	A = Number of forms, reports, archival records or similar other documents that are prepared, updated, deleted or searched by using IT applications B = Number of forms, documents, archival records or similar other documents in the process
Data Exchangeability	0.0	0	0	X=A/B	A = Number of activities in which no change is performed on the received data before using it (using the data as it has been transferred) B = Number of activities which have interactions with other processes
Access Auditability	0.7	4	6	X=A/B	A = Number of activities which have access to the data and this access can be audited with its actor B = Number of activities which have accesses to the data sources
Functional Understandability	1.0	17	17	X=A/B	A = Number of activities in which staff do not encounter difficulties in understanding the tasks to be performed, B = Number of process activities
Existence in Documents	1.0	17	17	X=A/B	A = Number of activities which are described in the available documents, B = Number of activities
Input Validity Checking	0.6	11	17	X=A/B	A = Number of activities in which validity checking can be performed for input parameters B = Number of activities
Undoability	0.8	14	17	X=A/B	A=Number of activities which can be undone, B= Number of total activities
Attractive Interaction	0.8	14	17	X=A/B	A = Number of activities in which staff can prepare, delete or update forms, reports, archival records or similar other documents with no difficulties

					B = Number of total activities