Lifecycle Effort Estimation for Component Based Software Engineering: A Model and its Soft Computing Based Implementation

Jahanzaib Khan [1], Aasia Khanum [2]

[1, 2] College of Electrical & Mechanical Engineering, National University of Sciences and Technology (NUST), Islamabad, Pakistan
aasia@ceme.nust.edu.pk

Abstract: Research in Effort Estimation for Component Based Software Engineering (CBSE) has so far focused on component selection and integration effort. However, in order to get a fair idea of feasibility of a project full lifecycle effort must be taken into account. This paper presents a lifecycle based effort estimation approach for CBSE. A new CBSE lifecycle model called Circular Process Model (CPM) is presented with the novel idea of rejuvenation of one lifecycle phase during the course of later phases, something which is inevitable in CBSE. Quality assessment of CPM in light of the PQMM process measurement model reveals essential merit of the process. Subsequently, CPM is used as a framework for introducing enhanced effort parameters that are employed by an original soft computing approach to estimate lifecycle effort of a CBSE project. Validation with industrial data shows high accuracy of the proposed system.

[Khan J, Khanum A. **Lifecycle Effort Estimation for Component Based Software Engineering: A Model and its Soft Computing Based Implementation.** *Life Sci J* 2014;11(5):145-155]. (ISSN:1097-8135). http://www.lifesciencesite.com. 20

Keywords: Component Base Software Engineering (CBSE), Project Management, Software Process Model, Effort Estimation, Soft Computing.

1. Introduction

Trend to adopt Component Based Software Engineering (CBSE) is on the rise. In this approach, software is engineered by assembling *components* [20], which essentially are stand-alone modules created for easy re-use beyond the original application. CBSE offers several advantages over traditional software development approaches; including flexibility in development, fast time-to-market, better quality of software, and cheaper cost of the product.

CBSE differs from traditional software development approaches not only in terms of advantages and challenges, but also with respect to process lifecycle activities. Whereas in traditional software engineering lifecycle the normal activities include requirements specification, system design, implementation, validation and maintenance, CBSE extends these activities with component search, selection, adaptation and integration.

Lifecycle process is the course of activities that produces a new software product and continues from its inception to maintenance. Software lifecycle is a loosely defined concept [8] and in case of CBSE there is no universally agreed upon lifecycle process model. Although several attempts have been made to define an effective process model for CBSE, still all the proposed models have their own tradeoffs. Even the IEEE Standard 1517 [20] which deals with software reuse process does not enforce single lifecycle to follow, rather it just tells a minimum set of attributes that a software lifecycle must have.

Management of Information Technology or Software Engineering projects is a unique challenge []. Effort estimation is an important concern in software project management. Many techniques have been proposed over the past two or three decades for effort estimation in traditional software development domain. Since CBSE is fundamentally different from traditional development approaches, effort estimation becomes a challenging problem in this framework. So far, majority of the approaches to CBSE effort estimation have focused on selection and integration related activities of the process. However, for fair analysis of cost-effectiveness of CBSE approach, it is necessary that a comprehensive lifecycle approach taking into account all activities of the process be adopted [21].

This paper is a two-fold contribution to CBSE. First, a new process model called the Circular Process Model (CPM) is proposed with the novel provision of rejuvenation of one lifecycle phase during later phases, something which is inevitable in CBSE. Second, a soft computing framework is presented for effort estimation in CBSE. The proposed framework employs enhanced effort parameters derived from CPM to activate a Fuzzy Rule Based System (FRBS) for estimating required effort of a component based software engineering project. We present validation of the proposed process model as well as the effort estimation framework.

Rest of this paper is organized as follows. Section 2 presents literature review of process models for CBSE and effort estimation of CBSE. Section 3 describes the proposed CPM process model and the fuzzy effort estimation framework. Section 4 presents experimental validation of CPM and effort estimation framework. Finally, section 5 concludes the paper.

2. Background

2.1. CBSE Process Models

One of the earliest process models for CBSE is Evolutionary Process for Integrating COTS -Based Systems (EPIC) [1] which is adapted from traditional Rational Unified Process (RUP) [30]. EPIC rewrites managerial, engineering and acquisition activities to control COTS market in better way. It is a risk-based spiral approach in which process phases are same as those in RUP.

Process model of Qureshi and Hussain [28] 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, the exact stage when components will be added to the repository is also unclear.

Sommerville proposed sequential lifecycle process model [3] in which component search phase is included before the design phase. If any discrepancies are encountered during design, modification of requirements will be carried out. In this fashion, requirements and design are based on the components actually in hand.

Lau et al proposed the W-Model [12] which is mainly focused on Verification and Validation (V&V) in Component Development and Component Based Software Development. They argue that V&V is necessary in both lifecycles i.e. Component Development lifecycle and Component Based Software Development lifecycle. In this model repository feature and maintenance phase are not included.

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

Kotonya et al proposed Classification Lifecycle Model for CBSE [4], in which the center of attention is CBSEnet knowledge Base. In this work, both short and long term objectives of the process are acknowledged, but the model addresses only short term objectives.

M. Morisio et al [14] proposed COTS lifecycle model in which emphasis is placed on the involvement of vendor throughout the lifecycle. New activities and roles related to vendor are also identified. The model is limited in the sense that it only focuses on development phase, whereas important phases like the maintenance phase are missing.

Kouroshfar et al [17] proposed Component Based Software Development Process (CBSDP). It is a generic process derived by reviewing seven CBSE based methodologies like FORM, RUP, and CORBA etc. One inherent limitation in the process is that due to its generic nature none of the activities is mandatory; thus, rendering it difficult to implement.

MyCL Process Model was proposed by Aris et al [13]. It is an attempt to make the lifecycle process very simple; however in doing so several phases have lost necessary detail. Furthermore, requirements and architecture are frozen before component availability which makes the process very rigid and unrealistic. Unit testing is also eliminated in this model.

Bassam et al [16] in their research focused on reusability and proposed two lifecycle processes; 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 independent of the lifecycle, which we argue, should be treated as part of the lifecycle. Another limitation of the model is that it deals only with in-house development.

The Knot Model [5] was proposed by Chhillar et al. In each phase of this model risk analysis and feedback is performed 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 Base Software System (CBSS) is also maintained in pool for later re-utilization.

The Umbrella Model for CBSE [8] was proposed by Dixit et al. This model mainly revolves around testing or verification. The 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.

The Y Model [7] proposed by Capretz supports iteration and overlapping. It allows both top-down and bottom-up approach of CBSE software development. However, definition of component model is overlooked by this model.

The V Model for CBSE [10] proposed by Crnkovic et al. is an adaptation of conventional V Model. This model also focuses on verification and validation. However, important steps like Domain Engineering and System Deployment are missing from the model.

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

The X Model [6] proposed by Gill et al focuses on software reusability for large software systems. Besides being quite complex, the model ignores best practices like feedback and risk analysis

The above described models have their individual strengths and weaknesses. Some of the strong features include support for reusability, component repository, and provision for testing. However, one common limitation of all these models is that there is no broad-based reflection of complete lifetime phases encountered during real CBSE operation. As an example, deployment and maintenance phases which are indispensable for any software lifecycle process are missing in almost all the models. Moreover, the models work in a strictly sequential fashion such that one phase must be completed before starting the next one. In actual practice however there is often a need to loop back to earlier phases on basis of conditions encountered in a later phase. For instance, after component acquisition one may find that all requirements are not directly satisfied by the acquired components and may need to revert back to requirement analysis stage. This allowance of reverting back to an earlier phase is not provided by current CBSE lifecycle models.

2.2. CBSE Effort Estimation

Although a lot of work has been done on effort estimation for traditional software development approaches, not many approaches exist for CBSE effort estimation. Below, we briefly survey the most up-to-date approaches specifically designed for CBSE effort estimation:

One of the earliest models on CBSE effort estimation was developed at Science Applications International Corporation (SAIC) in early 1990s [22]. The model is mainly focused on end-user cost of integrating the components into rest of the system. As a consequence, the model misses important effort parameters like the effort expanded on component search and selection, or the effort required for component validation and testing etc.

Stutzke's model [23] was introduced by SAIC to determine additional cost of adopting a component. This additional cost is treated as a function of the component's volatility and architectural coupling with rest of the system. The model still does not consider effort of searching and selecting the desired components.

Aoyama's CBSE effort model [24] introduced detailed phases like component acquisition, compositional design, and integration. The model however suffers from unrealistic assumptions. For instance, it assumes that unit testing costs are not relevant to CBSE. In actual practice component testing may have significant implications on glue code development, particularly in case of faulty components or components with unexpected behavior.

The most well-known model for CBSE effort estimation is the Constructive Commercial off the Shelf (COCOTS) model [25]. This model is an extension of the conventional Constructive Cost Model (COCOMO) with consideration of post-development activities, most importantly maintenance. COCOTS, however, focuses on integration-centric activities and ignores other important aspects.

Adjustable Cost Model [26] enhanced COCOTS by combining its effort parameters with communication overheads, concepts from system dynamics to simulate software process. The model inherits the basic shortcoming of COCOTS in that it misses important phases of lifecycle. In fact, according to [21] while COCOTS covers 55.56% of CBSE lifecycle the adjustable model covers only 33.3%.

An analysis of the above approaches indicates that one common limitation is the lack of full lifecycle coverage in effort parameterization. Most of the models cover only integration-centric activities. Moreover, majority of the models are algorithmic in nature and require hard calculations.

3. Proposed Approach

In this section, we first present our comprehensive CBSE lifecycle model called CPM, and then present an approach for CBSE effort estimation in CPM framework.

3.1. Circular Process Model (CPM)

The proposed Circular Process Model (CPM), shown in Figure 1, is derived by embracing the strengths of the above 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 CBSE. CPM comprises eight (08) phases which are further divided into sixteen activities as shown in Table 1. In an idealized CBSE process one phase follows another in sequential fashion. Phases start from Domain Engineering and continue till Maintenance, in clockwise direction. In Idealized CBSE process no phase repeat itself as all phases execute sequentially. In reality however, one often needs to resort back to earlier phases for adaptation.

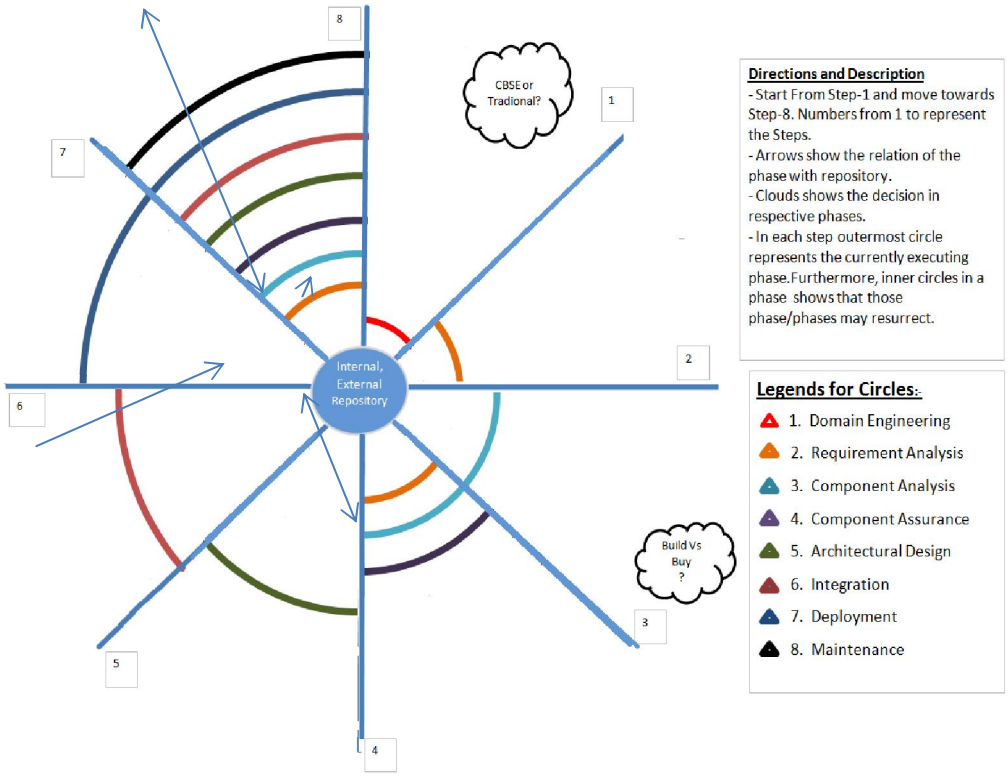

Figure 1: Proposed Circular Process Model

For instance, when required components cannot be found during Component Assurance phase the development team has to revisit the requirement analysis step to adjust the requirements. In actual CPM, 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. This rejuvenation is certain in CBSE because there are very rare chances that you may find your required components hence requirement analysis phase will have to be executed again. And if requirement analysis will be executed again then component analysis will be carried out on new requirements. Similarly, in Maintenance phase revisiting of previous phases are indispensable. Clouds in Figure 1 show the decisions to be taken in respective phases. In center, the Component Repository with labels *external* and *internal* is shown. By internal we mean in-house component repository of the software developing organization. External repository represents the open COTS (Commercial Off-the-shelf) market.

Unidirectional and bidirectional arrows in Figure 1 depict the relation of the phases with the repository. Arrow direction towards repository represents that components are being stored in internal repository as component are the assets for an organization. Arrow directions opposite to the repository show that components are analyzed or retrieved from both internal and external repository. We now discuss the phases of the proposed CPM in detail:

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

3.1.2. Requirement Analysis

In Requirement Analysis phase, software requirements are first elicited and then specified. The final outcome of this phase is Software Requirements Specification (SRS) document. This phase is not a one-time activity, especially in CBSE where it iterates again and again till the successful completion of the component assurance phase (See Figure 1).

Table 1: Phase-wise Distribution of Activities in CPM

Phases	Activities	Description	Output
Domain Engineering	Domain Engineering	Provides understanding regarding the application domain and help in taking the decision of following CBSE or Traditional approach.	Business Idea
1st Decision: CBSE or Traditional Development			
Requirement Analysis	Requirement Assessment	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	Preparing requirement specification document from the requirements finalized in requirement assessment activity.	
Component Analysis	Component Identification	Determining required components, by analyzing the SRS.	Requisite Components Specification Document
	Component Specification	Identified components are completely specified (i.e. interfaces, member functions etc.)	
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 application irrespective of integrated system [15].	
	Unit Test	Ensure component functioning in isolation	
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 along with preparation of executable copy of the product.	User Manual, Training Guide, CBSS
Maintenance	Substitution	Required if new version of COTS is available.	Component Based Software System CBSS
	Evolution	Required when new/changed requirements are demanded.	

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. Here, expert decides on the basis of his/her experience and the 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 in case of wrong decision. It is so because Component Assurance is a time consuming activity and if components are unavailable in the market then all the exercise of this phase will be futile.

3.1.4. Component Assurance

This phase is an important and distinct phase of CBSE 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. 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, the components are integrated one by one into the system. After integration of each component the system is tested to ensure smooth functioning. To accomplish the task of component integration glue code [15] is required, which works as interface between the component and the system under development.

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 increased lifetime of the product. As far as CBSE is concerned, maintenance may be required due to 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 must replace the existing ones.

3.2. Proposed Effort Estimation Framework

The proposed CBSE lifecycle Effort Estimation model is based around CPM. Effort parameters/drivers are identified for each CPM and applied in a bottom-up manner. 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. Figure 2 illustrates the framework. Below we discuss implementation of the proposed framework in detail.

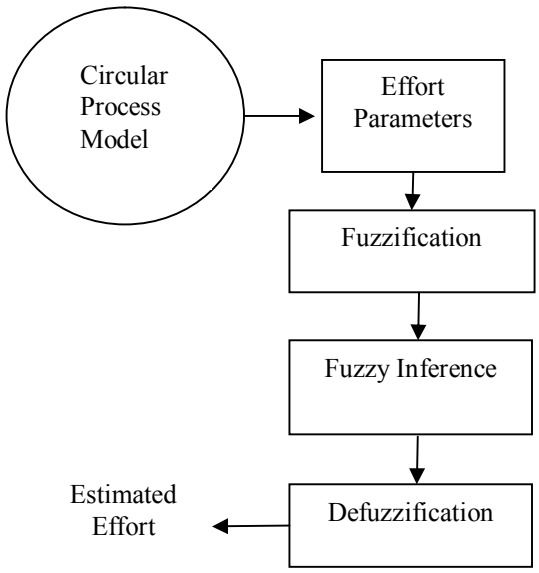

Figure 2: Proposed Effort Estimation Model

3.2.1. Effort Parameters

In the proposed framework, 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 [25], 03 parameters are taken from scale factors of COCOMO-II [29], 02 parameters are taken from [27] and rest are added according to CPM details. Complete list of effort parameters under CPM activities/phases is shown in Table 2.

3.2.2. Fuzzy Inference

For the implementation of the effort estimation we use soft computing paradigm of Fuzzy Logic because it is based on intuition and judgment and does not require rigid mathematical model. Furthermore, fuzzy sets provide smooth transition between members and non-members which makes them less sensitive to system fluctuations. Fuzzy inference is made using the following functional modules for each activity:

Table 2: Proposed Effort Parameters Using CPM

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 (Functional Requirements)		
		NONFR - No. of NFRs (Non-functional Requirements)		
		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 Design	RF - Requirements Flexibility		
		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		

3.2.3. Fuzzifier

The fuzzifier fuzzifies the values of input parameters of an activity using fuzzy sets. All input parameters are normalized to [0,1] interval before fuzzification. Gaussian membership functions have been employed for implementation of the fuzzy sets in the fuzzifier. An example of Fuzzification using three fuzzy sets is shown in Figure 3.

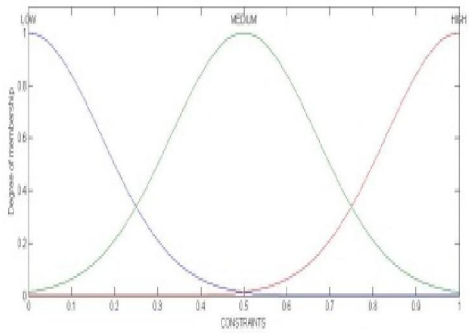

Figure 3: Fuzzification of Constraints

3.2.4. Fuzzy Rules

Fuzzy rules are simple IF-THEN relationships involving fuzzy linguistic variables as input and output. There are total 64 input variables corresponding to input parameters and one output variable for the estimated effort. These variables are distributed across 8 fuzzy inference systems corresponding to 8 phases of the CPM. Figure 4 illustrates the rule surface corresponding to effort of Requirements specification activity of requirements analysis phase of CPM.

3.2.5. Fuzzy Inference Engine

The Fuzzy Inference Engine uses the Fuzzy Rules to map current input parameters to output fuzzy effort value.

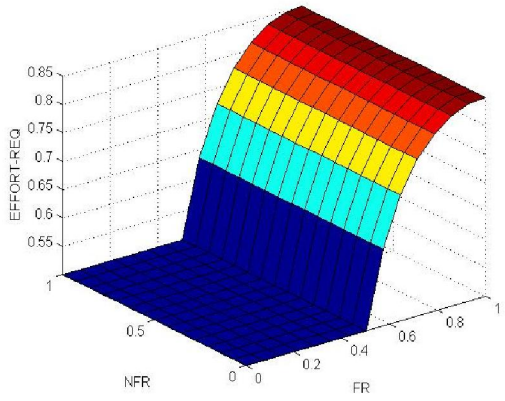

Figure 4: Rule Surface for Requirement Specification Effort

3.2.6. Defuzzifier

The defuzzifier converts fuzzy effort estimate into a crisp value in normalized interval.

4. Evaluation and Results

In this section we present an evaluation of the proposed approach including both the CPM process model and the effort estimation framework.

4.1. CPM Evaluation

CPM Evaluation is done along two aspects. First, the model is compared with an existing state-of-art model for CBSE. Second, model quality is assessed using a well-known process quality measurement approach.

4.1.1. Comparison 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 in a waterfall fashion, and later in the Component Development phase, components are adapted to comply with requirements. There is no recourse to requirement analysis phase if the requisite component could not be found. The model's only provision in such a case is development of component from scratch, which is not the essence of CBSE. This is not the case in the proposed CPM where you can build new component, or you can modify your requirements, as desired.

Secondly, in MyCL the Architectural Design phase is placed right after the Requirement Analysis Phase which is again very troublesome because when there is no component in hand it is not feasible to freeze the architecture. Second there is also no recourse to Architectural Design phase if the components assumed in architecture could not be found. This problem is resolved by CPM in which architectural design phase is placed after the Component Assurance phase.

Further, in MyCL Unit testing has been removed from the lifecycle by arguing that components are already tested. In CPM Unit testing is included because component tailoring is required which is to set the component to be used irrespective of the integrated system [13]. Thus, unit test is necessary.

4.1.2. Process Quality Assessment

We have validated the proposed Circular Process Model using Process Quality Measurement Model (PQMM) of Gucegliogluet 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 range 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. Table 3 shows the metrics (with definitions re-phrased or adapted from [19]).

Table 3: PQMM [19] Quality Attributes values for the proposed model.

Quality Attributes of PQMM	Quality Sub-Attribute	Explanation	Metric	Value
Reliability	Failure Avoidance	Number of failure avoidance techniques	X = A / B A = Number of activities in which review, inspection, checkpoint or similar techniques are applied B = Number of activities	0.2
	Restoration	Proportion of recorded activities which can be restored in case of some abnormal event	X = A / B A = Number of activities which are recorded on paper or computerized environment B = Number of activities	0.8
	Restoration Effectiveness	It examines efficiency of restoring recorded activities	X = A / B A = Number of activities which can be restored B = Number of activities	0.8
Functionality	IT Usage	Use of IT applications in the process activities	X = A / B A = Number of activities in which IT applications are used for preparation, deletion, updating or searching purposes B = Number of activities	0.8
	IT Density	It is the ratio between the number of forms, reports, archival records or similar other documents that are prepared, updated, deleted or searched by using IT applications and total number of forms, documents, archival records or similar other documents in the process.	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	1.0
	Access Auditability	This attributes identify the person for auditablity who have access to data source.	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	0.7
Usability	Functional Understandability	Understandability of process activities	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	1.0
	Existence in Documents	This attributes checks the presence of process activities in documents.	X = A / B A = Number of activities which are described in the available documents, B = Number of activities	1.0
	Input Validity Checking	Implementation of input parameter checking in process activities	X = A / B A = Number of activities in which validity checking can be performed for input parameters B = Number of activities	0.6
	Undo-ability	Undoability of the recorded process activities is examined after they are completed.	X = A / B A=Number of activities which can be undone, B= Number of activities	0.8
	Attractive Interaction	Usage and Design of prepared documents in the process activities.	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 activities	0.8

It can be seen that only failure avoidance attribute of the CPM 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 5.

We also performed a chi-square test of independence to test whether there is any difference between conventional practices of CBSE and proposed model with respect to expectations of the development team and management. A survey of 30 software developers/managers was conducted to find the number of expectations of project managers and developers that are addressed in conventional approach versus the proposed CPM. We set the null and alternate hypotheses as follows:

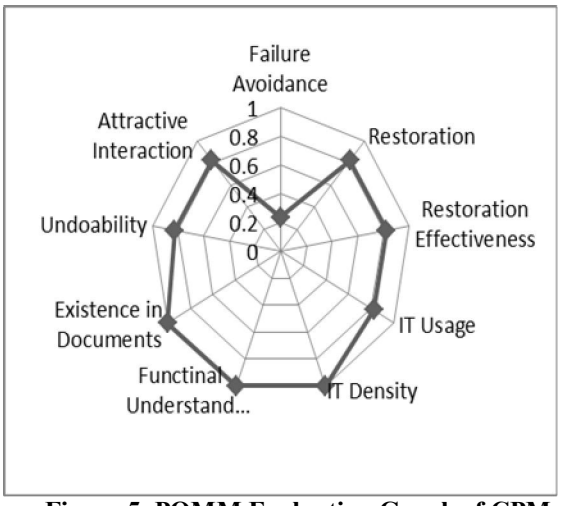

Figure 5. PQMM Evaluation Graph of CPM

Null Hypothesis (H_0): Independence (i.e. no association between process model and team satisfaction)

Alternate Hypothesis (H_a): Dependence (i.e. validated association between process model chosen and team satisfaction)

We use the chi-square ($\chi 2$) test statistic to validate the H_0. The value of this statistic is calculated as:

$$\chi 2 = \sum_{i,j}\left[\frac{(O_{ij}-E_{ij})^2}{E_{ij}}\right]$$

where O_{ij} and E_{ij} are observed and expected frequencies at index i and j respectively of a contingency table of size i×j. We have a contingency table of size 2x2 where rows represent the choice of process model and the columns represent the number of satisfied and dissatisfied survey respondents in each case. The survey results are represented by contingency table in Table 4.

Table 4: Contingency Table of Observed Frequencies (survey results)

	Satisfied	Not Satisfied	Total
Conventional	**6**	**9**	**15**
CPM	**10**	**5**	**15**
Total	**16**	**14**	**30**

The value of chi-square statistic for the above data is 2.143 with a p-value of 0.143. This leads to rejection of the null hypothesis and conclude that there is association between choice of process model (Conventional or CPM) and satisfaction of software team's requirement.

4.2. Validation of Effort Estimation

The proposed CBSD Lifecycle Effort Estimation Model is validated by conducting an industrial survey. Survey is designed on the basis of Effort Parameters/Drivers used in the proposed model. Around 48 questions were asked from 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 were then analyzed and compared with system's output. Accuracy (i.e. percentage of correct answers) of the model results is shown in Figure 6, demonstrating the viability of the approach.

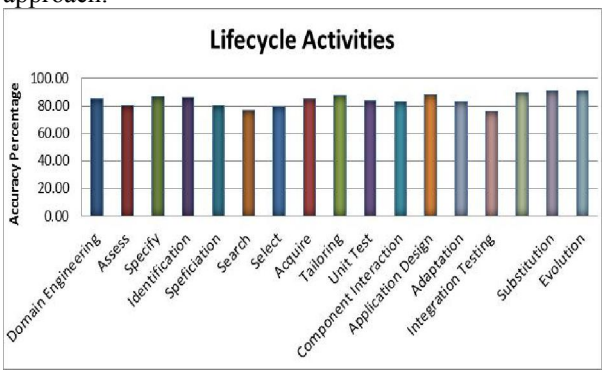

Figure 6. Lifecycle Effort Estimation Accuracy of the Proposed Approach

5. Conclusions and Future Work

In this study, Circular Process Model (CPM) for Component Based Software Engineering (CBSE) has been proposed by keeping in view the strengths and weaknesses of the process models proposed in literature. Main focus of CPM is rejuvenation of any phase during the execution of later lifecycle phases. The model has been validated using PQMM method, which shows that circular model is pragmatic and applicable in industry for achieving better results. Moreover, a fuzzy based effort estimation approach for CBSE lifecycle phases is presented, and validated with industrial survey. In future we intend to test the model thoroughly in practical industrial settings.

References

1. C. Albert and L. 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. Sommerville, 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, pp. 43–52. IEEE Computer Society, 2003.
5. R. S. Chhillar, P. Kajla, "A New Knot Model for Component Based Software Development," Int'l Journal of Computer Science Issues vol: 8 issue: 3 pp.: 480-484, 2011.
6. N. S. Gill and P. Tomar , "X Model: A New Component- Based Model", MR International Journal of Engineering and Technology, vol. 1, no. 1 - 2, pp. 1-9, 2008
7. L. F. Capretz, " Y: A new Component-Based Software Life Cycle Model ", Journal of Computer Science1 (1): pp.76-82, 2005.
8. A. Dixit and P.C. Sexena, "Umbrella: A new Component- Based Software Development Model", International Conference on Computer Engineering and Applications IPCSIT, Singapore, vol.2, 2011.
9. K. Kaur et al, "Towards a suitable and systematic approach for Component Based Software Engineering", World Academy of Science, Engineering and Technology, 27, 2007.
10. Crnkovic, "Component Based Development Process and Component Life Cycle," 27th International Conference on IT Interfaces, IEEE, 2005.
11. L. 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. K. Lau et al, "The W Model for Component-based Software Development", EUROMICRO-SEAA 2011: 47-50.
13. H. Aris and S. S. Salim, "The Development of a Simplified Process Model for CBSD," 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," Journal of Systems and Software 61, 189–199, 2002.
15. C. 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. Bassam, A. Badareen,et al, "Reusable Software Component Life Cycle", International Journal of Computers, issue 2, vol. 5, 2011.
17. E. 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. J. H. Sharp and S. D. Ryan, "Component-Based Software Development: Life Cycles and Design Science-Based Recommendations", Proc CONISAR, v2 (Washington DC), 2009.
19. 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. T. Wijayasiriwardhane, R.Lai, K.C.Kang, "Effort estimation of component-based software development–a survey". IET Software, Vol.5, Iss. 2, pp.216–228, 2011.
22. 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
23. Stutzke, R. "Costs impact of COTS volatility'. Proc. Knowledge Summary: Focused Workshop on COCOMO2.0, Los Angeles, CA,1995
24. Aoyama, M.:'A component-based software development methodology', IPSJ SIG Notes,1996, 96,(84),pp.33–40
25. Abts, C. "Extending the COCOMOII software cost model to estimate Effort and schedule for software systems using commercial-off-the-Shelf (COTS) software components: the COCOTS model". PhD thesis, University of Southern California, 2004
26. Naunchan, P., Sutivong,D.:'Adjustable cost estimation model for COTS-based development'. Proc. 18th Australian Software Engineering Conf., (ASWEC2007), Melbourne, Victoria, April 2007, pp. 341–348
27. Khaled Hamdan et al, "The Influence of Culture and Leadership on Cost Estimation", UAE University, Al Ain, UAE and University of Sunderland, Sunderland, UK
28. M. Rizwan Jameel Qureshi, S. A. Hussain. "A reusable software component-based development process model". 88-94
29. Center for Systems and Software Engineering, http://csse.usc.edu/csse/research/COCOMOII/cocomo_main.html
30. IBM Rational Unified Process http://www-01.ibm.com/software/rational/rup/
31. Navid Hamrahi, Nasir Modiri, "Presenting an Approach for Establishing Information Security Project Management Based on PMBOK Standard", Life Science Journal, 10(8s), pp. 146-152, 2013.
32. 3/11/2014