I.J. Information Technology and Computer Science, **2013, 08, 1-14**
Published Online July 2013 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijitcs.2013.08.01

Empirical Study of an Improved Component Based Software Development Model using Expert Opinion Technique

Asif Irshad Khan
Ph. D. Scholar, Department of Computer Science, Singhania University, Jhunjhunu, Rajasthan, India
E-mail: alig.asif@gmail.com

Md. Mottahir Alam
Faculty of Engineering, King Abdulaziz University, Jeddah, Saudi Arabia
E-mail: mohammad.mottahir@gmail.com

Noor-ul-Qayyum
Faculty of Computing and Information Technology, King Abdulaziz University, Jeddah, Saudi Arabia
E-mail: nulqayyum@kau.edu.sa

Usman Ali Khan
Faculty of Computing and Information Technology, King Abdulaziz University, Jeddah, Saudi Arabia
E-mail: usmaniit@rediffmail.com

***Abstract*—** IT industry in the present market situation faces high demand for performance and burgeoning user expectations; with the pressure manifesting itself in three forms – Development Cost, Time-to-market and Product Quality. Researchers have proposed several techniques to effectively deal with these conflicting scenarios and draw optimized output. One of the relevant techniques in this context is Component Based Software Development (CBSD) with a targeted and discriminative approach influencing all phases of development. Although, CBSD proposes a multi-faceted approach in complex scenarios, its prime focus lies in "write once and reuse multiple times" methodology with either no or minor modifications. The model has been markedly successful in large enterprise applications with companies deriving benefits from shorter development time, increased productivity and better quality product. This research paper focuses and discusses Empirical Study of an Improved Component Based Software Development (ICBD) Model using Expert Opinion Technique which covers both component based software development as well as Component development phases. ICBD Model tries to overcome some of the issues in the contemporary CBD Models. A case study was conducted to investigate and evaluate our model by experienced professionals working in the IT industry. Results have shown that our improved model registers significant improvement over previous models suggested by other researchers.

***Index Terms*—** Empirical Study, Component Based Software Development, CBD, Software Process Improvement

I. Introduction

Software industry over the last decade is in enormous pressure for supplying high quality software that meets the market dynamic requirements and deadline. The extensive use of software in every walk of life and with new demands for the integration among different area brings new challenges and complication to the software industry.

Development of software using the traditional software development methodology is not an ideal approach to meet the dynamic nature of the today's requirement and demands of the market. Software process improvement is considered as

the hottest research area in the field of software engineering and recently the concept of software development using component based software methodology has been discussed in several research articles [1][2][3][4].

Component based software development (CBD) as a process of building software address some of the main challenges faced by today's software industry. CBD methodology is different from the traditional waterfall software development methodology. CBD focus not only on system specification and requirement

decomposition but it also focus on searching and identifying candidate component according to matching requirements instead of building / coding the software component from scratch. Components are software artifacts that are already well build, coded, tested and reused in other or similar domain. Szyperski [5] define Components are units of independent production, acquisition, and deployment that interact with each other to form a functioning system.

One of the benefits involved in CBD methodology is that CBD reduces software development cost as software can be rapidly developed by customization of products through the reuse of existing standardized components and hence it reduces the development cost. It also increases flexibility as there are more choices of candidate components available in the market, so it is up to the development team to choose right candidate component as per system requirements.

Further CBD reduces process risk as selected components are already tested and reused in other domain and hence there are very less chances of discovering bugs in the selected component [6]. CBD process also helps in developing enhanced and quality applications as components are used and tested in many different other applications and therefore there is less scope of finding bugs and faults as they are already considered in advance [6]. Maintenance cost is also found to be low in CBD methodology as it is easy to upgrade old / discontinued component with the new improved version release.

Although CBD methodology carries a lot many benefits, there are few challenges too. One potential challenge that software developers face is the choice of suitable components matching all the requirements and specifications because it is very difficult to identify a component which covers all the requirements. This problem can be overcome by making a negotiation on the requirements with the project stakeholders without compromising overall system objectives so as to pick the right candidate component among available choices, which can cater not all the requirements but most of the requirements [6]. For a component to be broadly reusable, it must be sufficiently general, scalable and adaptable, and therefore more complex and more demanding of computing resources.

Other challenges of the CBD methodology include adaptability of the components in a newer environment, integration of selected component in the overall system architecture, reliability and sensitivity of the components to future changes and testing of the individual components as well as the overall system. One of the main reasons behind these challenges is that even though few of the components may be developed in-house, there might be several third party or Commercial Off-The-Shelf (COTS) components whose source code may not be available to developers.

Asif *et.al.* [12] proposed a complete model for Component Based Software Development for reuse. This Model covers both component based software development as well as Component development phases. This research study is the extension of [12]. Here authors carried an empirical study on the Improved Component Based Software Development (ICBD) Model using Expert Opinion Technique.

The paper is organized as follows: Section 2 covers related work. Section 3 provides an overview of our Improved Component Based Software Development (ICBD) Model. Section 4 empirically evaluates the ICBD model using Expert Opinion. In section 5, research findings are analyzed. Section 6 presents a case study and lastly, section 7 draws conclusions and future works.

II. Related Work

In this section, we provided a brief summary of the different performance evaluation methods for component-based software development in the last few years.

T. Ravichandran [7] conducted a research to study the degree of success of assimilation of component –based development (CBD) model by various information systems (IS) departments using partial least square analysis. The results provide evidence that organizations better positioned to overcome knowledge barriers because of their knowledge stocks are likely to be further along in the assimilation process than others and that knowledge sharing by technology vendors positively influences technology assimilation by reducing the learning burden of the adopters. It was found that adoption barriers accounted for significant variance in CBD assimilation and that supply side mechanism can reduce these barriers.

V. Sagredo et.al. [8] conducted an experimental study to compare the Azimut approach with a systematized Ad-Hoc approach, regarding generated solutions quality and cost, and effort implied in applying each approach.

The results of the experimental study carried by [9] suggest that Azimut approach generated better quality solutions at lower cost but the effort required for this approach is higher than for Ad-Hoc approach. Results concerning effort are aligned with a post-experiment survey answered by the participants, where they suggest various ideas for improving the usability of Azimut tool user interface.

L.Grunske et.al. [9] describes a generic framework for predicting quality properties based on component-based architectures, which is derived from a comprehensive study of recent architecture evaluation methods. This generic framework defines common aspects between the different evaluation methods and enables the improvement of evaluation methods for specific quality properties, by transferring knowledge from one quality domain to the other.

Wanyama and Far [10] carried out an empirical study to compare three COTS selection methods, namely CEP (Comparative Evaluation Process), CRE (COTS-based Requirements Engineering), and FCS (framework for COTS selection). The validation process was carried out by comparing the FCS framework with two COTS selection methods in literature-CEP and CRE because they are some of the few COTS selection methods that have been applied in practice. The result conclude that it is necessary to specify when a COTS selection method works so that prospective users are enlightened upfront on possibility of the method to work in the prevailing project and COTS selection conditions. For example, CRE is a fairly good method for requirements elicitation and negotiation [10].

However, it has limited capability for the purpose of COTS selection. Therefore, it would have been better if the method had been presented as a requirements elicitation method but not a COTS selection method. Users prefer COTS selection methods which are easy to comprehend and to apply, and which guide them through the COTS selection process.

That is, the users become easily disappointed if they get into a dilemma and the method cannot guide them out of it. Moreover, if that happens, they prefer to use ad hoc methods to continue with the COTS selection process, than to employ another formal COTS selection method.

Although simplicity of COTS selection methods is desirable, it should not be achieved at the cost of highly degraded capability. For example, the DSS associated with the FCS necessitated extra training for the FCS-group.

The authors Jalali et.al. [11] present a new approach to evaluate performance of component-based software architecture for software systems with distributed architecture.

In this approach, at first system is modeled as a Discrete Time Markov Chain and then the required parameters are taken from, to produce a Product Form Queuing Network.

Limitations of source, like restrictions of the number of threads in a particular machine, are also regarded in the model. The prepared model is solved by the SHARPE software packages. It predicts throughput and the average response time and bottlenecks in different workloads of system and suggests required scale up to improve performance.

III. Overview of the Improved Component Based Software Development Model

Reusing of existing artifacts is the most important concern of the Component Based Software Development. These reusable artifacts have already been done with system requirement, architecture, components and case study [12]. In this section we mentioned an overview of ICBD Model which has been empirically studied by Asif et.al. [12] in this paper.

3.1 Component Based Software Development Lifecycle

The main phases of our ICBSD model are 'System Requirement and Analysis', 'System Design', 'Component Identification and Adaption', 'Component Integration Engineering', 'System Testing and Acceptance' and 'System Release and Deployment' as shown in Fig 1. Fig. 2 shows the details view of our ICBD model. A brief discussion about the model is as follows:

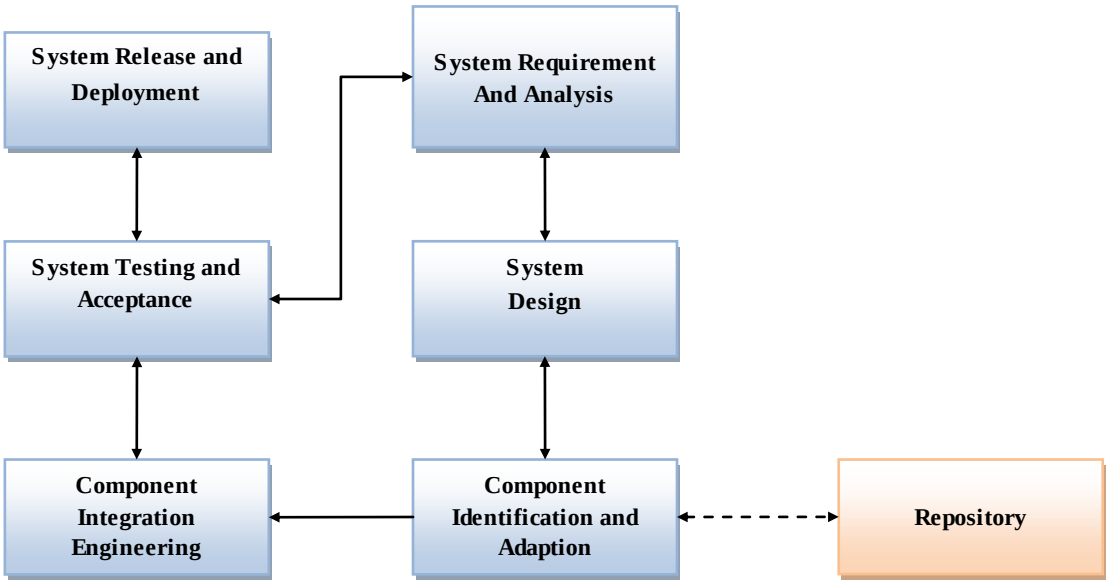

Fig. 1: an Improved Model for Component Based Software Development [12]

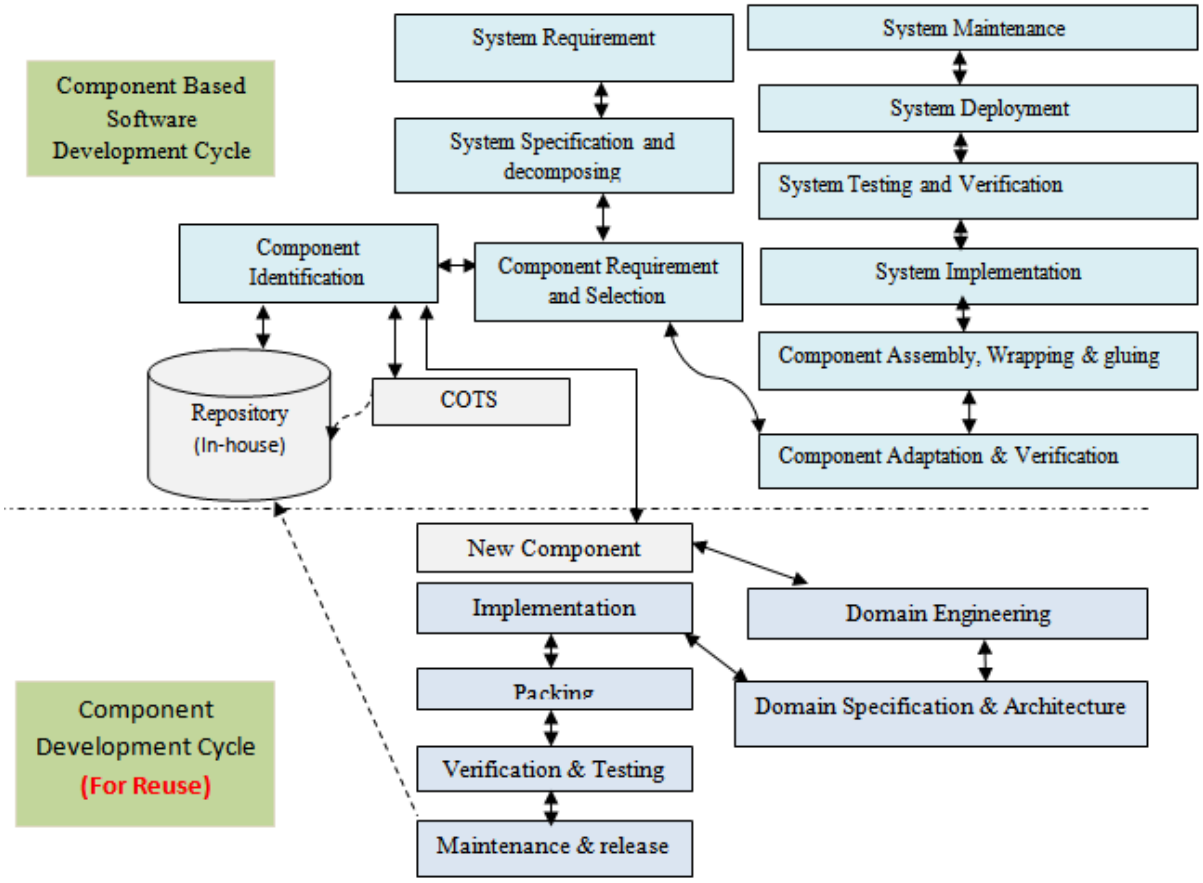

Fig. 2: A detail view of An Improved Model for Component Based Software Development [12]

3.2 System Requirement, Specification and Decomposition

First and foremost step in developing an application for a client or stakeholders is to study the system requirements by a team of software analyst to elicit the requirements. In CBSD this is done by reviewing existing System Requirement documents if any, and having meeting with the stakeholders.

Once the requirements are thoroughly collected, requirement analysis process starts to identify common requirements of the system and subsystems, and find possible reusable software components [13].

The major outcomes of this phase are: system requirements outline, identification of the components that can be reuse on the common requirements as system analysts has knowledge of available components in the in-house repository.

3.3 Component Requirement and Selection

Once the system requirements are collected a system architecture model is designed based on the matching requirement. The software team determines from the system requirements which of the software requirements can be considered to composition rather than building them from the scratch.

A complete cost benefit analysis is required to know various cost involved is adopting the component. These cost benefits analysis helps in making a decision to reuse a component or to acquire COTS [14].

3.4 Component Adaptation and Verification

Once candidate component is selected, issues related to its capability and fitness in the architecture need to be addressed. The selected component should be fit in the system architecture design, a study usually being done to know how far the selected component is compatible with the system architecture.

Verification of the component is being made through the software metrics and cost benefit analysis techniques [13]. The software architecture must be scalable, so that, components can be easily integrated into the system.

3.5 Component Assembly, Wrapping and Gluing

Developing an application by integrating components needs communication among the components through an interface or “glue”. Component wrappers help in isolating a component from other components of the system. Component wrappers are also helpful in mapping data representations.

3.6 System Implementation

Theoretically no coding is required during this phase but practically coding is usually required as all functionality is rarely found in a component and some functionality need to be coded which is not provided by component.

3.7 System Testing and Verification

The aim of testing an application is to investigate whether the software to be delivered is a quality product and it really works as per the given requirements and specifications.

In CBSD, the lack of details about component source code and design make it very difficult to track the faults occurring while using COTS components. This may leads to difficulty in testing of individual components and integrated systems.

3.8 System Deployment

System Deployment phase involves successful release of the complete product to the customer in the specified environment. In other word, the software is made available to the customer for use. System deployment must be delivered using some specialized tool to make the deployment easy for the customer.

3.9 System Maintenance

In a CBD model, up-gradation and substitution of components are the main job of the system maintenance. System Up gradation usually occurs when a COTS supplier releases a new version of a component or when a new COTS component is obsolete. Modifications to the code wrappers and glue code are required in system maintenance.

IV. Assessment using Expert Opinion

Different phases of our ICBD model are assessed and validated using expert opinion technique. Expert opinion technique is helpful in analysing some of systems specific questions related to system behaviour, usability and reusability, system performance and uncertainties [15]. This technique is also used to evaluate a product through group of experts using their side range of knowledge and experience in that area [16].

Expert opinion technique is a commonly used approach to evaluate software development model by many researchers. Several research papers mentioned the reliability of using expert opinion for example authors [13] conduct survey using an expert panel to validate their requirement process improvement model and found that expert predication on the requirements defect were very high when use in practice.

Moreover, Kitchenham et.al. [18] found in their research that taking expert opinion into consideration on a process model is helpful to validate model informally.

In our research we choose expert opinion technique to validate and assess phases of ICBD model architecture keeping the objectives of our research.

Questionnaires consisting of 15 questions based on different phases of the ICBD model were prepared. The selected participants were from software in-house development companies as well as software vendors. For selecting expert participants following criteria's are considered.

(1) Must work as a software architecture engineer or software system designer or developer with minimum 5-plus years of experience.

(2) Must have an experience and expertise of using state of art of CBD model and tools.

(3) Must be willingness to act as neutral assessor

(4) Willingness to provide valuable analysis and interpretation based on his experience.

The format of the questionnaire contain questions based on each phases of ICBD model and respondents were required to mark their expert opinion about the given statements in the form of questions.

We utilized the likert scale 1 to 5 which is a psychometric response scale most commonly used by research community in questionnaires to obtain participant's preferences or degree of agreement with a statement or set of statements. Points were labelled as shown in Table 1.

Table 1: Likert scale

5	Very high / Outstanding / Highly
4	High / Considerable / Moderately
3	Nominal / Average / Nominally
2	Low / Nominal / Poorly
1	Very low / Poor / Irrelevant

We represented the result of our analysis using frequency tables and bar chart showing the exact degree of analysis. Selection of expert participants to the blind expert opinion is the most crucial activity of the research. The main objective of this study is to validate ICBD model that promises a common architecture fulfilling write once and reuse any number of time policy with no or minor modification, this further results in shorter development time, increased productivity, quality product and reusability.

V. Research Finding

The evaluation of our ICBD model is made by using a survey method, Software practitioners are involved in the survey to validate the real challenges addressed in the ICBD model. The author evaluates issues by carrying out a cumulative evaluation of the issues in the ICBD model. To evaluate the main issues in the present state of art of CBD model, the authors has extensively evaluated the sub-issues related to the main issue which helps the authors to analyse and evaluate the ICBD model.

Questionnaire was sent to 50 different software professionals and practitioners in around 25 different software development companies around the globe. We received a total of 43 responses out of the total 50 questionnaires sent.

The other 07 did not returned back the survey. We excluded 5 received responses from the analysis as some of them were incomplete and some didn't meet out criteria of expert selection. Table 2- shows Code and its description used in our Analysis Tables.

Table 2: Code with their description

Code	Description
Q	Questions (sequence), see Appendix
LS	Likert scale
F	Frequency
P	Percentage (%)
CP	Commutative Percentage
AR	Average Rating

5.1 Evaluation Results for Ist Main Issue (System Requirement Decomposition, Specification and Design)

Table 3 shows three sample questions sequence of from the questionnaire. The complete questionnaire is given in the appendix section. The sequence of the question is important and it is referred as per given sequence in the appendix.

The questions are classified into different categories as per the areas they are associated with. To know the overall rating in a particular category, average is calculated.

For example, the first main issue is to propose an efficient technique for System Requirement decomposition, specification and Design phase. We addressed this issue in the ICBD model and to find how much improvement in the System Requirement decomposition, specification and Design phase of ICBD model, we added three questions in the questionnaire.

We want to know how much ICBD model will ease the developer's enormous efforts and time in searching, identifying and defining reusable components as per the system requirement needs.

From the table 3 – we conclude that the respondent average rating of 4.10 means that the respondents considered that our ICBD model ease their effort in searching, identifying and defining reusable components, also, they rated high improvement in searching the candidate component. Improvement in the design phase is also rated as high by the expert respondents.

Table 3: Evaluation results for Ist main Issue

Q	LS	F	P	CP	AR
1	1	0	0.0	0	
	2	0	0.0	0	
	3	11	28.9	28.9	4.00
	4	16	42.1	71.0	
	5	11	28.9	100.0	
	Totals	38	100.0		
3					
	1	0	0.0	0	
	2	0	0.0	0	
	3	6	15.8	15.8	4.18
	4	19	50.0	65.8	
	5	13	34.2	100.0	
	Totals	38	100.0		
12					
	1	0	0.0	0	
	2	0	0.0	0	
	3	7	18.4	18.4	4.11
	4	20	52.6	71.0	
	5	11	28.9	100.0	
	Totals	38.0	100.0		
Overall Average Rating					4.10

5.2 Evaluation Results for IInd Main Issue (Component Integration and Risk Analysis)

For Evaluation of the IInd main issue which is related to Component Integration and risk analysis we added two questions in the questionnaire as shown in table 4. The average rating for improvement in this category is 3.89 which mean the improvement falls to the right of nominal and close to high.

The respondent agreed that ICBD model addresses bridging interface gaps and approaching component wrapping and integration problems.

Table 4: Evaluation results for III[rd] main Issue

Q	LS	F	P	CP	AR
2	1	0	0.0	0	
	2	4	10.5	10.5	
	3	8	21.1	31.6	
	4	17	44.7	76.3	3.61
	5	9	23.7	100.0	
	Totals	38	100.0		
5					
	1	0	0.0	0	
	2	0	0.0	0	
	3	5	13.2	13.2	
	4	21	55.3	68.5	4.18
	5	12	31.6	100.0	
	Totals	38	100.0		
Overall Average Rating					3.89

5.3 Evaluation Results for III[rd] Main Issue (Cost Benefit Analysis and Project Planning)

To evaluate the third main issue which we categories as cost benefit analysis and project planning, we added three questions in the questionnaire. Clearly from the Table 5, it is concluded that the respondent rate nominal to high improvement in the cost benefit analysis and project planning in the ICBD model.

The respondent agreed that the budgetary planning and resource management are handled in a better way in the ICBD model. They also rated near to high improvement in the cost effectiveness of the software development compared to the older CBD Models.

5.4 Evaluation Results for IV[th] Main Issue (Documentation, Tagging and Repository)

Under evaluation of the IV[th] main issue, we addressed Documentation, Tagging and Repository in our model. We added 3 questions to evaluate this category in our questionnaire.

Table 6- shows the average rating for this category which is 3.85 which mean the respondent rating falls to the right of nominal and closer to the high improvement.

Table 5: Evaluation results for III[rd] main Issue

Q	LS	F	P	CP	AR
13	1	0	0.0	0	
	2	2	5.3	5.3	
	3	11	28.9	34.2	
	4	17	44.7	79.0	3.71
	5	8	21.1	100.0	
	Totals	38	100.0		
14					
	1	0	0.0	0	
	2	4	10.5	10.5	
	3	9	23.7	34.2	3.61
	4	15	39.5	73.7	
	5	10	26.3	100.0	
	Totals	38	100.0		
15					
	1	0	0.0	0	
	2	2	5.3	5.3	
	3	6	15.8	21.1	3.95
	4	18	47.4	68.5	
	5	12	31.6	100.0	
	Totals	38.0	100.0		
	Overall Average Rating				3.75

5.5 Evaluation Results for Issues Related Interoperability, Complexity, Reliability, Upgradeability, and Efficiency of the CBD Model

To know the improvement in terms of interoperability, complexity, reliability, upgradeability, and efficiency in the ICBD model, we included 4 questions in the questionnaire. Table 7 shows the evaluation results under this category in which the overall improvement rating is 3.52 .This means that the respondent rating falls to the right of nominal and closer to the high improvement.

To know the improvement in terms of interoperability, complexity, reliability, upgradeability, and efficiency in the ICBD model, we included 4 questions in the questionnaire. Table 7 shows the Evaluation results under this category.

The overall improvement rating under this category is 3.52 which means that the respondent rating falls to the right of nominal and closer to the high improvement.

The respondent rated nominal improvement for the statement which evaluates the ICBD model as mouldable which is as claimed by various methodologies used in software development processes esp. agile methodology.

Table 6: Evaluation results for IVth main Issue

Q	LS	F	P	CP	AR
7	1	0	0.0	0	
	2	1	2.6	2.6	
	3	20	52.6	55.2	
	4	12	31.6	86.8	3.50
	5	5	13.2	100.0	
	Totals	38	100.0		
8					
	1	0	0.0	0	
	2	0	0.0	0	
	3	8	21.1	21.1	
	4	18	47.4	68.4	4.11
	5	12	31.6	100.0	
	Totals	38	100.0		
9					
	1	0	0.0	0	
	2	2	5.3	5.3	
	3	6	15.8	21.1	
	4	18	47.4	68.5	3.95
	5	12	31.6	100.0	
	Totals	38.0	100.0		
	Overall Average Rating				3.85

Also, the respondent rated near to high improvement against the statement which evaluates the ICBD model shall improve efficiency in SDLC processes across all domains, technologies and sizes of the projects.

Further, most of the respondents rated nominal against the statement which evolutes the ICBD model moldable, a claim that it holds in reference to the prevalent frameworks in the industry esp. the spring framework.

5.6 Overall Evaluation Results for the ICBD Model

Overall results of the evaluation results for the ICBD Model are shown in Table 8 and bar chart fig 3. From the overall average rating of 3.82, it is concluded that the respondent improvement rating is nominal to high for our ICBD Model.

Table 7: Evaluation results for interoperability, complexity, reliability, upgradeability, and efficiency of the ICBD Model

Q	LS	F	P	CP	AR
4	1	0	0.0	0	
	2	1	2.6	2.6	
	3	20	52.6	55.2	
	4	10	26.3	81.5	3.55
	5	7	18.4	100.0	
	Totals	38	100.0		
6					
	1	0	0.0	0	
	2	2	5.3	5.3	
	3	15	39.5	44.8	3.54
	4	14	36.8	81.6	
	5	6	15.8	97.4	
	Totals	37	97.4		
10					
	1	0	0.0	0	
	2	1	2.6	2.6	
	3	14	36.8	39.4	3.66
	4	18	47.4	86.8	
	5	5	13.2	100.0	
	Totals	38.0	100.0		
11					
	1	0	0.0	0	
	2	2	5.3	5.3	
	3	22	57.9	63.2	3.32
	4	10	26.3	89.5	
	5	4	10.5	100.0	
	Totals	38.0	100.0		
	Overall Average Rating				3.52

Clearly, in our model we have tried to suggest improvement in every phase of the CBD model from System Requirement decomposition, Specification and Design to Component Repository, upgradeability, Integration and Cost Benefit Analysis.

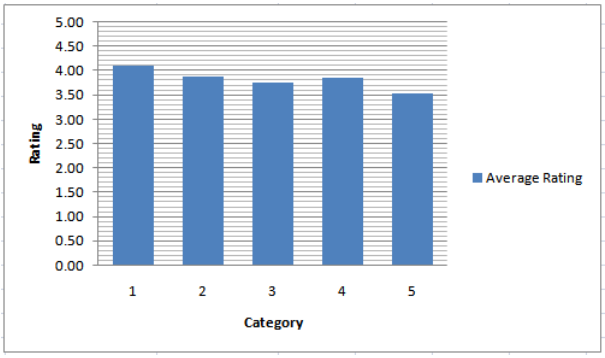

Fig. 3: Bar chart of overall average Improvement in ICBD Model

Table 8: Evaluation results for overall ICBD Model process Improvement

S. No	Category	Average Rating
1	Improvement in System Requirement decomposition, specification and Design	4.10
2	Improvement in Component Integration and risk analysis process	3.89
3	Improvement in Cost Benefit Analysis and Project Planning process	3.75
4	Improvement in Documentation, Tagging and Repository	3.85
5	Improvement in interoperability, complexity, reliability, upgradeability, and efficiency process	3.52
	Overall Average Rating	3.82

VI. Case Study

We illustrated our model with a case study of a Customer Relationship Management Module (CRM) of a Saudi Arabian based company Binzagr [19]. Binzagr operates its own sales, service, marketing and business development functions in Saudi Arabia from its headquarter in Al Khobar and via regional offices in Jeddah, Jubail, Riyadh and Yanbu [19].
Binzagr requires its sale persons to play smart when they are interacting with customers on daily, weekly, monthly basis. For organized, structured and efficient selling of products, Binzagr requires customer transactions to be digitized.

To ensure that the software system developed for Binzagr should be in-lined with their requirements, various models were designed at system requirement specification and system design stages like Application Hierarchy, Database Model, General Architecture model etc. Building above mentioned models helped us in analysis and design feature of overall architecture and behavior of the system. Further these models helped in understanding the System architecture, reduce the ambiguity that occurs in natural language descriptions and also, it helps the development team to visualize the design and architecture of the system.

In our approach we kept the adaptive nature of the system in our mind so as to use the application in different context and situation. We first built a feature hierarchy, and then associating features with component's philosophy as show in fig 4 to fig 8. Following points were considered and implemented in the System.

- Some of the high-level requirements of Binzagr some of them are listed as follows.
  i. For the management of orders placed by different customers and distribution of products, it is required that the real time stock information and delivery dates be on the finger tips of sales persons.
  ii. For keeping exact track of amount received and balances pertaining to a particular customer and management of invoices with the convenience of a few touches.
  iii. For planning and setting up targets related to increase in sales of particular products for certain customers, it should be easy for the sales persons to fetch that information quickly while he is in the customer's premises.
  iv. For keeping customers abreast of the latest promotions and schemes on products and mobility of products, it is important for the sales persons to be handy with all the required information.
- Architecture choice [Base technology choices, Frameworks choices, Integration method choices] so as to selecting up-to-date core technology for the system development.
- Components choices [picked based on system specification and architecture].
- Quality of service consideration that the system must meet [such as system performance, usability, Applicability, Technicality, security, robustness and portability] .
- Sequence diagram, activity diagram and component diagrams were drawn to study and analyze the message flow between the components.
- Component interfacing and interaction with other components in the System.
- Costs benefit analysis.
- Testing and implementation of the system.

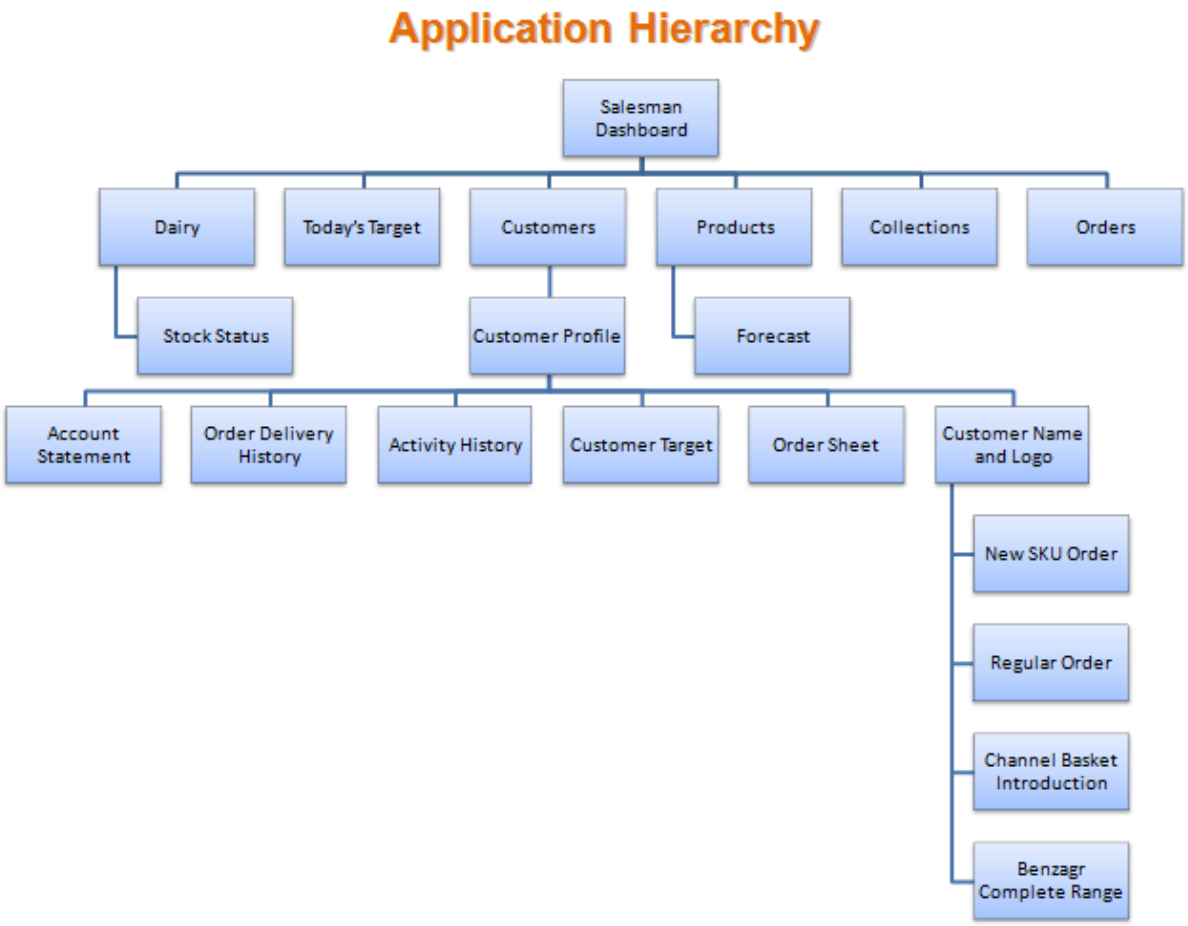

Fig. 4: Application Hierarchy of the solution (Salesman Dashboard)

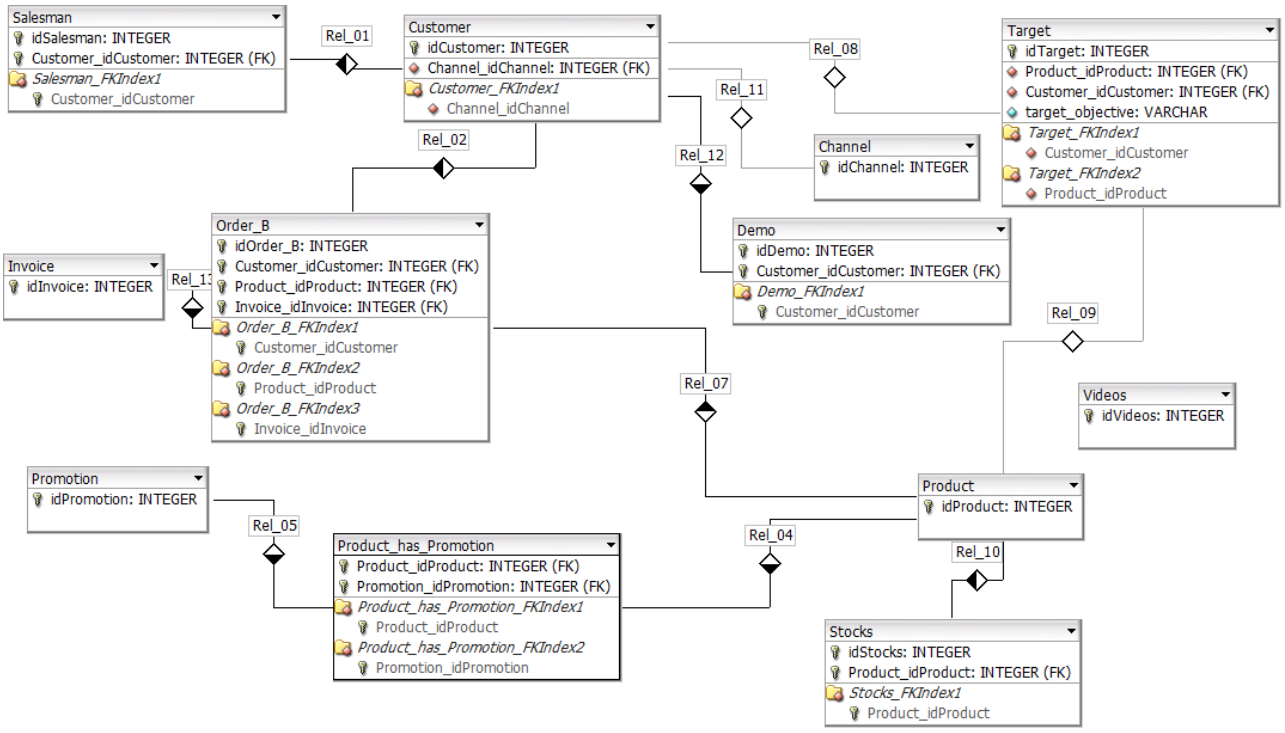

Fig. 5: Database Model

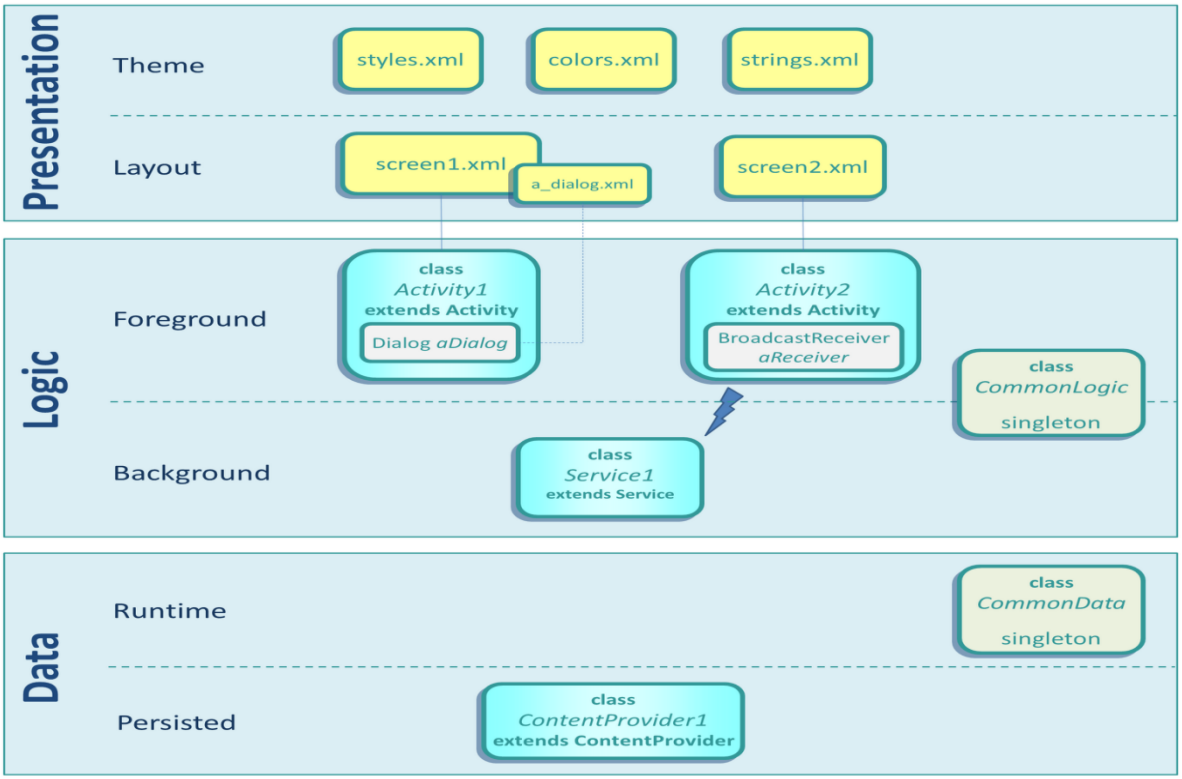

Fig. 6: General Architecture Diagram

The application hierarchy of the proposed salesman dashboard is shown in fig. 4, while fig. 5 explains the database model of the CRM module.

General System architecture diagram is shown in fig. 6, and Technical Specification of the CRM module is shown via fig. 7. The approach leads to a highly modular and extensible integrated system. Multi-tier applications architecture (client, web, and business) was adopted, as per the needs of case study i.e., Model View Controller (MVC) design pattern.

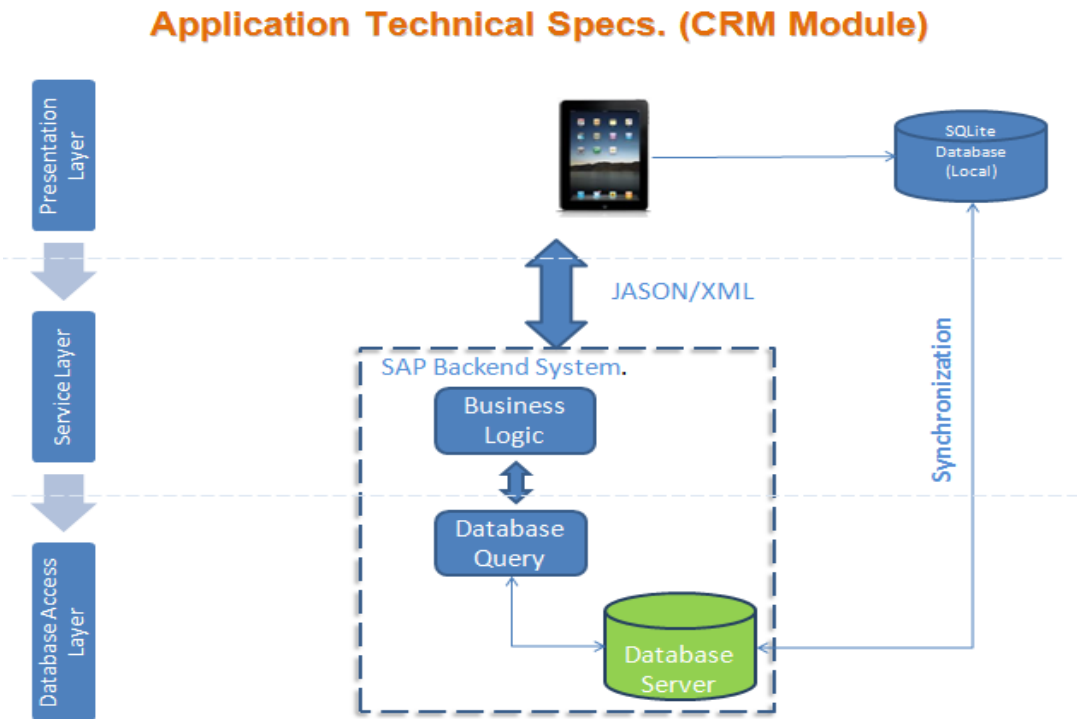

Fig. 7: Application Technical Specification (Customer relationship management Module)

VII. Conclusion and Future Work

Different software life cycle models have their own advantages and disadvantages. In this paper, we have discussed a number of evaluation tools for evaluating our Improved Component Based Software Development Model. In the component based development, reusing of existing artifacts is the most important concern. We assessed and validated various phases of our improved ICBD Model using expert opinion technique.

We conducted extensive questionnaires based survey on different phases of the ICBD Model. Questionnaire containing statements in the form of questions were sent to the expert respondents for evaluation of different phases and parameters of the Model based on their industrial experience. The participants were selected from software in-house development companies as well as software vendors. Overall average rating of the expert's opinion for the ICBD Model was 3.82 (on scale of 5) that proved the respondent improvement rating is nominal to high for our ICBD Model.

The model is implemented on Binzagr CRM that mainly focuses on streamlining customer interactions. In our future work we will further validate our model using the formal mathematical modeling techniques and evaluate the ICBD Model functionality.

Acknowledgements

The authors gratefully acknowledge many expert individuals that have facilitated the undertaking of this study. We would like to extend our sincere thanks to all the expert participants who participated in our survey and provide us their valuable expert feedback on our model which helps us to evaluate ICBD model.

References

[1] Crnkovic I., Sentilles S, Vulgarakis A, Chaudron M.R.V: A classification framework for software component models. IEEE Transaction on Software Engineering, 2011, 37(5), 593 - 615.

[2] Landry C, Benjamin Klatt, Klaus Krogmann, Reverse Engineering Software-Models of Component-Based Systems, Proceedings of the 12th European Conference on Software Maintenance and Reengineering, 2008, 93-102.

[3] Colin A., Oliver H. Iterative and incremental development of component-based software architectures: CBSE '12 Proceedings of the 15th ACM SIGSOFT symposium on Component Based Software Engineering, New York, NY, USA, 2012 , 77-82.

[4] Lata N, Umesh, K, Sushil C, Elite: A New Component-Based Software Development Model, Int.J.Computer Techology & Applications, 2012, 3 (1),119-124.

[5] Szyperski, C., Component Software: Beyond Object-Oriented Software, Addison-Wesley, 1998.

[6] S. Mahmood, R.Lai and Y.S. Kim, Survey of Component-based software development, IET Software, 2007, 1(2).

[7] T. Ravichandran , Organizational Assimilation of Complex Technologies: An Empirical Study of Component-Based Software Development, IEEE Transactions on Engineering Management, 2005, 52(2).

[8] V. Sagredo, C. Becerra, and G. Valdes, Empirical Validation of Component-based Software Systems Generation and Evaluation Approaches, The CLEI Electronic Journal, 2010.

[9] L. Grunske, Early quality prediction of component-based systems - a generic framework, Journal of Systems and Software, 2007, 80(5), 678 -686.

[10] Tom W. and Behrouz H. Far, An Empirical Study to Compare Three Methods for Selecting COTS Software Components, International Journal of Computing and ICT Research, 2008, 2(1).

[11] Jalali S S, Rashidi H, Nazemi E , A New Approach to Evaluate Performance of Component-Based Software Architecture, Fifth UK Sim European Symposium on Digital Object Identifier, 2011, 451 – 456.

[12] Asif I K , Noor, Usman A K, An Improved Model for Component Based Software Development, Scientific & Academic Publishing, Journal of Software Engineering, 2012, 2(4).

[13] Sajid R. , Moving Towards Component Based Software Engineering in Train Control Applications, Final thesis, Linköpings universitet, 2012, Sweden.

[14] Critical software practices, Pro-Concepts LLC, Online available: http://www.spmn.com/www2/16CSP.html

[15] Ali B M, Kitchenham B, Assessment of a Framework for Comparing Software Architecture Analysis Methods. Proceedings of 11th International Conference on Evaluation and Assessment in Software Engineering, BCS, 2007.

[16] Matthias R, Tools & Techniques Expert Opinion, Online available http://www.idemployee.id.tue.nl/g.w.m.rauterberg/lecturenotes/UFTexpertopinion.pdf

[17] Lauesen S. and O. Vinter, Preventing Requirement Defects: An Experiment in Process Improvement,

Requirements Engineering Journal, 2001, 6 (1), 37-50.

[18] Kitchenham B, et al., An empirical study of maintenance and development estimation accuracy. Journal of Systems and Software, 2002, 64 (1), 57-77.

[19] Profile, Binzagr group of companies, Online available:http://www.bfim.com.sa/en/Profile.htm

Appendix

(a) Questionnaire

Q#	Question	Response
1	Do you think requirements decomposition as defined in the proposed CBD model will ease your effort in searching, identifying and defining reusable components?	☐ Outstanding ☐ Considerable ☐ Average ☐ Nominal ☐ Poor
2	Is the model relevant to the problems faced in integration of components borrowed from other domains?	☐ Highly ☐ Moderately ☐ Nominally ☐ Poorly ☐ Irrelevant
3	Do you believe the requirements gathering phase is the appropriate juncture as proposed in the CBD model to work on component search?	☐ Very low ☐ Low ☐ Nominal ☐ High ☐ Very high
4	How much improvement does the Proposed Model registers over other CBD models in terms of interoperability, complexity, reliability, upgradeability, and efficiency?	☐ Very low ☐ Low ☐ Nominal ☐ High ☐ Very high
5	How do you rate the proposed CBD model in bridging interface gaps and approaching component wrapping and integration problems?	☐ Very low ☐ Low ☐ Nominal ☐ High ☐ Very high
6	How much do you think the proposed CBD model is moldable to various methodologies used in Software Development Processes esp. Agile Methodology?	☐ Very low ☐ Low ☐ Nominal ☐ High ☐ Very high
7	Do you believe the proposed CBD model handles component classification through repository in a way it adds to the efficiency of the developer in large organizations with huge component database?	☐ Very low ☐ Low ☐ Nominal ☐ High ☐ Very high
8	Does the CBD model duly handle the versioning of the components that are heavily used by multiples teams esp. in the perspective of large teams?	☐ Very low ☐ Low ☐ Nominal ☐ High ☐ Very high
9	How do you rate the effectiveness of the documentation phase in searching and identifying components in the repository by diverse teams?	☐ Very low ☐ Low ☐ Nominal ☐ High ☐ Very high
10	Do you think the proposed CBD model shall improve efficiency in SDLC processes across all domains, technologies and sizes of the projects?	☐ Very low ☐ Low ☐ Nominal ☐ High ☐ Very high
11	How much relevance do you think the proposed CBD model holds in reference to the prevalent frameworks in the industry esp. Spring Framework?	☐ Very low ☐ Low ☐ Nominal ☐ High ☐ Very high
12	Do you find considerable improvement in the design phase with the proposed CBD Model?	☐ Very low ☐ Low ☐ Nominal ☐ High ☐ Very high
13	Do you believe the proposed model improves the cost effectiveness of the software development compared to the older CBD Models?	☐ Very low ☐ Low ☐ Nominal ☐ High ☐ Very high
14	Do you think the proposed model influences the budgetary planning and resource management in a better way?	☐ Very low ☐ Low ☐ Nominal ☐ High ☐ Very high
15	How do you think the proposed CBD model will affect the budget allocation, code maintenance and project plan in case third party components are identified?	☐ Very low ☐ Low ☐ Nominal ☐ High ☐ Very high

Authors' Profiles

Mr. Asif Irshad Khan: received his Bachelor and Master degree in Computer Science from the Aligarh Muslim University (A.M.U), Aligarh, India in 1998 and 2001 respectively. He is a Ph.D. research Scholar in the Department of Computer Science, Singhania University, Jhunjhunu, Rajasthan, India.

He has more than seven years experience of teaching as lecturer to graduate and undergraduate students in different universities and worked for four years in industry before joining academia full time.

He has published more than 11 research papers in International journals, His current research interests include software engineering with a focus on Component Based and Agent Oriented Software Engineering.

Mr. Md Mottahir Alam: has around six years of experience working as Software Engineer (Quality) for some leading software multinationals where he worked on projects for companies like Pearson and Reader's Digest. He is ISTQB certified software tester. He has received his Bachelors degree in Electronics & Communication and Masters in Nanotechnology from Faculty of Engineering and Technology, Jamia Millia Islamia University, New Delhi.

He is presently working as a Lecturer in the Faculty of Electrical and Computer Engineering, King Abdul Aziz University, Jeddah, Saudi Arabia. His research interest includes Software Engineering esp. software reusability, Object-oriented, Component-based and Agent-based software engineering.

Mr. Noor-ul-Qayyum: is currently working as a lecturer in King Abdul Aziz University. He has industry experience in SCORM based e-learning courseware development using ADDIE model. His research interest includes e-learning, software watermarking, and mobile agent security issues.

Dr. Usman Ali Khan: is an Associate Professor in the Information Systems Department, Faculty of Computing and Information Technology, King AbdulAziz University, Jeddah, Saudi Arabia. He received his Ph.D. in Software Engineering from the Integral, University, India. He has been working is the field of research and teaching at graduate and undergraduate level since 1995. He has published many research papers at national and international forums.

How to cite this paper: Asif Irshad Khan, Md. Mottahir Alam, Noor-ul-Qayyum, Usman Ali Khan,"Empirical Study of an Improved Component Based Software Development Model using Expert Opinion Technique", International Journal of Information Technology and Computer Science(IJITCS), vol.5, no.8, pp.1-14, 2013. DOI: 10.5815/ijitcs.2013.08.01

