Available online www.ejaet.com

European Journal of Advances in Engineering and Technology, 2015, 2(7): 56-61

Research Article ISSN: 2394 - 658X

An Approach for Optimizing CPU and Memory Performance by Selection and Deactivation of Optional Components

Jasneet Chawla and Ashima Singh

Department of Computer Science Engineering, Thapar University, Patiala, India
er.jasneetchawla@gmail.com

ABSTRACT

Component based software engineering has become a modern approach for software development. It is a multifaceted approach in complex scenarios that focuses on "develop once reuse multiple times" methodology. As per user requirements, the components from repository are selected and integrated to develop software. It helps developers to deliver high quality softwares within a less amount of time and cost with less effort. Component based softwares are based on modular approach that provides the benefit of easy scalability and flexibility of the software. But along with this advantage comes the disadvantages too. Adding more components may result in performance degradation in terms of responsiveness, throughput, bandwidth as well as incompatibility problems in terms of resource requirement. This research paper propose to identify the non participating components of a component based software like excessive graphics , unnecessary animations and other optional components and deactivating them for that time to increase compatibility and optimize the system for best performance. By using the case study of Windows XP as reference, we successfully demonstrated that a component based software that has higher system requirements can run smoothly on a lower configured system by identifying and disabling the non participating sub-components. We successfully achieve up to 27.43% increase in performance by this method.

Key words CBSE, COTS, CBD, Non Participating, System Compatibility

---

INTRODUCTION

Component based software engineering (CBSE) has become a modern approach of software development that provides an optimal, efficient, economic and quick software development as per user requirements. This is achieved by using external components as well as in house built components. The main objective is to shorten the product development time, reduce the cost, and at the same time to improve the system quality drivers that includes Adaptability, Maintainability, Integration, Reliability, Flexibility and Interpretability [1]. The following Fig 1 represents the component based software development process model that covers two major aspects Domain engineering and Component based development. Domain engineering constructs a domain model of application that is used during CBD for analysis of user requirements, a structural model which is used as an input to architectural design, and provides reusable components to developers for component based development.

Component-Based Development (CBD) is an approach of developing software systems by reusing pre built components. CBSE is the best process model because it divides the application into two parallel activities i.e. component based development and domain engineering that helps to develop new components and reuse the old components from COTS for developing the new applications. So, as a result, CBSE process improves quality and productivity and hence reduces development schedule and effort [3]. The programmer uses already existing components to fulfil the desired function which is required in the new application. Let's say, if a component is created by some developer which allows a user to "log in" then other programmer can use it in their applications for that functionality.

Component based development involves the analysis of user requirements and enhancement of architecture that is suitable in accordance with the analysis model being designed for the application. Then the architecture is supplied with components that are either already existing or newly developed. If the component is available to be reused, it has to undergo component qualification (i.e. ensuring that the selected component executes all the desired functions), component adaptation (i.e. ensuring that constant ways for managing resources are used for every component and interfaces are realized in a constant way) takes place, however if not then component is engineered and then finally all the components are integrated and tested.

It is a reuse-based approach to defining, implementing and composing loosely coupled independent components into systems. Modular approach of CBSE provides the properties such as scalability and flexibility that helps in easy extendibility of a functionality of software. But this leads to some performance and compatibility issues.

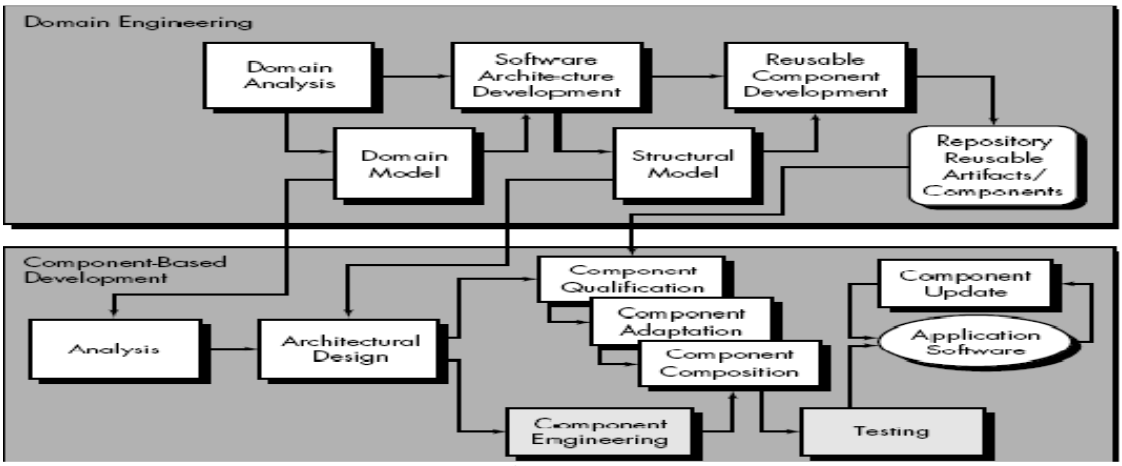

Fig.1 Component Based Software Engineering Process Model [2]

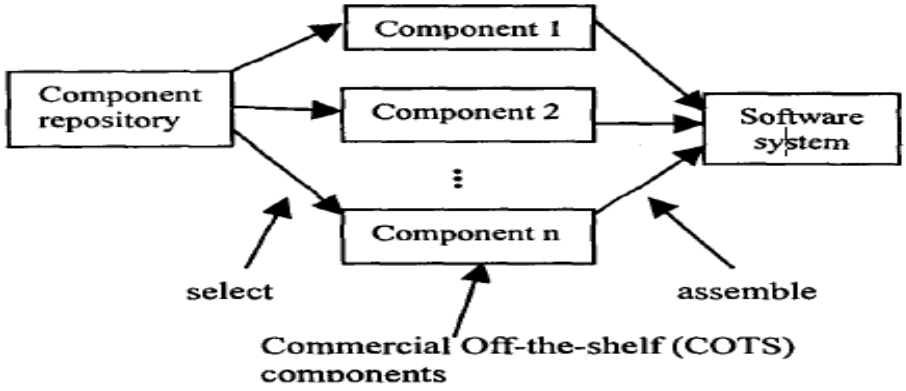

Fig.2 Component Based Development [3]

LITERATURE SURVEY

The wide spread literate found in the component based software engineering and its various optimization techniques. Gaoyan [4] described about the verification technique for CBSE through formal analysis as well as traditional Software Engineering. Summarizing most of the principle research done in this field is Gaoyan demonstrated an Automata-Theoretic approach for model checking. In this paper, the Model-checking Black-box Testing Algorithms or program for Systems with Unspecified Components Here the he presented both LTL (linear time temporal logic) and CTL (computation tree logic) model-testing algorithms for the systems with unspecified software components. The LTL (resp. CTL) formula about the system, directly deduce the condition in terms of the communication graphs. The approach suggested by Egon et al [5] He says that the without verifying components and interaction it was nearly impossible to robust systems. The Testing of such systems required combination of unit and integration tests, and must deal with the verifying contracts that enabled interaction of components. Osama et. al[6] identified various problems related to CBD like inadequate inclusive tools, less efficient methods to manage and collect the information required for selection of COTS for a specific application. He proposed an Optimal Performance Model (OPM) that made the selection of COTS for ERP systems in a more effective and efficient manner. OPM is based on several Standards of Quality. This information helps in attaining more useful and quality based ERP solutions (whether for implementing a new ERP or upgrading the existing one) that meet the business needs in a better way.

Balsamo et al [7] presented a paper in which the issue of performance evaluation at early stages of SDLC was addressed. An approach that was based on Queuing network analysis evaluation of CBS was proposed. By using software specifications that are annotated in terms of UML use case, activity and deployment diagrams are used to analyse the performance bound. This is based on multi-class and multi-chain QN model. The approach performed successful performance evaluation at architectural level. Kaur [8] addressed the need to recognise reusable components from a software and their reusability was determined using neural networks. The approach works in two steps. In First step, the code is parsed to calculate metric values: Cyclometric Complexity Using Mc Cabe's Measure, Halstead Software Science Indicator, Regularity Metric, Reuse-Frequency Metric and Coupling Metric.

The generated metric values are supplied as input dataset for different neural networks to evaluate reusability. In second step, the neural network is designed and is used for evaluation. Firstly, the neural networks are trained using the training dataset. After training, the neural network is evaluated against the testing data and comparison is made on the basis of MAE (Mean Absolute Error), RMSE (Root Mean Squared Error) and Accuracy values of neural networks. Khan [9] Proposed an Improved Component Based Development Model that uses Expert Opinion Technique to overcome some of the problems associated with Traditional Component Based Development Models. Figure 3 shows an overview of various phases of ICBD Model and Figure 4 show a detailed view of ICBD Model.

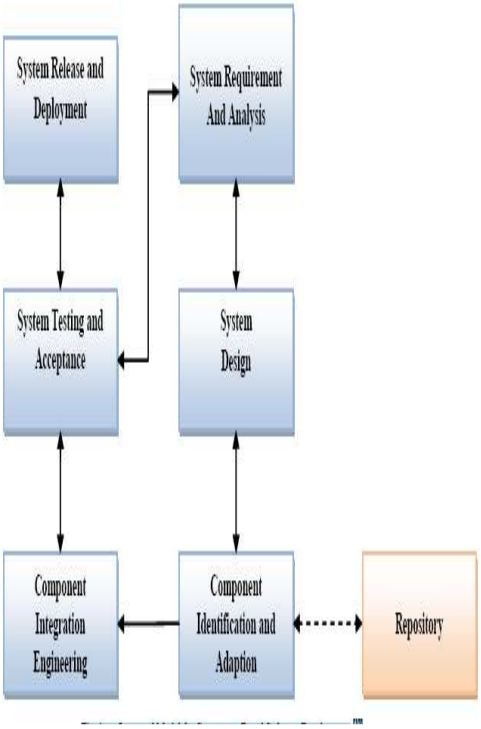

Fig.3 An Overview of various Phases of ICBD Model [9]

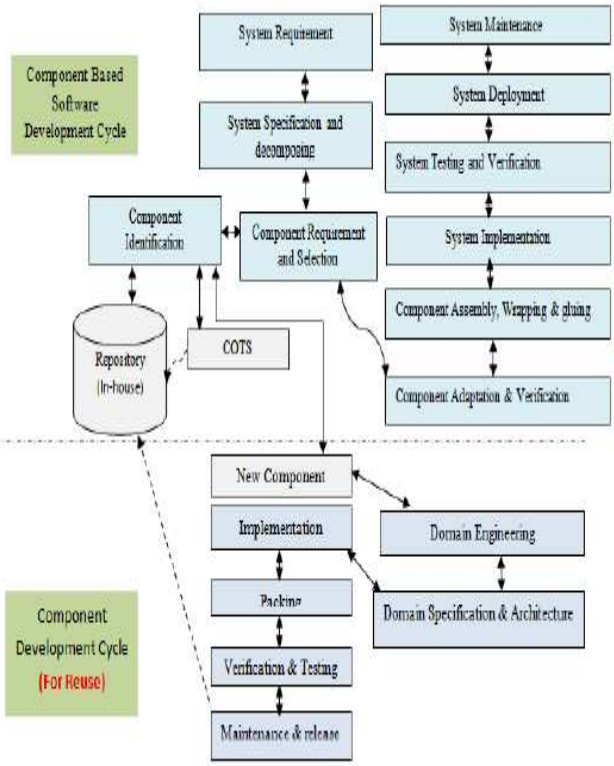

Fig.4 A Detailed View of ICBD Approach [9]

Experts who were software engineers and those who have been working with component based software in many renowned organizations were sent the questionnaires and their responses were analyzed. Likert Scale was used by the experts. It was analyzed from the survey that the rating for ICBD Model was between nominal to high. Kahkipuro[10] presented a performance modelling framework to produce predictive performance models that can help in generating information related to performance at all stages of SDLC for development and maintenance of component based distributed systems. The framework describes a UML based notation for describing performance model and set of special techniques for the modelling of component based distributed systems. Diaconescu and Murphy [12] devised an approach AQuA for automating management of component based enterprise systems in which multiple component variants serving same functionality are categorised as a redundant group. At run time based on the execution environment, the selection of component from RG is done to optimize the system for best performance. Bertolino and Mirandola[13] devised an easy to use technique for prediction and analysis of performance of a component based system. For this purpose, a CB-SPE framework composing a methodology for software performance engineering and a supporting tool was proposed. The approach is divided into component layer and application layer. At component layer, developers model the schedulable resources demand of individual performance service in dependence to environment parameters. Parameteric performance evaluation of components is done in isolation. At Application layer, software architecture pre-selects the performance models and then composes them into architectural models. They model the flow of control using sequence diagrams. CB-SPE technique also includes the free available modelling tools, transformation tools and performance solver tools.

Beydeda [14] proposed that it is very beneficial to use of CBSE based softwares for the gigantic software systems as it surely have benefits for cost cutting as well as faster delivery. However, CBSE complexity remains an issue in software engineering. The user of a component was generally found a problem with the information's that is necessary is not available in general as it is black box and having different vendor make. In absence of adequate information distinguishes testing of components from other software entities that is called as non-component-based development. The author gave an overview of testable bean approach, built in testing approach and STECC approaches to testing component. This method described can simplified as component user's test insofar that the component user might not need to create test cases for testing.

Zheng et al [15] demonstrated that software yield was often configured with commercial-off-the-shelf (COTS) components. Whenever the new releases of these components were made accessible for incorporation and testing, source code was frequently not specified. There are wide regression test selections processes are developed and have been exposed to be cost efficient. However, the majority of these test selection techniques depends upon the access to source code for change detection. Based on their earlier work, it was studied the solution to regression testing COTS-based applications that include the mechanism of dynamic link library (DLL) files. They developed the Integrated - Black- box Approach for Component Change Identification (I- BACCI) technique that aims regression tests for component based application programs based upon static binary code analysis. The possibility case study was conducted at ABB on the CBSE software products written in some C/C++ language to demonstrate the efficiencies of the I-BACCI. As a results of the case study specify this process could cut the requisite number of regression tests by as much as 100 percentages. They propose that software crop were often configured with commercial-off- the-shelf (COTS) components. When new releases of these components were made available for incorporation and testing, source code was frequently not given. Different regression test selection techniques have been developed that is cost effective. However, the most of these test selection methods rely on access to source code for transform recognition. The whole work, were studying the solution to regression testing COTS-based programs that integrate components of dynamic link library (DLL) files. Navneet et al [16] introduced that rapid and quick development of software's can be made possible by means of CBSE. In CBSE, the software product was built by combining different techniques of on hand software from diverse suppliers or vendors. By means of this technique, cost and time of the software package reduced significantly. However in the testing stage there are many challenges for a software tester, due to limited access to the source code of the reusable component of the software product. The component meta-data could be used to join extra information with the components to facilitating of CBSE based software testing. The Black box testing was used as in this method the code of the component was not available. Generally, a component has a concealed interface and a tester is not able to put input values in it until its interface was not finished. The challenges in component based testing by use of metadata method for black box testing would be used when component's interface not exists. Here they demonstrated the techniques that how the metadata could be used in black box testing.

PROBLEM STATEMENT

Modular approach of Component based software engineering has advantages of adding and removing and updating system components as per user requirements. It makes system more efficient towards problem solving. But adding more and more components have some side drawbacks too. It degrades the system performance measured in terms of system response time, throughput and also degrades the system compatibility measured in terms of system resources. This overall degrades gradually the components based softwares performance for system with constant configuration.

As software versions are upgrading very rapidly with changing business requirements whereas the hardware components of a system almost remains constant that becomes incompatible in due course of time which is a major problem.We propose to indentify and detect the non participating components viz.

a) Excessive Graphics
b) Unnecessary animations
c) Other non required subcomponents

And deactivate them for that particular time. This will optimize the system for best performance. Thus the software will be able to run even on lower configured systems efficiently.

OBJECTIVE OF RESEARCH

The principal objective of this research includes

a) To Study and examine the existing Component Based Software Development for their performance and compatibilities.
b) To propose an approach to enhance the performance and compatibility of component based system.
c) To validate the proposed approach using a case study.

RESEARCH METHODOLOGY CASE STUDY

As Microsoft windows XP is widely available and accessible for common people therefore we adopt it as a media of Case study.

Table -1

Memory Consumption	
When all animations are enable	2.69 GB
When all animations are disable	2.64 GB
Difference	0.05 GB
Percentage Increase	(0.05/2.69)*100=1.85%

Table-2

CPU Utilization	
With full Animation	100%
With Reduced Animation	47%
Difference	53%

Overall Increased Performance Index=(1.85+53)/2=27.43%

Thus we enhanced the Performance of the system by keeping disable the optional components.

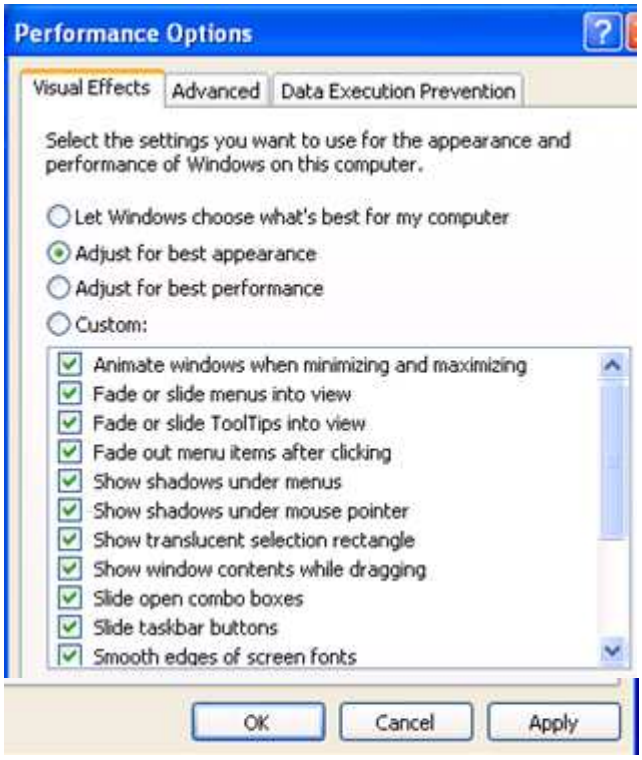

Fig.5 In case of enabling all the non functional sub component

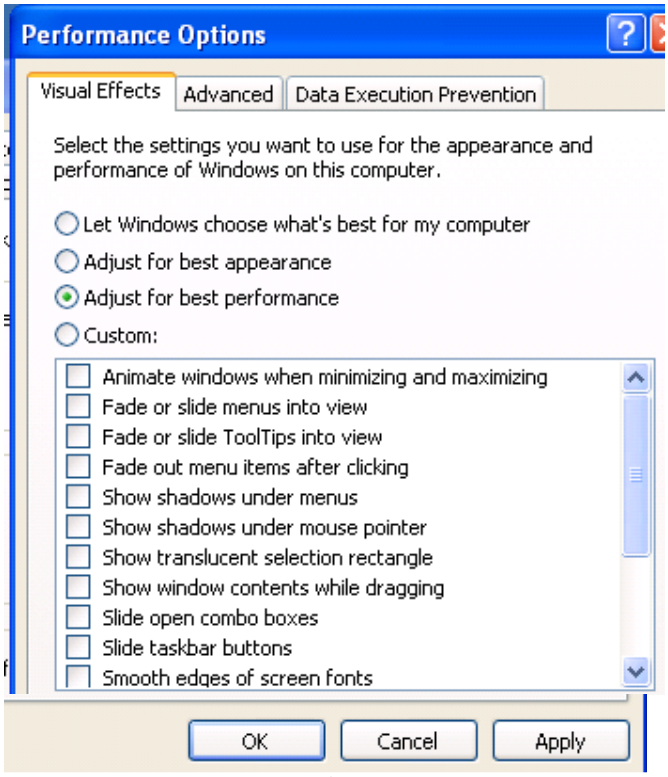

Fig.6 Disabling all non functional sub components

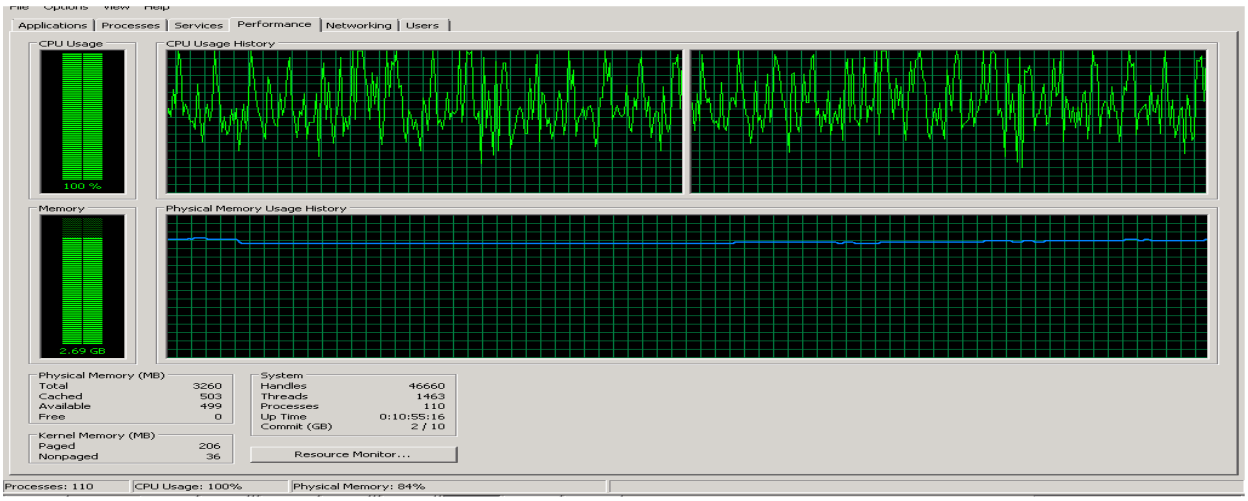

Fig.7 Performance (In case of enabling all non participating sub components)

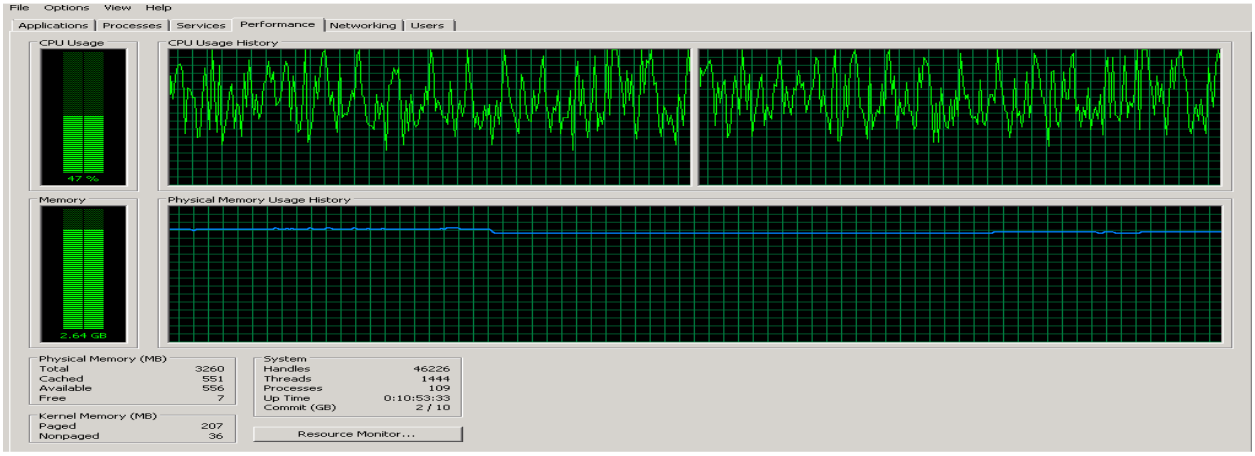

Fig.8 Performance (In case of disabling all non functional sub components)

CONCLUSION

We successfully demonstrated that a component based software that has higher system requirements can run smoothly on a lower configured system by identifying, selecting and disabling the non participating and non required functional components. This results in a better resource management. We successfully achieved up to 27.43% increase in performance by this method.

The future work aims to develop an application or an independent component that detects and identifies the optional i.e. non participating and non required functional components automatically and deactivates them for a period of time to enhance the performance of the system and to increase compatibility among components.

REFERENCES

[1] MRV Chaudran, Component Based Software Engineering, Leiden Institute for Advanced Computer Science, www.win.tue.nl/~wstomv/edu/2ii45/.../Intro_CBSE_SW-ARCH_13.pdf
[2] Roger S Pressman, *Software Engineering: A Practitioner's Approach*, 5th edition, New York McGraw-Hill, **2001**.
[3] X Cai, MR Lyu, KF Wong and R Ko, Component-Based Software Engineering Technologies, Development Frameworks, and Quality Assurance Schemes, *Proceedings of IEEE Seventh Asia-Pacific Conference on Software Engineering*, **2000,** 372-379.
[4] Gaoyan Xie, Decompositional Verification of Component-based Systems - A Hybrid Approach, *Proceedings of the IEEE 19th International Conference on Automated Software Engineering [ASE'04]*, **2004,** 414-417.
[5] Egon Valentini, Gerhard Fliess and Edmund Haselwanter, A Framework for Efficient Contract-based Testing of Software Components, *Proceedings of the IEEE 29th Annual International Computer Software and Applications Conference [COMPSAC'05]*, **2005,** 219-222.
[6] Muhammad Osama Khan, Ahmed Mateen, Ahsan Raza Sattar, Optimal Performance Model Investigation in Component-Based Software Engineering (CBSE), *American Journal of Software Engineering and Applications*, **2013**, 2(6), 141-149.
[7] S Balsamo, M Marzolla and R Mirandola, Efficient Performance Models in Component-Based Software Engineering, *32nd IEEE EUROMICRO Conference on Software Engineering and Advanced Applications*, **2006**, 64-71.
[8] A Kaur, H Monga, and M Kaur, Performance Evaluation of Reusable Software Components. *International Journal of Emerging Technology and Advanced Engineering*, **2012**, 2(4).
[9] AI Khan, MM Alam and UA Khan, Empirical Study of an Improved Component Based Software Development Model using Expert Opinion Technique, *International Journal of Information Technology and Computer Science (IJITCS)*, **2013**, 5(8).
[10] P Kahkipuro, UML-based Performance Modeling Framework for Component-Based Distributed Systems, In *Performance Engineering, State of the Art and Current Trends, Springer-Verlag*, **2001**, 167-184.
[11] A Diaconescu and J Murphy, Automating the Performance Management of Component-Based Enterprise Systems through the Use of Redundancy, *Proceedings of the 20th IEEE/ACM International Conference on Automated Software Engineering*, **2005**, 44-53.
[12] A Bertolino and R Mirandola, CB-SPE Tool: Putting Component-Based Performance Engineering into Practice, *Component-Based Software Engineering Springer Berlin Heidelberg*, **2004**, 233-248.
[13] S Gobel, C Pohl, S Rottger and S Zschaler, The COMQUAD Component Model: Enabling Dynamic Selection Of Implementations by Weaving Non-Functional Aspects, *Proceedings of the 3rd International Conference on Aspect-Oriented Software Development*, **2004**, 74-82.
[14] Sami Beydeda, Research in Testing COTS Components Built-in Testing Approaches, IEEE *3rd ACS/IEEE Conference on Computer Systems and Applications*, **2005**, 101.
[15] Jiang Zheng, Laurie Williams, Brian Robinson and Karen Smiley, Regression Test Selection for Black-box Dynamic Link Library Components, IEEE 2nd *International Workshop on Incorporating COTS Software into Software Systems Tools and Techniques*, **2007**, 9.
[16] Navneet Kaur and Ashima Singh, Generating More Reusable Components while Development A Technique, *International Journal of Innovative Technology and Exploring Engineering*, **2013,** 2 (3).