**IJITKM** Volume 7 • Number 2 Jan– June 2014 pp. 169-172 (ISSN 0973-4414)

# A Review on Optimizing COCOTS model in Component based software engineering approach

Dr. Gundeep Tanwar, Anshula Verma
Department of Computer Science, BRCM college of Engineering and Technology, Bahal
mr.tanwar@gmail.com, anshulaverma4@gmail.com

**Abstract** - *The field of software engineering and software technology is developing very fast.*
*That is, we introduce new concepts, methods, techniques and tools—or change existing ones and emphasize their value. A major turn in software engineering leading to Component ware has dramatically changed the shape of software development. Traditional software development approach is incapable to meet all requirements. CBSE the new paradigm in software development based on the idea of integrating COTS components. Component based software engineering (CBSE) approach that offers inherent benefits in software quality, development productivity and overall system cost.COCOTS model is widely used in CBSE to estimate effort, cost and time. In this paper we will study the comparative analysis of traditional and CBSE approaches using models to measure the development cost, effort and time.*
**Keywords**: COCOTS, COCOMO II, Effort estimation, COTS, scale factors, effort multipliers.

## INTRODUCTION

Today's software systems are becoming more and more complex, large scale, and difficult to control. These things cause production costs to skyrocket and higher risks to move to new technologies. As a result of this there is a demand for a software development paradigm that will lower cost and raise efficiency.
One of the most promising development models is component based software engineering. This model is based on the notion that developers can select appropriate off-the-shelf software components and build them together using well defined software architecture [1].
Component based software engineering is concerned with the rapid assembly of systems from components. These components and frameworks have certified properties; which provide the basis for predicting the properties of systems built from components.
This kind of software approach is very different from the traditional approach in which the software is built from the ground up. Each of these commercial off-the-shelf (COTS) components can be developed by different companies using even different languages and platforms.
If you look at Figure 1 you can see how these COTS can be taken out of a component repository and assembled into a target software system.[2]

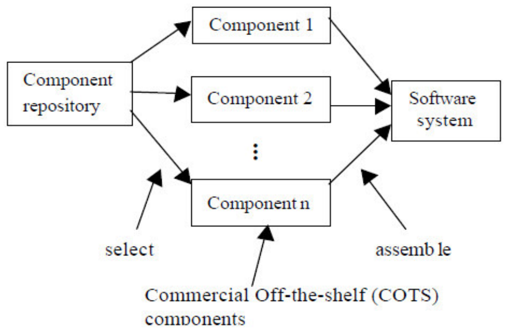

**Figure 1. Component-based software development**

The constructive COTS integration cost model (COCOTS) is an extension of COCOMO II. It is developed for estimating cost of integrating COTS software into the new system. There are two defining characteristics of COTS software in this model:-
The COTS product source code is not available to the application developer.
The future evolution of the COTS product is not under the control of the application developer.
COCOTS effort estimation is made by summing up the resulting effort of these models:-
(1) Candidate COTS component assessment
(2) COTS component tailoring
(3) The development and testing of any integration or "glue" code needed to plug a COTS component into a larger system [3]

## 1. Traditional and component based software development

In traditional approach development approach, we implement the system from scratch, it require large time, cost &effort in s/w development.
CBSE approach more focus on the development of s/y by selecting appropriate reusable components. Development by assembling the pre-existing or reusable components in CBSE approach is helping in increasing productivity, quality and wider range of usability.[4]

**2. Component based lifecycle process model**

Component-based software engineering (CBSE) focuses on building large software systems by integrating previously existing reliable, reusable and robust software components rather than implementation the entire software system from scratch [4]. In CBSE, the notion of building a system by writing code or programming the entire system has been replaced with building a system by assembling and integrating existing software components. In contrast to traditional development, where system integration is often the tail end of an implementation effort. A component-based system (CBS) is integration centric with a focus on assembling individual components, to develop the application. In CBS, component source code information is usually unavailable. Each component introduces properties such as constraints associated with its use, interactions with other components and customizability properties [5].

**3. Stages in CBSE life Cycle**

Component-based software systems are developed by selecting various components and assembling them together rather than programming an overall system from scratch, thus the life cycle of component-based software systems is different from that of the traditional software systems. The life cycle of component based software systems can be summarized as follows [6]

Components selected in accordance to the system requirements.

a) Requirements analysis

b) Software architecture selection, construction, analysis, and evaluation c) Component identification and customization d) System integration e) system testing

f) Software maintenance

**a) Requirements analysis: -** Component requirement analysis is the process of discovering, understanding, documenting, validating and managing the requirements for components.

**b) Software Architecture Selection**:-The objective of this phase is to select the architecture of the component according to the user requirements .In this we will construct the component and that component can be Component off the shell (COTS) component.

**c) Component Identification and Customization:-**

Identification of the component can be done by selecting the right components in accordance to the requirement for both functionality and reliability. Component Customization is the process that involves: -

1) Modifying the component for specific requirement.

2) Doing necessary changes to run the component on special platform.

3) Upgrading the specific component to get a better performance and higher quality.

**d) System Integration:** - It is the process of assembling components selected into a whole system under the designed system architecture. The objective of system integration is the final system Composed of several components.

**e) System Testing: -** System testing is the process of evaluating a system to

i) Confirm that the system satisfies the specified requirements.

ii) Identify and correct the defects in system in system implementation.

**f) System Maintenance:-**It is the process of providing service and a maintenance activity needed to use the software effectively after it has been delivered.

## 4. COCOMO II Model

The COCOMO II model was created to meet the need for a cost model that accounted for future software development practices. The new model new version of COCOMO 81 and ada COCOMO models. COCOMO II is an objective cost model for planning and executing software projects.. The current version of COCOMO II application supports only COCOMO II calculation, which is the estimation of cost, effort, and schedule of the new project.

COCOMO II has three sub models [7]:

1) Application Composition model.
2) The Early Design
3) Post- Architecture models

ALGORITHM used to find out the effort:

```
STEP1->input develop software.
STEP2->Compute value of PMnominal/Effort.
STEP3->Find the size of software KLOC.
STEP4->calculate the effort equation by
        Effort/PMnominal=A (size/KLOC) ^B
STEP5->Analyze the result
STEP6->END [1]
```

## 5. COCOTS Model

The" constructive COTS integration cost model "is an extension of COCOMO 11. COCOMO 11 models does not work there for the estimating of the effort of s/w s/y where it is not able to access the original source code of components. A COCOT is developing for estimating cost of integrating COTS s/w into new s/y.

COCOTS effort estimation is made by summing up the resulting effort of these models:-

(1) candidate COTS component assessment,
(2) COTS component tailoring,
(3) the development and testing of any integration or "glue" code needed to plug a COTS component into a larger system [8].

ALGORITHM used to find out the effort:

```
STEP1-> input develop software.
STEP2-> Component Assessment Effort-
        PAE=IFE+DAE
 Initial Filtering Effort (IFE) = ∑ [(#COTS candidates in class)(average initial filtering effort for class)] over all classes
 Detailed Assessment Effort (DAE) = ∑ [(#COTS candidates in class) (average detailed assessment effort for class)] over all classes, by project domain
STEP3-> Project Tailoring Effort (PTE) = (#COTS tailored in classes) (average tailoring effort for classes and complexity) [9]
STEP4-> Glue Code Effort = A* [(size)(1+CREVOL)] ^B* ∏ (effort multipliers)
 Where
 A = linear scaling constant
 Size = size of the glue code in source lines-of-code or function points
 CREVOL = percentage rework of the glue code due to requirements change or volatility in the COTS products
 B = an architectural nonlinear scaling factor
 Effort multipliers = 13 multiplicative effort adjustment factors with ratings from very low to very high [9]
STEP6->Analyze the result
STEP7->END
```

## 6. Conclusion

In this paper we have discussed about the cost, effort computing models. The earlier COCOMO II model was being design for the cost computing and this is an overview for the COCOTS model. The comparison will be done by optimizing COCOTS model.

## REFERENCES-

[1] Puneet Go swami, Pradeep Kumar , "Effort Estimation in Component Based Software Engineering," International Journal of Information Technology and Knowledge Management, Vol. 2, No. 2, pp. 437-440 (2009).

[2] Xia Cai, Michael R. Lyu, Kam- Fai Wong, Roy Ko, "Component Based Software Engineering: Technologies, Development Frameworks, and Quality Assurance Schemes," In Proceedings of APSEC'2000, pp. 372-379 (2000).

[3] B.W. Boehm, J. R. Brown, Y. Yang, "A Software Product line Life Cycle Cost Estimation Model," IEEE Proceedings of Empirical Software Engineering ISESE'04, 0-7695-2165-7, USA, pp. 156-164, August (2004).

[4] Poom Naunchan, Daricha Sutivong, "Adjustable Cost Estimation Model for COTS-based Development," Preliminary Proceeding of the Australian Software Engineering Conference (ASWEC'07) 0-7695-2778-7, pp. 341-348, April (2007).

[5] Frank Luders, "Use of Component-Based Software Architectures in Industrial Control Systems," Sweden, ISBN number: 91-88834-19-0, pp. 124-127 (2003).

[6] M. Morisio, C.B. Seaman, V.R. Basili and A.T. Parra, "COTS-Based Software Development: Processes and Open issues," In Proceedings of Journal of Systems and Software, USA, Vol. 61,Issue 3, pp. 189-199, April (2002).

[7] M. Vieira, M. Dias, D.J. Richardson, “Describing Dependencies in Component Access Points,” Proceedings of the 4th Workshop on CBSE, 23rd International Conference on Software Eng. (ICSE 2001), Toronto, Canada, pp : 115-118 (2001).

[8] B.W. Boehm, J. R. Brown, Y. Yang, “A Software Product line Life Cycle Cost Estimation Model,” IEEE Proceedings of Empirical Software Engineering ISESE’04, 0-7695-2165-7, USA, pp. 156-164, August (2004).

[9] Chris Abts, Barry W. Boehm and Elizabeth Bailey Clark, “COCOTS: A COTS Software Integration Lifecycle Cost Model-Model Overview and Data Collection Findings,” Technical Report USC-CSE-2000-501, USC Center for Software Engineering, (2000).