A Comparative study of Traditional and Component based software engineering approach using models

Anshula Verma[1,] Dr. Gundeep Tanwar[2]
[1, 2]Department of Computer Science
BRCM college of Engineering and Technology, Bahal
E-mail-anshulaverma4@gmail.com, mr.tanwar@gmail.com

Abstract **-** Traditional software estimation models are directed towards large monolithic software development projects. Contemporary software development practices require a new approach to software cost estimation. 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 have done the empirical and graphical study of both approaches and compared the effort.

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. .[2]

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: - 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 provides the following three-stage series of models for estimation of Application Generator, System Integration and Infrastructure software projects:

1. Application Composition Estimation Model- The earliest phases or spiral cycles will generally involve prototyping, using the Application Composition model capabilities. The COCOMO II Application Composition model supports these phases, and any other prototyping activities occurring later in the life cycle.

2. Early Design Model- This phase will generally involve exploration of architectural alternatives or incremental development strategies. To support these activities, COCOMO II provides an early estimation model called the Early Design model. This model is used when a rough estimate is needed based on incomplete project and product analysis. This model uses Unadjusted Function Points (UFP) as measures of size [2].

PM *nominal* = A*(size)^B

B=0.91+ 0.01*(sum of rating scale factor for the project)

Here PM *nominal*= effort of the project in person month

A= Constant set to 2.5

B= scale factor

Size= software size

- If B= 1.0 means that there is a linear relationship between effort and size.
- If value of B < 1.0 non-linear relationship exist between effort and size and the rate of increase of effort decreases as size of the product increases.
- If B > 1.0, the rate of increases of effort increases as size of the product increases.

The five Scale Factors are:

1. PREC Precedentedness (how novel the project is for the organization)
2. FLEX Development Flexibility
3. RESL Architecture / Risk Resolution
4. TEAM Team Cohesion
5. PMAT Process Maturity

Early design cost drivers are:

- Product Reliability and Complexity (RCPX)
- Required Reuse (RUSE)
- Platform Difficult (PDIF)

- Personnel Capability (PERS)
- Personnel Experience (PREX)
- Facilities (FCIL)
- Schedule (SCED)

3. Post-Architecture Model-This model is used when top level design is complete and detailed information is known about the project. The Post-Architecture model covers the actual development and maintenance of a software product. This stage of the lifecycle proceeds most cost-effectively if software life-cycle architecture has been developed.

$$\text{PM adjusted} = \text{PM nominal} * \prod_{i=1}^{17} \text{EM } i$$

- EM: Effort multiplier which is product of 17 cost drivers

The 17 Post Architecture Cost drivers are mapped to 7 Early Design Cost drivers are given in table below. This mapping is essential because many parameters will not be known correctly n early design phase.

Early design cost drivers	Counterpart combined post Architecture cost drivers
RCPX	RELY,DATA,CPLX,DOCU
RUSE	RUSE
PDIF	TIME,STOR,PVOL
PERS	ACAP,PCAP,PCON
PREX	AEXP,PEXP,LTEX
FCIL	TOOL,SITE
SCED	SCED

Table 1 Mapping Table

We have assumed that

- Glue code Size=60 loc
- CREVOL =10 percent
- Application code =500 KLoc
- SCREVOL= 20 percent
- REVL= 10 percent

COCOMO II Result for Traditional approach is:

E= A *(KLOC)^B
E= 2.5 *(500) ^0.9
=2.5* 268.579
=**760** Person Months

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].

COCOTS result for CBSE is:

Part 1 – Computation of The assessment effort is:

Correctness	Availability/Robustness	Security
Product Performance	Understandability	Easy to use
Version Compatibility	Intercomponent Compatibility	Flexibility
Installation	Portability	Functionality
Price	Maturity	Vendor Support
User Training	Vendor Concession	Reliability

Table 2 Cots Assessment Attribute

- Initial Filtering Effort (IFE) = $\sum_{i=0}^{i=classes}$ [(# COTS candidates in class) (initial filtering effort for class)]

In the initial filtering effort, we first select the cots products for our project based on the rough idea. Here we select 14 cots candidates for inclusion in the system to handle a variety of functions and put these candidate components in two classes.

In IFE we assumed constant "initial filtering effort for class" = 0.2 person-months:

- IFE (person-months) = $\sum_{i=0}^{i=2}$ $14 \cdot 0.2 = 2.8$ (person-months)

Now, we estimate COTS candidates product for the final selection effort .We have to use *rule of thumb* to estimate a percentage of COTS products that make it through initial filtering to final selection. Final selection of COTS products is based on an assessment of each candidate product in the light of certain product attribute.

So, after initial filtering we assumed that there are three component candidates in class 1 and two component candidates in class 2.

- Detailed Assessment Effort (DAE) = $\sum_{i=0}^{i=classes}$ [(# COTS candidates in class) (average detailed assessment effort for class)

In DAE the assumed constant" average detailed assessment effort for class" = 0.5 person- months:

- DAE (person-months) = $\sum_{i=0}^{i=2}$ $5 \cdot 0.5 = 2.5$ (person-months)

Final Project Assessment Effort = IFE + DAE = 2.8 + 2.5 = 5.3

Part 2- Computation of Tailoring Effort is:

Now, we have to determine at what level of complexity each of the selected COTS components to be used in our system will be tailored. After the DAE the components left are one in class 1 and one in class 2. The complexity level for these selected components is low and hence the assumed constant" average tailoring effort for class and complexity" = 0.2 person-months:

- Project Tailoring Effort (PTE) = $\sum_{i=0}^{i=classes}$ [(# COTS tailored in class) (average tailoring effort for class and complexity)]

PTE *(person-months)* = $\sum_{i=0}^{i=2}$ $2 \cdot 0.2 = 0.4$ (person-months)

Part 3- Computation of Glue Code Effort is:

- Glue code effort (GCE) = A* [(size)(1+CREVOL)]^B*∏ (effort multipliers)

Constants assumed:

A = 0.512 person-months per LOC (based on the COCOMO constant A = 3.2 in person-months per kilo LOC for organic development mode)

Constants Assumed

- Size = 60 LOC (glue code)
- CREVOL = 10 percent (% rework of the glue code due to requirements change or volatility in the COTS products
- B = 1.06 (based on the COCOMO constant for organic development mode)
- ∏ (effort multiplier) = 1

GCE *(person-months)* = 0.512*[(60)(1+0.10)]^1.06*1 = 43.45 (person-months)

Part 4 –Computation of System Volatility Effort (SVE) is:

System Volatility Effort (SVE) =(Application effort) *{[1+(SCREVOL/1+REVL)]^E-1}*(COTS effort multipliers)

Constants assumed:

- Application code = 500 KLOC (application code)
- SCREVOL = 20 percent (% rework in the system due to COTS volatility and COTS requirements change)
- REVL = 10 percent (% rework in the system independent of COTS effects due to requirements change
- E = 1.01 + 1.06 (based on the COCOMO Scale Factor)
- ∏ (effort multiplier) = 1

The volatility sub model considers two other kinds of rework: that in the larger system application code due to integration of updated COTS software versions (SCREVOL), and that in the application code independent of COTS product effects. (This latter rework is actually the same as defined within COCOMO itself)To find the application effort in person-months we have the estimation on COCOMO for organic development mode:

- application effort = $a \cdot size^{b}$ = 0.512 · = 371.69 (person-months)

System Volatility Effort =
SVE (person-months) = 371.69 · ([1+ (0.20 / (1+0.10))] 2.07 −1) ·*1 = 525.24 person-months

Computation of Total COTS integration effort is:
The total COTS integration effort =
Assessment Effort + Tailoring Effort + Glue Code Effort + System Volatility Effort
= 5.3 + 0.4 + 43.45 + 525.25 = **587** (person-months)

6. Proposed Model (COCOTS model with system perfective effort)

This is the modified version of COCOTS model by taking into consideration of system perfective effort.

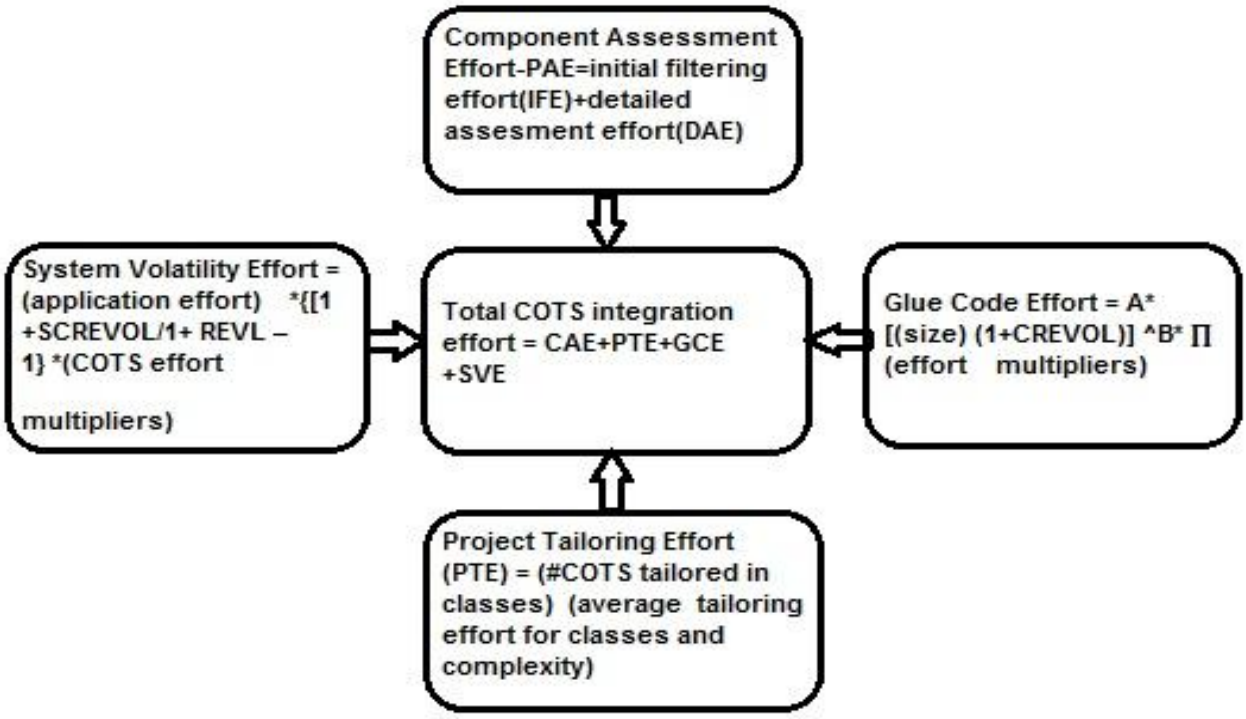

Figure 1 COCOTS Model with System Perfective Effort

7. Results

COCOMO II & COCOTS Effort Table

Sr.No	Application Code (Kloc)	COCOMO II (Person Month)	COCOTS (Person Month)
1	400	549	473
2	500	760	587
3	600	791	701
4	700	909	817
5	800	1025	931

Table 3 Effort table

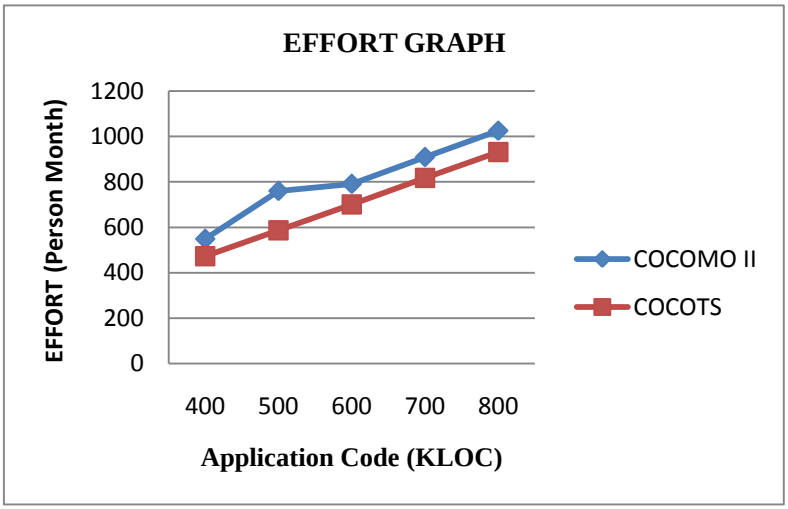

Figure 2 Effort Graph

8. Conclusion

In concern of these two techniques we proposed two models COCOMO II and COCOTS for traditional approach and CBSE respectively. We have done the empirical and graphical study of both models and compared the effort by using Traditional approach and CBSE approach. We have found out that development effort is reduced in CBSE as compared to Traditional approach. Results are quite encouraging and have established the effectiveness of the CBSE in software development. We proved that CBSE is best software development approach in reducing the development effort and time of software under some constraints. The given framework can be used by developer, researcher, Tester ,Engineer and also the user who is deploying the COTS software so that they can use the information for the assessment of cots benefits that they can encounter and thus helpful for increasing the confidence on COTS Software. Results obtained from effort estimation can further be used as input for various other calculations. We also explored the development life cycles of CBSE approach and various benefits of it.

No work when done for the first time is perfect. There is always a scope for improvement. Since it is a new approach in this field so a lot of research is needed. Some of problem I faced

- How to select the no of candidates COTS products that best meet all requirements of the project.
- Development process of component take long time and cost as compare to overall software development process

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 (2012).
[5] 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).
[6] 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).
[7] 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).
[8] 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).
[9] Dr. Gundeep Tanwar, Anshula Verma "A Review on Optimizing COCOTS model in Component based software engineering approach" IJITKM Volume 7 Number 2 Jan– June 2014 pp. 169-172 (ISSN 0973-4414).