International Journal of Engineering Research and Science & Technology

ISSN : 2319-5991
Vol. 5, No. 2
May 2016

www.ijerst.com

Email: editorijerst@gmail.com or editor@ijerst.com

International Journal of Engineering Research and Science & Technology

ISSN 2319-5991 www.ijerst.com
Vol. 5, No. 2, May 2016

Research Paper

DEVISE AND ENLARGEMENT OF COMPONENT ASSORTMENT PROCESS AND NEW OPTIMUM ASSORTMENT MECHANISM IN COMPONENT BASED SOFTWARE ENGINEERING

Vishnuvardhan V[1]*, Santana Lakshmi S[1] and S Gunasekaran[1]

*Corresponding Author: Vishnuvardhan V ✉ vishnu272010@gmail.com

Component Based Software Engineering (CBSE) is a most up-to-date expertise for the development of hefty and difficult software system with the help of using the components of the shelf or reusable components. Software system's competence and production can be improved by reusability approach of CBSE. Some components need to be urbanized individually for the software system and some components are chosen from the third party warehouse. In x-model of CBSE broadly depend upon the best-fit and first fit approach. Component Based Software Development (CBSD) not only decrease the production time to market but also bring down the expenditure of the enlargement profoundly. CBSE is most recent expertise which is primarily goal is to enhance the reusability functionality with enlargement of CBS from the COTS software components according to customer-precise prerequisite and pave way to component assortment process and new proposed algorithm to opt for the optimum component sets for customer-precise necessities which formulate a difficult software system with the composition of optimum reusable software component.

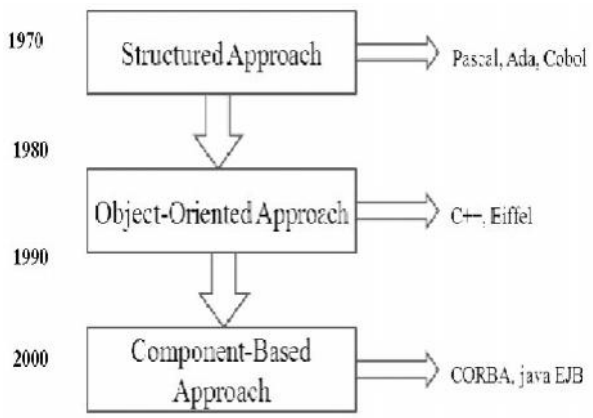
Keywords: COTS, CBD-Component based development, X-model

INTRODUCTION

Evolution of CBSE started in the late 60's in the form of Structure approach (Pascal, Ada, Cobol) then it was replaced by Object Oriented Approach OOA (C++, Eiffel) in 1980, Finally OOA replaced by the Component based approach (CORBA, java EJB) in 2000. Recent trends in CBSD expertise facilitate software reuse that includes enhancement in productivity, dependability, quality, effort, time to market and standards. Systematic reuse adopted as a devise process in order to achieve enhanced software quality, more rapidly, at lesser expenses. These metrics are:

[1] Computer Science Engineering, Coimbatore Institute of Engineering and Technology, Coimbatore, Tamilnadu, India.

Figure 1: Evolution CBSE



1. MTBR (Mean Time Between Failure)
2. MTTR (Mean Time to Repair)
3. ROCOF (Rate of Occurrence of Failure)
4. MTBM (Mean Time Between Maintenance)

ANALYSIS OF LITERATURE

CBSE Process Models

Main process models in CBSE are

1. Stojanovic Process Model
2. COSE Process Model

Stojanovic model (Cai *et al.*, 2002) has lay emphasis on notion of component from Requirement elicitation to execution and verdict of the components should be made to construct components, procure using COTS software components rather than scratch improvement.

COSE Model sort out into four vital phases and integrated with software system phase

1. System decomposition
2. System specification

Component Assortment Progression

A component (Crnkovic *et al.*, 2005) is a nontrivial, almost autonomous, and disposable part of a system that fulfills a clear function in the context of a well defined design. A component can be organized as a black box. In component, surveyor has no awareness of implementation. It has an exterior specification.

Table 1: Component Classification

Component	Characteristics	Examples
Pure computation	Effortless input/output relations, no retained state	Math functions, filters, Renovates
Memory	Shared collection of persistent organized data	Database, Hypertext, file system, symbol table
Manager	State and closely related operations	Abstract data type, many servers
Controller	Manages time series	Scheduler, coordinator
Link	Passes information in entities	User interface, communication bond

CBSE Interface

Components tune-up and amalgamation of operations George and William, 2001) will be governed by the interface. An interface is an integration of operations in which indicates their code of behavior and signature.

CBSE Process

CBSE is moderately related to Object Oriented Technology. CBSE is a parallel occurring process.

1. Domain Engineering
2. Component Based Development (CBD)

Domain Engineering Analysis

Technology used to determine and enlarge subset of software modules. Foremost intend is to widen mechanisms which assist in identification of software components and reuse them foe CBSD. Domain engineering comprises domain investigation, devise and implementation method which assist in recognition and

assortment of precise application domain of CBSD.

Component Based Development

CBSD embraces two processes

1. Amalgamation software product (Dellarocas, 1997) from COTS or reusable components.
2. Developing software reusable component.

CHALLENGES OF COMPONENT ASSORTMENT

Main defy in software reuse is to choose optimum components (Gill and Tomar, 2006) from the component repository in CBSD. Challenges in component assortment comprises of performance, time, component size, error tolerance, dependability, components functionality, compatibility and available component subset for consideration during component assortment.

a) Performance ~1/boundary ~ structure ~1/ coupling

b) Coupling ~1/performance ~ interface ~1/ structure

c) Structure ~ performance ~1/coupling ~1/ interface

d) Time ~1/COTS

e) Component mass ~1/programming language

f) Consistency ~ accessibility

CBSE QUALITY

CBSE is based on three bases

1. Component assortment
2. Price and time
3. Component based software quality

CBSE progression has 8 steps

1. Domain Engineering
2. Software analysis and specifications
3. Making Component repository
4. Optimum component assortment algorithm
5. Composition of component
6. System testing
7. Consumption
8. Preservation

Technique to forecast error prior to the testing stage:

Halstead's software can forecast error with reusability for CBS before testing phase0

Capacity (V) = N $\log_2$ (n1+n2)

Error B = V/S_0

$S_{REU} = S_{RC} * S_{DC} * C_{CC} / * C_C * S_{C\text{-}REU}$

$B = S_{REU} * V/S_0$

Figure 2: Quality Characteristics

S.NO	General Factors for analyzing Reusability of CBS	Impacts on Reusability	
1.	Requirements	Change	↑
		No Change	↓
2.	Design	Change	↑
		No Change	↓
3.	Code	Change	↑
		No Change	↓
4.	Component Complexity	Increase	↓
		Decrease	↑
5.	Software Complexity	Increase	↓
		Decrease	↑

Specifications

$S_{C\text{-}REU\ -}$ Software Complexity for investigation of reusability factor at different levels C_{CC} is a code change in component

S_{DC} – Devise change in software

S_{RC} – Requirement change in software

C_C – Component Complexity

Rationale New-Fangled Component Assortment Algorithm

Begin

Step 1: Select {CS, DR}

//CS = Component set, DR = Domain Repository

Step 2: DR = {1 to n}

//n is the number of components set available in domain repository

Step 3: Sol = Ø

//Sol = elucidation and at current, we have not available optimum component solution according to the client specifications

Step 4: Sum = 0

//sum of software components in the solutions set

Step 5: for I ← 1 to n

//I is the index number of component in domain repository

Step 6: CS ← Select the largest component

Step 7: If UR = CS

//UR = User requirement

Step 8: return scratch data

Step 9: If UR = CS

Step 10: else

Step 11: if UR ≤ CS

Applying these factors for assortment of optimum software component

{performance[1], size[2], reliability[3], Error tolerance[4], Time[5], Complexity[6]}

Step 12: Sol ê CS

Step 13: Sum ê sum + CS

Step 14: Else error "Apply Greedy Algorithm"

Step 15: Return "sol"

Performance[1]

Performance ~ Cohesion 1/Interface ~ 1/Coupling

We know that IDC = #I/#I_{Max} {I = Actual interactions IDC = Interaction Density of a Component, I_{Max} = Maximum available interactions}

So I_{Max} = #I/IDC

Size[2]

LOC ~ 1/High level language

Size ~ 1/High level language

For I ê 1 to n

X ê selects the component set which has written in high level language as possible

Reliability[3]

Availability = MTBF/(MTBF/MTTR) = MTTF/MTBF

Reliability ~ Availability

Error tolerance[4]

Error tolerance ~ Mean time to failure

Time[5]

Time ~ 1/COTS

Complexity[6]

CC = E-N+2P

// CC = Cyclomatic complexity E = number of edges of the graph P = no of connected components

N = No of nodes of the graph

CONCLUSION

The proposed algorithm is executed in c# in .net framework 3.5 with help of visual studio. This new component assortment helps to select the optimum component sets according to customer-precise requirements. These factors save the problem solving effort increase the reliability because each reused component set has been already evaluated and examined. Optimum component assortment process not only improves assortment process but also has as an optimistic impact on the quality and maintainability of software products. The new optimum process supports quality software development with algorithms to select the optimum component according to the user demand even after the completion of domain engineering, software analysis and specification from subset of components.

REFERENCES

1. Cai X, Lyu M and Wong K (2002), "Component-Based Software Engineering: Technologies, Development Frameworks, and Quality Assurance Schemes", Vol. 12, No. 2, pp. 107-133.
2. Crnkovic I (2003), "Component-Based Software Engineering – New Challenges in Software Development", in Proc. 25th Int'l Conference on Information Technology Interfaces June 16-19, pp. 9-18, IEEE Press, Cavtat, Croatia.
3. Crnkovic I, Larsson S and Stafford J (2005), "Component-Based Software Engineering: Building Systems from Components", 9th IEEE Conference and Workshops on Engineering of Computer-Based Systems.
4. Dellarocas C (1997), "The SYNTHESIS Environment for Component-Based Software Development", in Proc. of 8th Int. Workshop on Software Technology and Engineering Practice, London UK.
5. George H T and William C T (2001a), "Component-Based Software Engineering – Putting the Pieces Together", May, Addison Wesley.
6. George H T and William C T (2001b), "Component Based Software Engineering", Publisher Addison Wesley Longman.
7. Gill N S and Tomar P (2006), "Major Challenges Faced in Component-Based Software Reuse", *Journal of Computer Science*, Vol. 2, No. 01, pp. 53-58.
8. Lau K and Taweel F C (2011), "The W Model for Component-Based Software Development".
9. Li J *et al.* (2006), "An Empirical Study of Variations in COTS-Based Software Development Processes in Norwegian IT Industry", *J. Empirical Software Engineering*, Vol. 11, No. 3, pp. 433-461.
10. Voas J (1998), "Certifying Off-the-Shelf Software Components", *IEEE Computer*, Vol. 31, No. 6.

International Journal of Engineering Research and Science & Technology
Hyderabad, INDIA. Ph: +91-09441351700, 09059645577
E-mail: editorijlerst@gmail.com or editor@ijerst.com
Website: www.ijerst.com