International Journal of Electronics Communication and Computer Engineering
Volume 3, Issue 3, ISSN 2249 –071X

Three Pillars for Component-Based Software Engineering

Sandeep Singh
Ph.D., (D.I.T.)
Babasaheb Bhimrao Ambedkar University, Lucknow
er.sandeep_vhdl@yahoo.com

R.A. Khan
Associate Professor (D.I.T.)
Babasaheb Bhimrao Ambedkar University, Lucknow
khanraees@yahoo.com

Abstract **- Component-Based Software Engineering (CBSE) is an approach which is slightly different from traditional software engineering; with the help of CBSE developer can produce software with low cost in a short period of time because CBSE is dependent on the concept of reusability of code. CBSE works as a technology, through which we can design and construct new software with the help of reusable software components according to the customer requirements. CBSE is the combination of Domain Engineering and Component-Based Software Development and to make CBSE process strong and more effective. Developer should have to work on some different pillars like (1) Component Selection (2) Fault Detection and (3) Software Quality Assurance. In this research task we analyze the algorithm for optimal selection of components using seven basic steps of CBSE process. Hence a method has been developed to predict the fault before the testing phase. By applying and using the methods and algorithm mentioned a user can get a quality component based software.**

Keywords **- CBSE, CBSD, Software Quality Assurance, Software Engineering.**

I. Introduction

Component-Based Software Engineering (CBSE) is an approach which is used to enhance the reusability because reusability is a way to improve productivity and efficiency of software systems. CBSE process is a combination of domain engineering and component-based software development (CBSD), and a domain is defined as a set of component which has same properties and Domain engineering is a mechanism which is used to identify and develop a subset of software components that have applicability to existing and future software in a specific application domain or we can say it is used to develop a mechanism which helps in identification of software components and to reuse them for CBSD. CBSE is based on three pillars 1) Component Selection 2) Cost and Time 3) Component-Based software quality.

It is a big challenge to make CBSE three pillars strong, but if we apply some algorithm's to select an optimal component according to customer requirement from the repository then the first pillar would be strong , and we talk about the second pillar then there should be a condition that – as we know that reusability means the reuse of code with modification or without modification as per according to the need of customer, so when the code will modified then we will see the customer requirements again then the design would be prepare again and as like it the code will also written again so if after coding phase we apply the a method which can find the fault in program before testing then there will be a positive effect on cost and time and know when we talk about the last pillar means to ensuring our product on the bases of quality with the help of software quality then we have to follow a software quality model and if we are using ISO-9126 software quality model, then firstly we have to add an attribute with the six main attribute of ISO-9126 because without this seventh attribute it is not sufficient to justify or proof the quality of Component-Based Software, because it is totally reusability of code.

II. Three Pillars

Algorithm for Optimal Selection of Component

CBSE process is a combination of Domain Engineering and CBSD, both run in a parallel manner, so if we want to get a quality product with CBSE then we have to maintain our process from initial step and for that we have to apply an algorithm to select optimal component from the repository which would be comes after the domain engineering and after getting a optimal set of component from the repository according to the customer need, we will compose them to make a product.

The CBSE process has seven basic steps and these are 1) Domain Engineering 2) Software Analysis and Specification 3) Making Component repository 4) Optimal Component selection Algorithm 5) Composition of Component, and by applying this process we can make our

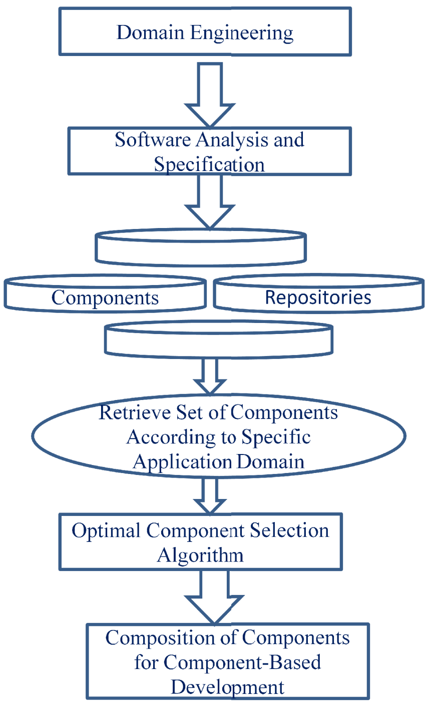
Domain Engineering
Software Analysis and Specification
Components
Repositories
Retrieve Set of Components According to Specific Application Domain
Optimal Component Selection Algorithm
Composition of Components for Component-Based Development

Fig.1.

product according to the customer requirement and the quality of product should be good. The software analysis phase demands the systems analyst to study the application and its constraints, understand the essential features of the system, understand the requirements expected to be satisfied by the software system and create an abstract model of the application in which these requirements are met. Domain engineering is a mechanism which is used to identify and develop a subset of software components and it gives a repository of components, after that we will apply the method on repository to get the optimal set of component.

Method to predict the fault before the testing phase

Halstead's software science can use to predict the fault with reusability for component based software before the testing phase. We are using Halsted Software Science as a base for quality of component based software. A computer program is a collection of tokens according to software science and token can be classified as either operators or operands. In 1972 Halstead model gives an idea of software science Primitives Halsted model are shown below in equation 1 and 2.

Volume - **$(V)=N\log_2(n_1+n_2)$** **...1**

Fault - **$B=V/S$** **...2**

Here n_1 is no of distinct operators in a program, n_2 is no of distinct operands in a program, N_1 is no of operator's occurrences and N_2 is no of operands occurrences. B is the number of faults in program and V is the volume in terms of operators and operands. S_0 is the mean number of mental discriminations (decisions), $S_0 = 3000$ according to Halsted Software Science. Now if the faults and the factors which affect the reusability, detect before the testing phase then it would be beneficial for both developer and customer because developer wants to give the delivery of software with high quality & low bugs and customer wants software with good quality an, low cost, on scheduled time . If these factors can predict in early phases of a software development then it would be possible to decrease the percentage of software crisis. Now to find the faults and the factors which affect the reusability we will use Halsted Software Science equations about fault and volume with addition of a variable for the reusability affecting factors, as shown in Table. 1

$S_{REU}= S_{RC}*S_{DC*}C_{CC}/C_C*S_{C\text{-}REU}$............ 3

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	⬆

Table.2

S_{RC} is *Requirement Change* in software, **S_{DC}** is *Design Change* in software, **$S_{C\text{-}REU}$** is software complexity for analysis of reusability factor at different levels, **C_{CC}** is *Code Change* in Component. To calculate the above variables for **C_C** is *Component Complexity* the factors which have positive effect taken as numerator and the factors which has negative effect taken as denominator to make a variable. And then put this variable in equation 2, through which we can get the maximum number of faults occurred before the testing and also get the value of factors which effect the reusability. After putting the value of equation 3 in 2 we get-

$B=S_{REU}*V/ S_0$……….................................4

This method is very helpful in reducing the risk and cost in our component based development process to remove the obstacle before software testing phase according to the Halsted software science. It would be very beneficial because the testing phase consume more time in compare to other phase and by applying this method researchers and practitioners predict the software reliability and reusability with general factors viz. software requirement, design, code, component complexity, software complexity. So this method will help to make a reliable component based software based system and thus it has great applicability in measuring the quality of component-based software.

III. Quality Component–Based Software

When we talk about the quality of component-based software then we will use some quality models to satisfy it by satisfied the different conditions of different attributes of that software quality model. Like when we talk about the software quality model ISO-9126 then we will find that there is lack of an important attribute according to Quality Component-Based Software, and that attribute is reusability because CBSE is based on the concept of reusability.

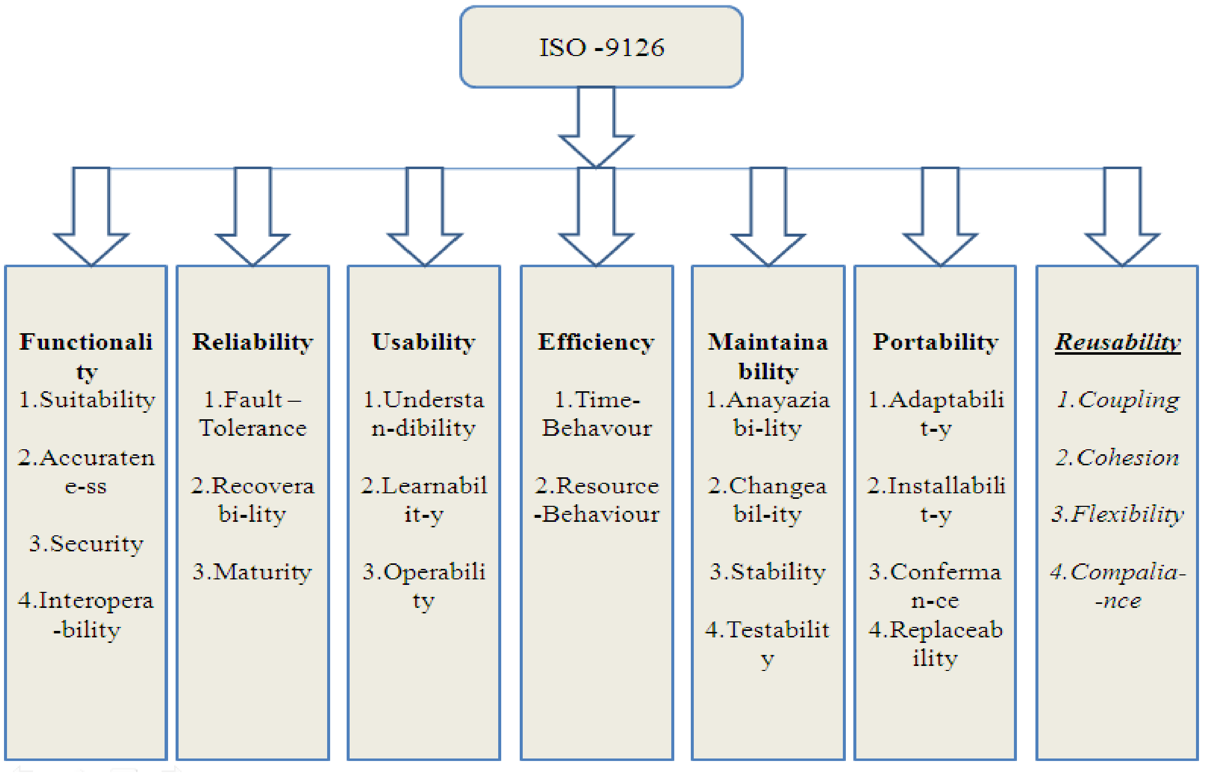
ISO -9126
Functionality 1.Suitability 2.Accurateness 3.Security 4.Interopera-bility
Reliability 1.Fault – Tolerance 2.Recoverabi-lity 3.Maturity
Usability 1.Understan-dibility 2.Learnabilit-y 3.Operability
Efficiency 1.Time-Behavour 2.Resource -Behaviour
Maintainability 1.Anayazia bi-lity 2.Changea bil-ity 3.Stability 4.Testability
Portability 1.Adaptabilit-y 2.Installabilit-y 3.Confermance 4.Replaceability
Reusability 1.Coupling 2.Cohesion 3.Flexibility 4.Compalia--nce

Fig.2. Modified ISO – 9126 Quality Model for CBSE

Quality Component–Based Software

When we talk about the quality of component-based software then we will use some quality models to satisfy it by satisfied the different conditions of different attributes of that software quality model. Like when we talk about the software quality model ISO-9126 then we will find that it not has an important attribute according to Quality Component-Based Software, and that attribute is Reusability because CBSE is based on the concept of reusability. We are using this model because ISO 9126-1 is an extension of previous work done by McCall (1977), Boehm (1978), FURPS and others in defining a set of software quality characteristics. The ISO 9126-1 software quality model identifies seven **main quality characteristics**, namely:

- Functionality
- Reliability
- Usability
- Efficiency
- Maintainability
- Portability
- Reusability

But when we are talking about the CBSE then we have to use one more attribute to use the ISO -9126 as a quality model for the CBSE or for CBSE process to produce a quality product., because the concept of CBSE is totally depend upon the reuse of code because reusability is an important characteristic of a high-quality software component. Programmers want to design and implement software components in such a way that many different programs can reuse them. So we can say that without the reusability attribute the ISO 9126 is not able applicable for CBSE. Now the ISO 9126 model would be a contribution of seven main attributes and there sub attributes like as shown in fig.2

The different factors have different sub factors with their different functionality parameters. We are introducing reusability as a seventh factor with its sub factors and the sub factors are-1) Coupling 2) Cohesion 3) Flexibility 4) Compliance. The sub factors are defined as - **Coupling** *or dependency is the degree to which each program module relies on each one of the other modules*; **Cohesion** *is a measure of how strongly-related and focused the various responsibilities of a software module are.* **Flexibility** is defined as the ease with which a system or component can be modified for use in applications or environments, other than those for which it was specifically designed and the last **Compliance** means the satisfaction with the rules and norms of organization.

IV. Conclusion

The biggest challenges to make CBSE three pillars more strong we used the algorithm for optimal selection of components using all seven basic steps of CBSE process. Halsted software science is used as a base for achieving the quality of component based software. The flexibility and scalability of above discussed method, algorithm and software quality models would be very helpful to make CBSE process effective, and also to reduce the risk, if the process will use these then the quality software will produce with defined cost and in defined time. By applying and using this method, algorithm and model researchers and practitioners can get a quality Component-Based Software.

REFERENCES

[1] Pressman, R. S. "Software Engineering: A Practitioner's Approach", 7th Edition, (2006). McGraw Hill.

[2] Jalote, P. "An Integrated Approach to Software Engineering", 3rd Edition, (2008). Narosa.

[3] Kandt, K. R. "Software Engineering Quality Practice". 1st Edition,(2006), Auerbach Publications

[4] www.testingfaqs.co.in/WaterfallModel.html.

[5] www.cs.colorado.edu/~kena/classes/5828/s07/lectures/04/index.html.

[6] www.tkomarek.com/category/embedded-software.

[7] Mall, R. "Fundamentals of Software Engineering". (2009), PHI Publication.

[8] Xia Cai; Lyu, M.R.; Kam-Fai Wong; Roy Ko. "Component-based software engineering: technologies, development frameworks, and quality assurance schemes". Software Engineering Conference, (2000). pp. 372-379.

[9] Stallinger, F.; Dorling, A.; Rout, T.; Henderson-Sellers, B.; Lefever, B. "Software process improvement for component-based software engineering: an introduction to the OOSPICE project". Euromicro Conference, (2002) pp. 318-322.

[10] Pour, G. "Towards component-based software engineering". Computer Software and Applications Conference, (1998). pp. 599.

[11] Narasimhan, V. L.; and Hendradjaya, L.; "Theoretical Considerations for Software Component Metrics", World Academy of Science, Engineering and Technology, (2009). Volume 10.

[12] George, T.; Heinemann, L.; and William, T.; Council "Component Based Software Engineering" (2001). Publisher: Addison Wesley Longman.

[13] Sharma, A.; Kumar, R,; and Grover, P. S.; "Empirical Evaluation and Critical Review of Complexity Metrics for Software Components", published in the proceedings of the 6th WSEAS International Conference on Software Engineering, Parallel and Distributed Systems, Corfu Island, Greece, (2007). pp. 24-29.

[14] Microsoft Corporation. Definition of the term component; available at http://www.msdn.microsoft.com/repository/OIM/resdkdefinitionofthetermcomponent.asp.

[15] Faisal Siddiqui "Component Based Software Engineering: A look at reusable software components", available at: http://www.smb.uklinux.net/reusability/

[16] Terence Zhao, "Component Based Software Development- An overview", 2007, available at: http://hi.baidu.com/lovelink/blog/item/bed292823f328593f603a699.html

[17] Ropponen, J.; Lyytinen, K "Components of Software Development Risk : How to Address them". IEEE transactions on Software Engineering.,(2000), pp. 98-111.

[18] Lyu, M. R. "Software Reliability Engineering: A Roadmap", proceedings of Future of Software Engineering'07, IEEE Conference, (2007). pp. 153-170.

[19] Kan, S. H.; "Metrics and Models in Software Quality Engineering", 2nd Edition, (2002). Pearson Education.

[20] Sadasivam, S. G.; "Component- Based Technology".1st Edition". (2009), Wiley India Pvt. Ltd.

[21] Chatterjee, S.; "Software Engineering Practice in Computer Science Courses". Software Engineering, (2008). pp. 611.

[22] Sibisi, M.; van Waveren, C.C.; A process framework for customising software quality models" AFRICON, (2007). pp.1-9.

[23] www.cse.dcu.ie/essiscope/sm2/charact.html.

[24] www.cse.dcu.ie/essiscope/sm2/9126ref.html.

[25] Oliveira;, M. F. S.; Redin, R. M.; Carro, L.; da Cunha Lamb, L. and Wanger, F. R. "Software Quality Metrics and their Impact on Embedded Software", proceedings of Model-based Methodology for Pervasive and Embedded Software '08, IEEE Conference, (2008). pp. 68.

[26] Parnas, D. L., "Software Engineering Programs are not Computer Science Programs". Software IEEE Journal, (2002). Vol.16, No.6, pp-19.

[27] Daniel Galin, "Software Quality Assurance", 1st Edition, (2009). Pearson Education.

[28] Narasimhan, V .L and Hendradjaya, B "A New Suite of Metrics for the Integration of Software Components", (2004) http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.88.681

[29] Panwar, D.; Tomar P.; Gill, N. S. and Kumar, A. "New Method to Analyze the Impacts on Reliability and Reusability in Component-Based Software Development", (2011). IMS Manthan.

AUTHOR'S PROFILE

Sandeep Singh

has received his B. Tech. degree in Electronics and Communication Engineering from Chaudhary Charan Singh University, Meerut. He completed his M.Tech. in Wireless Communication & Networks from Gautam Buddha University, Greater Noida. He is a Ph.D. Scholar at Babasaheb Bhimrao Ambedkar University, Lucknow. He has been worked as a faculty and Assistant Center Superintendent (ACS) in confidential department of examination cell at Dr. K. N. Modi Institute of Engineering and Technology, Modinagar, Ghaziabad. He is a life time member of IACSIT, Singapore and member of CSI. He has published several research papers in different national and International conferences and Journals. His research interest includes Ad-hoc wireless networks, Mobile IP, Quality of Service, MPLS based networks and Traffic Engineering.

Dr. Raees Ahmad Khan

has received his doctorate degree (Ph.D.) from Jamia Millia Islamia (A Central University) New Delhi. He is Working with Babasaheb Bhimrao Ambedkar University (A Central University) Lucknow, UP, India, as an Associate Professor & Head in the Department of Information Technology, School for Information Science & Technology since December 2006. He has vast publications over repute journals. His research interest includes Software Quality, Software Security, Software Reliability and Software Quality Metrics.