A Formal Review On Inter Module Compatibility in Component Based Software Engineering

Nidhi, Ms.Suman Lata
Research Scholar,Asstt. Professor,
Software Engineering
Baddi University of Emerging Science and Technology, Baddi, India

Abstract: The component based software engineering (CBSE) is the class of engineering in which software components can be used again on distinct software's. The reusability aspect decreases the implementation cost and maximizes probability of software failure due to incompatibility. The software modules which are joined together must be compatible with each other to form original software. The compatability testing will be based on coupling and cohesion values. The coupling and cohesion values will generate faulty and non- faulty objects which reduce compatability between the software modules.

Keywords **– module compatibility, Component Based Software Engineering(CBSE), cohesion, coupling, compatibility testing.**

I. INTRODUCTION

It is an organized approach to the enlargement, preservation and departure of software. Software is compilation of workable programming code, linked libraries and software documentations. It is the function of computer science all along with mathematics and inventive science [10]. In the existing scenario the software engineering has a definite significance for creating peculiar software.

1.1. Application of software:

- Embedded software
- Web application software
- Product line software
- System software

The modern world can't be run without software. The computer supported systems are frequently used to manage utilities and nationwide infrastructures. As there are no usual restrictions to potential of software, the national and international implementation involves software engineering. Software systems are abstract and intangible. It is completed by software engineer who exercise software engineering values on designing, mounting, testing and assessment of software systems. There are number of different type of software systems that vary from simple embedded systems to complex worldwide information systems. There is want of diverse approaches for all different methods of software engineering. All applications need a software engineering but they all don't need similar software engineering methods.

1.2 Component based software engineering

A component is a standard, moveable, disposable and reclaimable set of well distinct functionality that encircles its execution and commercializing it as higher-level surfaces. A component is a software entity, planned to relate with additional components.
There are 3 views of component:

- Object oriented view
- Conventional view
- Process- related view

The concept of building software from components is not new. A "conventional" plan of composite software systems ever begins with the recognition of system parts, selected subsystems and on inferior level modules, classes, measures etc. However, the current rise in new technologies has considerably increased the possibilities of constructing systems and applications from recyclable components.
Practices has revealed that component supported development involves a logical approach to and focus on module characterstics of software development [11]. Conventional software engineering disciplines must be adjusted to the latest approach, and modern

measures should be developed. Component based software engineering (CBSE) has turned out to be documented as anovel sub – discipline of software engineering.

1.3 Issues in component based software engineering

Trusted components: Nowadays, there is a trend to deliver components in binary form and process of component development is outside the control of computer users. This has raised a question of component trustworthiness. The main issue is that the effects of different degrees of trustworthiness on system attributes are not known.

Components certification: Commonly it has been considered that certification means absolute trustworthiness but actually it only gives the results of test performed and a description of the enviorments in which tests were performed.

Requirement management and component selection: Requirement management is a complex process [8]. A problem with requirement management is that requirements in general are incomplete, not precise and contradictory.

Component configurations: complex system may include many components which, in turn, include other components. In many cases composition of components will be treated as components. As the client will start to work with complicated or complex structures, the difficulties involved with structure design emerge.

Tool support: the purpose of software engineering is to provide practical solutions to practical problems and the existence of appropriate tools is essential for productive CBSE performance. Advanced tools, such as, Visual Basic has proven tremendously successful, but numerous new tools are yet to emerge.

Development models: even though the various existing development models exhibit effective technologies, they have countless indefinite characterstics, they are incomplete, unfinished and complex to employ.

Long term management of component based systems: As component -based systems include sub- systems and components with independent lifecycles; the problem of system evolution becomes significantly more complex.

Reliable systems and CBSE: The use of component based development (CBD) in real time systems and distinct process-Controlling systems, in which the uniformity in requirements or demands are more precise, is a difficult and challenging task [9]. A main difficulty with module based development is the restricted or limited possibility of assuring the quality and extra non-functional features or characterstics of the components and hence our lack of ability to assure explicit system attributes.

II. COMPATIBILITY TESTING

Compatibility testing is a sort of non-technical software testing to check whether your software is capable of running on different hardware, operating systems, applications etc. It assures the compatability of systems with different objects like web browsers, hardware and software platforms, end users, operating systems etc. This type of testing serves to uncover how excellent a system is capable to execute in peculiar enviorments hardware, operating system and other software etc. It moreover test the application or product built with computing environment [10].

The compatibility testing encompasses formation of hardware and software designing and executing test in real environments assuring that application is well-matched with various hardware, operating systems, databases and browser versions. The compatibility testing must assure:

- End user encompasses the similar visual practice regardless of the browsers by which they observe the web application.
- In practicality, the application should perform and reply in the similar manner over various browsers.
- To come to a decision what we require to test we must know what is possible to break.

A. Types of compatability testing

There are two types of compatibility testing:

Forward compatibility testing: this testing facilitates validating the compatability of software among existing or forthcoming versions [8].

Backward compatibility testing: the testing facilitates inspecting that intended application for most recent version is as well as appropriate to existing older version of an environment. Hardware/ software behavior can also be checked by performing this testing with older versions of hardware/software.

B. Some of the most common compatability testing defects

- Distinctness in the user interface with respect to look and feel
- Changes with respect to font size, alignment issues
- Existence of broken tables or frames
- Marked changes in CSS style and color, issues related to scroll bar.

III. LITERATURE REVIEW

Y. Mohana Roopa, et.al (2017), in this paper [1] the self- adaptive software design is developed based on component based software engineering, which focusses on the reusability of software modules to attain enhanced output. The proposed design avails component repository to adjust the changes to the design. The GRASP algorithm is used to optimize the system design and planning. The new outcomes are validated based on real time environments and evaluate the outcomes with different existing and accessible methods.

Umesh Tiwari, et.al (2017), has recommended software testing as an essential part for both traditional or computer based software. In this paper [2], authors have proposed a strategy for component testing in component based development. This testing technique must integrate methods of software testing into predefined and planned sequence of actions**.** The components faults and mistakes are tested using architectural design specifications and its test documentation that includes directive, important internal logics and in-out data and control paths. This has been analyzed that most common errors in component testing are misinterpretation or incorrect requirement, incorrect design or operations, inaccurate initialization of selected data and incorrect representation of an expression.

Bhupender Yadav, et.al (2016), in this paper [3], authors have discussed some existing testing techniques that is used by software engineer in CBSE models. There is lack of tools, methods and strategies that covers the integration testing problems as a whole from defining the integration order to test case selection.

Surendra Mahajan, et.al, (2015), has considered that testing experiments are included in test case prioritization in a request that builds the viability in accomplishing some execution objectives. In this paper [4], authors have developed and proved that there is necessity of Component-based software testing prioritization framework that helps in uncovering more extreme bugs. Genetic Algorithm (GA) had been used with java decoding method in order to improve the quality of software products for delivery. They moreover include a set of prioritization keys to project the planned Component- based software Java framework.

Karambir Singh, et.al, (2015), there is a requirement of detailed and complete information about the module provided by component developers in the form of data sheet. The users of component are not satisfied with available information of components due to which understanding data flow while integrating these components have become a challenge. In this paper [5], authors aim is to uncover the selection of existing component characterstics, repository of components, testing and challenges in science of CBSE.

Neelam Sirohi, et.al, (2013), in this paper [6], this has been seen that only specifications are included in third party components or it can be just black box components. The source codes are not accessible because of black box nature of components subsequently it becomes a really complex task to test black box component. It had been concluded that through advancement of software system the need of software testing has been emerged in turn to make sure the quality, consistency, robustness and functionality of software.

Dhawaleshwar Rao, et.al, (2013), has analyzed that existing systems software are complex and there is need of cost estimation in such complex software's. In this paper [7], an automation tool has been designed that facilitate examining the compatibility. The use of integration testing will also result in increase of overall system cost. The simulation results of proposed scheme shows that overall software cost has been reduced and it provide better integration testing.

J.Bosch, et.al, (2010) large scale software development is complicated, more effort consuming and unpredictable and after decades of software engineering study we still encompasses difficulties organizing and managing the continuously evolving complexity. Three trends are driving an acceleration of the complexity, i.e. software product lines, universal expansion and software ecosystems. In this paper[8], authors comprises the resulting difficulties, organized and controlled around architecture, planning, procedure and organization and the troubles are associated to the effectiveness of software advancement as these companies used integration-centric approaches.

Parnas, David 24, et.al,(2008), in this paper[9], it has been considered that, modularity which refers to consistent decomposition of "software design or plan" that endures complicated software to be manageable or convenient as a method for enhancing the flexibility and comprehensibility of a system whereas enabling the fall in its development time. Within this two system designs are represented, and for each one, uniformly a conventional and unconventional disintegration is represented. The standards used in accomplishing the disintegration are addresssed. The unconventional decomposition, if realized with conventional supposition, that a component consist of two or more subroutines, will be not as much efficient in the majority of cases. A different approach to implementation which does not enclose this result is represented.

IV. CONCLUSION

In this paper, it has been concluded, reusability of components is the key factor. The Component Based Software Engineering provides the flexibility to reuse the software components. The software modules are dynamic in nature which can be used with the other multiple modules. The compatibility is the major problem in component based software engineering (CBSE) which decreases its reliability.

REFERENCES

[1] Y.Mohana Roopa, Dr.A.RamaMohan Reddy, "Cost Optimization Component Selection Approach For Component Based Self-adaptive Architecture Using Component Repository", IEEE, 2017.

[2]Umesh Kumar Tiwari, Santosh Kumar, "Component Level Testing in Component-Based Software Development", International Journal of Innovations & Advancement in Computer Science IJIACS, vol. 6, pp. 69-73, 2017.

[3] Bhupender Yadav, Anshu Yadav, "A Review on Component-Based Software Engineering and Testing", Special issue on international journal of recent advances in engineering and technology (IJRAET), vol. 4, pp. 127-129, 2016.

[4] Surendra Mahajan, Shashank D.Joshi, V. Khanaa, "Component-Based Software System Test Case Prioritization with Genetic Algorithm Decoding Technique Using Java Platform", 2015 International Conference on Computing Communication Control and Automation, vol. 5, pp. 847-851, 2015.

[5] Karambir Singh, P. K. Suri, "Technical Review: Inheritance of Component Based Software Engineering", International Journal of Advanced Research in Computer Science, vol. 6, pp. 97-103, 2015.

[6] Neelam Sirohi, Anshu Parashar, "Component Based System and Testing Techniques", International Journal of Advanced Research in Computer and Communication Engineering, vol. 2, pp. 2378-2383, 2013.

[7] Dhawaleshwar Rao. CH, Nishant Gupta, "COMPATIBILITY ESTIMATION FOR COMPONENT BASED SOFTWARE ENGINEERING", Journal of Global Research in Computer Science, vol. 4, pp. 80-82, 2013.

[8] J. Bosch, P. Bosch-Sijtsema, "From integration to composition: On the impact of software product lines, global development and ecosystems", Journal of Systems and Software, vol. 83, pp. 67–76, 2010.

[9] Parnas, David24, "On the Criteria To Be Used in Decomposing Systems into Modules", 25. Wikipedia: Communications of the ACM26, vol. 12, pp. 1053-1058, 2008.

[10] Leondes, "intelligent systems: technology and applications", CRC Press. ISBN16, 2002.

[11] Ivica Crnkovic, "Component-Based Software Engineering — New Challenges in Software Development", Journal of Computing and Information Technology - CIT 11, vol. 3, pp. 151-161, 2003.

[12] Szyperski C., "Component Software-Beyond Object -Oriented Programming", Addison-Wesley, vol. 3, pp. 132-137, 1998.

[13] Heineman, g., Councill, W., "Component Based Software Engineering, Putting the Pieces Together", Addison Wesley, vol. 3, pp. 142-148, 2001.

[14] Ali Mesbah, Mukul R. Prasad, "Automated Cross-Browser Compatibility Testing", Conference Paper in Proceedings - International Conference on Software Engineering, vol. 4, pp. 181-203, 2011.