Compatibility Testing of Component Based Software Using Particle Swarm Optimization(PSO) Algorithm

Nidhi, Ms.Suman Lata
Graduate Student Researcher, Asstt. Professor
Software Engineering,
Baddi University Of Emerging Sciences & Technology, Baddi, India

Abstract : The software engineering is the field which can process software information for the testing or for other purposes. The component based software engineering is the advance technology in which plug and play based modules are used to build software's. The compatibility is the type of testing which is applied which is applied on the component based software's. In this research work, PSO is applied for the compatibility testing. The software modules which are maximum compatible is selected to design new software. The proposed algorithm is implemented in MATLAB and results are analyzed in terms of certain parameters .

IndexTerms – Component Based Software Engineering(CBSE), Compatibility testing, Particle Swarm Optimization(PSO).

I. INTRODUCTION

Software engineering is all about the production of software, from its initial stage to its final stage. Computer science is related to the software engineering in many aspects. As computer science is used to provide the discipline that uses for the theory basis professions in software engineering. The nature and the complexity of the software system are changing. In the today's world the applications that are used to develop the software system are more complex, because in these applications GUI i.e. graphical user interface and client server architecture is used. Software are distributed among the various system, hence it can works in one or more than none processes. It is also works on the various operating systems [1]. A software engineering is related to all the aspects that are used in the software production. Software is basically a generic term, which is used for organizing the data and instructions that are collected to develop it. The software is broken into the two categories: system software and the application software. The system software is used to manage the hardware components, so that other software or user sees it as a functional unit. It is different from the program in many ways. As in software many things are includes: as it consists of the programs, the complete documentation of that program, the procedure that is use to set up the software and the various operation of the software system. Any program is the subset of the software. As on the other hand, program is the combination of source code and object code. The nature and complexity of the software systems had changed significantly in the last 30 years [2]. The previous applications run on single processor and produce fixed output. But with the advancement in the technology application are having the complex user interface and these applications run on the various systems simultaneous like applications which support client server architecture. As software requirement is increases day by day. So it is necessary to maintain the good quality software. To develop good quality software, software engineering is required. For this, the developer's needs to adopt the software engineering concepts, strategies, and practices to avoid the conflicts that are occur during the development process. Software engineering is an approach to develop, maintain and operate the software. The software development plays a crucial role in software engineering. Many specific techniques are required to develop software [3]. The most common thing in development process is the requirement gathering and customer needs. If a developer fails to complete the needs of the customer than he or she may fails to develop good quality software. Software can be said to of good quality, if it is able to fulfill the needs of the customer. The customer can be satisfied in terms of quality, cost and design of the system software. Component-based software engineering is the branch of software engineering that emphasizes the separation of concerns. These separations are concern with the wide ranging functionality of the software system. Component based software engineering assembles the software products from pre-existing smaller products [4]. These products are known as the components. A component model generally defines a concept of components and rules for their design time composition and is usually accompanied by one or more component technologies, implementing support for composition and interoperation. A software component can be deployed independently and is subject to composition by third parties. The first part of the definition is technical, and states that software components should be "black boxes" to be composed without. It asserts that source code modules do not qualify as software components since they make it possible for the composer to rely on implementation details, thus violating the principle of black box composition. The second part of the definition is more market-oriented, effectively stating that it should be possible to market software components as independent products and those buyers should be able to use them as parts in their own products [5]. Naturally, independent deployment also has technical implications, namely that it must be possible to deploy a single component without any modification, recompilation, or similar of the rest of the systems of which the component is a part. As already mentioned, a software component model specifies standards for composition of and interaction between software components. To facilitate the use of such models, dedicated software tools and infrastructures are often implemented. These may include run-time environments for component execution and interaction as well as tools for component development, composition, and deployment. A software component technology is a set of dedicated software products supporting the use of a specific software component model. 5Heineman and Council use the term component model implementation to denote the run-time parts of a software component technology [6]. A component model is a definition of standards for component implementation,

documentation and deployment. Examples of component models are: EJB model (Enterprise Java Beans), COM+ model (.NET model), Corba Component Model. The component model specifies how interfaces should be defined and the elements that should be included in an interface definition.

II. LITERATURE REVIEW

Roy ko, et.al (2000) has emphasized the problem of the self-optimization and adaptation in the context of adaptive systems is becoming increasingly important with the development of complex software systems and unpredictable execution environments. Here, a general framework for automatically deciding on when and how to adapt a system whenever it deviates from the desired behavior is presented [7]. In this framework, the target system's behavior is described as a high-level policy that establishes goals for a set of performance indicators.

Kung-Kiu Lau et.al (2005) stated that CBSE currently lacks a universally accepted terminology. Existing component models adopt different component definitions and composition operators. We believe that for future research would be very important to clarify and unify the CBSE terminology, and that the starting point for this endeavor should be a study of current component models. In this paper, we take this first step and present and discuss taxonomy of these models [8]. The purpose of this taxonomy is to identify the similarities and differences between them with respect to commonly accepted criteria, in order to clarify and potential unification.

Kuljit kaur chahal, et.al (2008) reviewed on current component-based software development makes use of existing software components to create new applications [9]. But none is complete enough to carry out the assessment. It says users of the components are not necessary to worry about the internal details of the components. However, here it is believed that the complexity of the internal structure of the component can help estimating the effort related to evolution of the component. This research focused on the quality of internal design of a software component and its relationship to the external quality attributes of the component.

Prasanta Bose, et.al (2009) emphasized that the available evidence in a legacy software system, which can help in understanding and architecture recovery are not always enough. Too often, system documentation is poor and outdated. One could argue that the most reliable information is the source code of the system. However, a significant knowledge about the problem domain is required to facilitate the extraction of useful information from the system architecture. In this modeling approach feature is introduced as an additional step in the process of retrieval system architecture [10]. Retrieving information about features and architecture is collected in a tank to allow passage refactoring as follows. The approach is currently applied on a large project for reengineering of a system of industrial image processing.

Sanjay Misra, et.al (2011) discussed about the use of software metrics. Software metrics can be used to improve the quality and productivity of the system. For designing the object oriented code complexity factor can be used. In the object oriented system inheritance is used to check the complexity of the system [11]. The design of the system is difficult and expensive to change. The object oriented software development is expensive due to some features like encapsulation, inheritance, polymorphism and reusability. The complexity of the metrics can be calculated by the traditional metrics. The object oriented metrics do not consider the cognitive characteristics in calculating the code complexity.

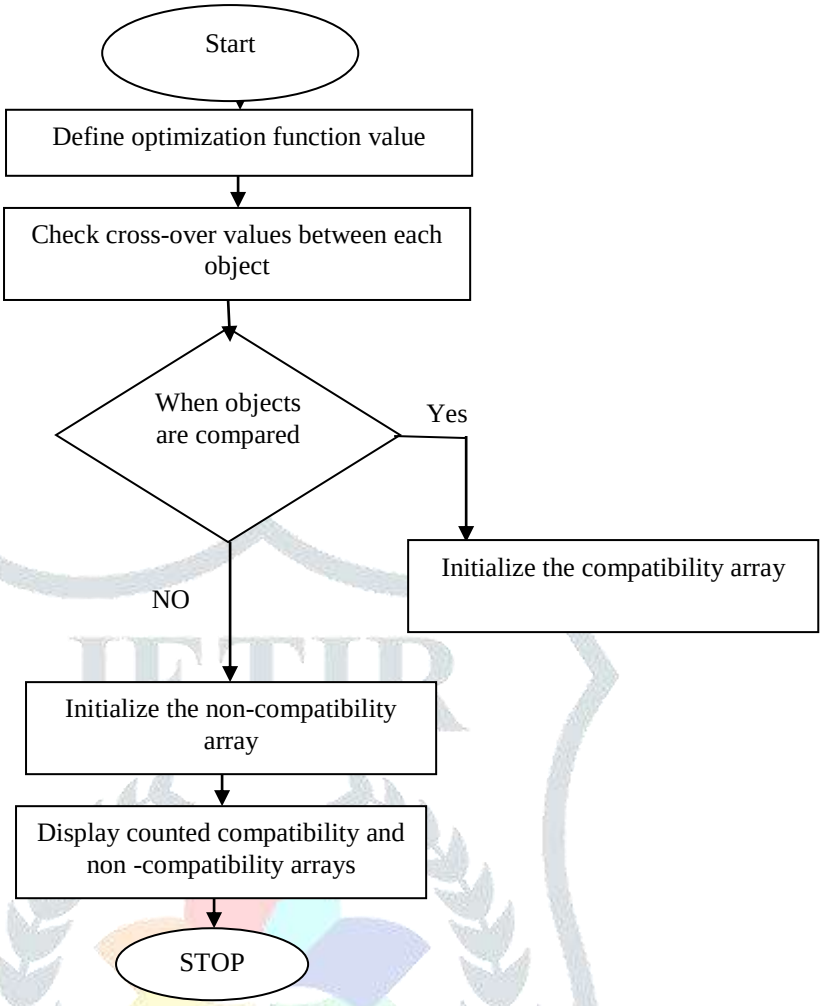
Gilda pour, et.al (1998) presents an approach to the growing demand for rapid and cost effective development of software systems to large-scale, complex, and highly maintainable has introduced a great challenge to the software community. The new trend is to adopt the approach of component-based software development to meet the challenge [12]. The aim of this paper is on the key aspects of the approach of component-based software development and new opportunities and challenges associated with the approach. The article describes the new important activities in the life cycle of component-based systems and discusses the technical and non-technical problems to be solved for the widespread adoption of this approach. The paper also presents several directions for future research in this area.

III. RESEARCH METHODOLOGY

In the CBSE, reengineering is used. Reverse engineering is the process used to analyze the software. It analyzed the software with the objective, which are used to recover its design and specification. In the reverse engineering process, the source code of the existing components is available. The source code acts as the input in reverse engineering. The reverse engineering is different from the reengineering as in the reengineering a new product is produce. This product is highly maintainable. When the modules are reused the concept of plug in and plug out is came into picture. Here to increase the efficiency of the software the modules are integrated. These modules are integrated in that manner so that the compatibility of the system remains the same. When modules of the existing software are integrated to develop new software, there exist objects. Object is basically the instance of anything. The objects may be of two types:

- Faulty objects
- Un faulty objects

The un faulty objects are helps to increase the performance and efficiency of the system. the faulty objects harm the system badly. The faulty objects may create some problem in the system and it may affect the maintainability, scalability and efficiency of the system. When the components are integrated in the system, it makes a class hierarchy. In this hierarchy it maintains the several objects. The PSO is the particle swarm optimization algorithm which can calculate single value from the multiple values. The Particle Swarm optimization algorithm take input the coupling and cohesion value of the module and return the compatability value.



IV. EXPERIMENTAL RESULTS

The projected approach is executed in MATLAB and the outcomes are evaluated by making various comparisons are shown below.

Fig 1: Implementation graph

In this figure, the compatibility among the several components is shown. The computability of the components are shown with PSO Algorithm.

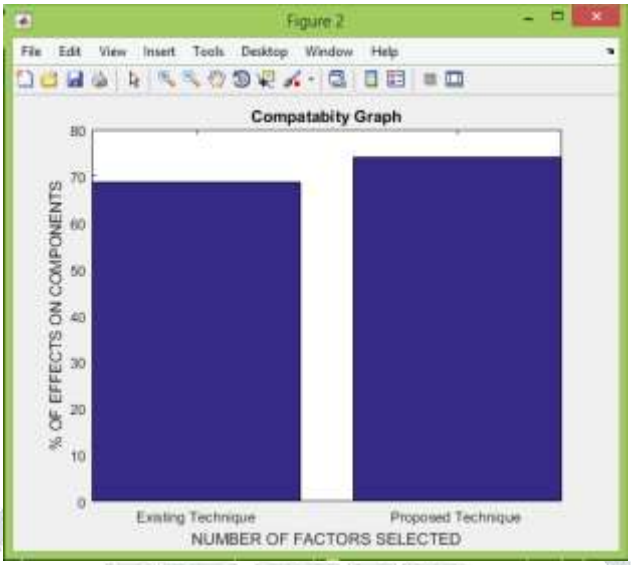


Fig 2: Comparison Graph

As illustrated in figure 2. The two algorithms are compared the knowledge-based learning and PSO for the performance analysis. It is analyzed that PSO performs well in terms of compatability as compared to the knowledge based learning.

V. CONCLUSION

In this work, it is concluded that computability is the major issue of the component-based software's. The selection of most appropriate module is the major issue which is resolved in this research work The PSO is the optimization algorithm which is taken as input and evaluates the compatability of the module. The proposed algorithm is implemented in MATLAB and simulation results shows that it performs effectively in terms of certain parameters.

REFERENCES

[1] Wang Y, "The measurement theory for software engineering", 2003, Proc. Canadian Conference on Electrical and Computer Eng. (CCECE 2003); 1321–132).
[2] Joy B., Steele G., Gosling J., and Brach G., The Java Language Specification (2nd edition), ISBN 0-201-31008-2, Addison-Wesley, 2000.
[3] Lindholm T. and Yelling F., The Java Virtual Machine Specification (2nd edition), ISBN 0201-43294-3, Addison-Wesley, 1999.
[4] Ralph, P., and Wand, Y., "A Proposal for a Formal Definition of the Design Concept", Design Requirements Engineering: A Ten-Year Perspective: Springer-Verlag, 2009, pp. 103-136.
[5] Marinescu R, "Measurement and quality in Object–oriented design", 2005, Proc. 21st IEEE International Conference on Software Maintenance, Beijing, 701–704.
[6] Basci D, Misra S, "Data Complexity Metrics for Web-Services", 2009, Advances in Electrical and Computer Eng. 9(2): 9–15.
[7] Roy ko, Xia Cai, Michael R. Lyu, Kam-Fai Wong "Component-Based Software Engineering Technology, Development frameworks, Quuality Assurance Schemes" 2000, IEEE.
[8] Kung-Kiu Lau and Zheng Wang, "A taxonomy of software Component Models", 2005, IEEE.
[9] Kuljit Kaur Chahal, Harpeep Singh, "A metrics Approach to Evaluate Design of software Components", 2008 IEEE International Conference on Global Software Engineering.
[10] Prasanta Bose, "Scenario-driven Analysis Of Component-Based Software Architecture Model", 2009, Semantic Scholars.
[11] Sanjay Misra, Ibrahim Akman and Murat Koyuncu, "An inheritance complexity metric for object-oriented code, A cognitive approach", 2011, Indian Academy of Sciences, Vol. 36, Part 3, June 2011, pp. 317–337.
[12] G. Pour, "Component-Based Software Development Approach: New Opportunities and Challenges," Proceedings Technology of Object-Oriented Languages, 1998. TOOLS 26, pp. 375-383