ISSN No. 0976-5697

Volume 6, No. 5, May - June 2015

International Journal of Advanced Research in Computer Science

REVIEW ARTICLE

Available Online at www.ijarcs.info

Technical Review: Inheritance of Component Based Software Engineering

Karambir Singh
Department of CSE, UIET
Kurukshtrea University
Kurukshtrea, India-136119

P. K. Suri,
Dean(Academic & Research)
Chairman, CSE/IT deptt.,
HCTM Technical Campus
Kaithal, India

Abstract: Component-based software development has grown rapidly as an approach that for rapid development of system by only using few resources and person-effort. The idea of popularity of component was its reuse and reducing the development costs and it can be achieved if the components offer reliable services and working raw lay architecture is available. Thus, integration of compatible components and their testing become an important part in CBSE and is the understanding of communication, dependency and coordination between the components. Component developers have to provide the detailed information about the component in the form of data sheet but in contrast of this the users of component are not satisfied with information available components. As a result of this, understanding data flow while integrating these components was a challenge. Component-based software as result is the development of complex systems by allowing integration of reusable and simple components. In-house testing of these components was a challenging area of research. There has always been a trouble in integrating the components and getting the optimized reliability as mentioned in the data sheet by the vendor. This in turn affects the quality and reliability of the software. Their research aims at finding the existing component selection, characteristics, repository of components, testing and challenges in science of CBSE. The systematic literature survey was based on 51 international journals collected from multiple-stage selection process.

Keywords: component; component based software engineering; component developer; reliability; quality; repository

I. INTRODUCTION

The historical change of software development of Software Engineering begins from use of traditional software development life cycle (SDLC) but the method of writing the code was structural. Then object oriented approach came into use and made a dramatic change in the development of software system. This approach had many new useful features like abstraction, independency, inheritance, encapsulation and data hiding. This was starting of project management, ease of software development, including testing and differentiating the function by using the concept of class/object. Component based approach; one of the essential characteristics of engineering disciplines is to build a product by ready made assembling, standard components. The component based approach is the most recent trend in industry for development as well as for testing. Right now, component based development (CBD) is reached in the leading edge phase. Indeed, there are now a number of technologies appropriate for the people with experience in the application of CBD.

Component Based Software Engineering (CBSE) is a branch of Software Engineering that emphasizes the separation of concerned service and their abstraction in respect of the wide-ranging functionality available throughout a given software system developing. Software components form of objects or collections of objects may be in some binary or textual form, described by some Interface Description Language (IDL) so that the component may be used autonomously from other components in a while developing software. In other words, a component can be used as it is without changing its source code by using the maximum its functionality.

Even for developing the small system, the developer has to start its development from the zero level or scratch. It used to take more time and had a large cost for it. To cut unnecessary cost and to reduce time of embedding advanced feature in old project Component Based Software Engineering (CBSE) gave it a very high importance. This was attributed to the reduction of cost and time in building the software using reusable components. A component was generally defined as a piece of executable software of some defined function with a published interface. The advantages of CBSE came into picture are: Reduced lead time, enhanced quality, Maintenance of Component-based applications. But one thing that is a gap between vendors and developers are that no sufficient information is provide to the develop about these components like logical or structural implementation. Overall our research mission was to find and scrutinize the current techniques and issues in Testing of components in CBSE. This will be a good starting point in furthering the research. We aim at conducting a systematic literature survey of the state of the art in Integration Testing of components in CBSE. The most crucial aspect for a researcher was to have adequate knowledge of what has been produced in the area of interest. Performing a literature search helps to define an unsolved problem. In this survey different strategies and technique of developing component, its integration, compatibility and testing was studied.

II. REVIEW OF COMPONENT BASED SOFTWARE ENGINEERING

Walid Kobrosly *et al.* [1] proposed several technical articles in the area of software testing and provided a cross-section of software functional testing techniques. Then all different methodologies were found useful, a complete testing effort may need to include different techniques, each to be applied in the appropriate phase of the automation of testing process. Zhenyi Jin and A. Jefferson Offutt [2] proposed the coupling-based testing technique and 12 coverage criteria were defined for testing of whole system. The coupling based technique was also compared with the category partition method on a case study, which found that the coupling-based technique detected more faults with fewer testing cases than category-partition. Harald A.Stieber [3] proposed that if classical testing strategies are used, the application of Software Reliability Growth Models may be difficult to apply and reliability predictions could be misleading. This paper presented an approach which allowed a reliability comparison between different versions or different parts of one project as case study. Elaine J. Weyuker [4] classified truly useful component repositories by considering component design, interfaces, associated directories, storage conventions, and maintenance arrangements. Author proposed that Components should be tested for each new expected or unexpected environment so that developers and users could do better predict their expected behaviour and performance once installed. Only with this investment of time and resources will components and its reusability existing components or using COTS (commercial off the self) components could truly be cost-effective and provide the reliability assurances that many of today's industrial environments require.

Hoijin Yoon and Byoungjn Choi [5] proposed the factor for component customization failures by implementing the inter-class test technique between the black box class, and the white-box class. Their proposed test technique was based on a fault injection technique where a fault was injected into the interface of the component. The author then defined the fault injection operators, which were inserted to the fault injection targets. Since the fault injection operators covered most of possible failures that occurred within component customization, the proposed testing technique was suitable for component customization testing. Jerry Ago *et al.*[6] implemented a Java framework and a systematic approach to support tracking and monitoring software components in component based programs. Moreover, the paper introduced the concept of traceable components, including requirements, design guidelines, and architecture style. The presented results were useful to add systematic component tracking features into the current Java and EJB technology to support software components, embedding third party components while in software maintenance. This solution had several advantages:1) Simple and easy to use for building traceable component with low programming effort. Flexible and configurable to allow system supporters to monitor various component behaviors, including GUI behaviors, performance, errors and interactions. 2) Consistent trace format and lightweight tracking code. 3) Scalable and useful for both in-house and third party components.4) Changeable to fit into different requirements and technologies in organizations.

Gaoyan Xie [7] presented a test model that depicts a generic infrastructure of component based systems and suggested key test elements for this system. This test model was implemented using a Component Interaction Graph (CIG) in which the interactions and the dependence relationships among components were illustrated. By utilizing the CIG, he proposed a family of test adequacy criteria which allowed optimization of the balancing among budget, schedule, and quality requirements typically necessary in software development. The proposed methodology was efficient and effective, as demonstrated by promising results obtained from a case study. The Component-Based Software Engineering Techniques were gaining substantial interesting in developer because of their potential to improve productivity and lower development costs of new software applications, yet satisfying high reliability requirements without much effort. The Eliane Martins *et al.* [8] presented an approach to improve component testability by integrating testing resources into it, and hence obtained a self-testable component. A prototyping tool "Concat", was developed to support the proposed approach. The tool was intended for OO components implemented in C++ . So the preliminary results of an empirical evaluation of the fault detection effectiveness of the proposed testing approach was also discussed.

The Dick Haulet *et. al.* [9] described the concept that how component developers should design and test their components to produce measurements that would be used by system designers to calculate composite system reliability – without implementation and test of the system being designed and this theory also addressed the basic technical problems inherent in certifying components to be released for later use in an arbitrary system. Marlon Vieira and Debra Richardson [10] proposed a technique to analyze dependencies in large component-based systems. Components communicate, share information, and depend on each other in a CBS. By Identifying of the dependencies embedded was the key to checking the semantic integrity of CBS. Therefore, it was important to give more time to research, for scalable and flexible ways to apply dependence analysis over large and complex CBSs. This approach was scalable and gave a broad idea of the system interaction network, thus facilitating analysis of system dependencies.

Hans-Gerhard Gross and Nikolas Mayer [11] described search-based execution-time analysis techniques under the more recent object-oriented and component-based software development paradigms. It was based on inbuilt testing artifacts and on the execution and optimization of an object's invocation history through a genetic algorithm. Valerie Maxville *et al.* [12] outlined the process for selecting and evaluating third party components. Developer always has a hitch of using the third party components by such issues as how to source, select and test components. Application developers need to be confident that they had the most suitable component for their system. This approach was aimed at developers sourcing third party components from external repositories. Some components come with varying levels of documentation. Their process provided a systematic approach for sourcing and selecting components. Automation of the process will save time, allow for a wider field of components to be considered, and gives traceable reasons for any choices made.

Syed A.Ghazi and Moataz A. Ahmed [13] addressed the test configurations generation problem and proposed a GA-based technique as a solution to the problem. Initially, result of the proposed technique was used effectively to generate test configurations. The proposed technique was easy to apply, and overcomes the exponential complexity associated with other techniques. However, an extensive set of experiments, with more challenging problems, was required to have a fair assessment on the technique. Ye Wu, Dai Pan and Mei-Hwa Chen [14] introduced a decomposition verification approach for component-based systems through both formal analysis as model checking and traditional software testing like of Model of checking Driven Black-box Testing Algorithms for Systems with Unspecified Components. The author also presented both LTL and CTL model-checking algorithms for systems with unspecified components. With respect to an LTL (resp. CTL) formula about a system, their algorithms directly designed a condition in terms of communication graphs i.e. witness graphs over the system's unspecified components, and then tested the unspecified components with test-cases generated automatically from the condition.

Jerry Gao *et. al.* [15] designed component-based software using reusable components and proved that the quality of a component-based system is highly dependent on the quality of its components; component quality validation becomes very critical to both component vendors and users. Effectively validating component quality needs adequate test models and testing coverage criteria. Egon Valentine *et al.* [16] proposed that Component based frameworks become more and more state-of-the art but without verifying the components and their interaction it was nearly impossible to build correct and robust systems. Testing of such systems requires a combination of unit and integration tests, and must deal with verifying the contracts that enabled the interaction of components and also presented "Crash It - a test framework" for component-based testing. The main concept of Crash It was the introduction of expandable contract-checkers that verified the communication between a client and a supplier component. These checkers were also able to communicate with each other and with other modules of Crash It. Thus, Crash It allowed to check the state of each component at every time.

Fevzi Belli, Christof J. Budnik [17] introduced an approach for reducing the test costs of user side oriented component testing by identifying and analyzing of manual activities during the test process to enable automation. For modeling the system, state-based or event-based methods were used. For demonstrating the practicability and benefits of the approach, a commercial test tool was augmented by test facilities (as add-ons) which are developed by their group. A prototype of the test environment was already available at that stage. First, the approach was not required the insight into the code of the CBSUT. Second, once the test script was implemented then the CBS was automatically tested. Specifically, the component user cannot specify the adequacy criterion to be used for test case generation. Sami Beeydeda and Friedhofstr.I [18], explained that the testing of component user was different from the test cases employed in testing at the side of component developer. A component built-in testing enabled, according to one of the approaches explained either contained a predetermined set of test cases or the generation, even if conducted on-demand during runtime, solely dependent on parameters which the component user could not influence. As per Aynur Abdurazik and Jeff Offutt [19], software classes exhibit relationships that complicate integration, including method calls, inheritance, and aggregation. During the integration of class and their testing, an order of integration must be followed. The difficulty arises when cyclic dependencies exist - the functionality that was used by the first class to be tested must be tested by creating stubs, an expensive and error-prone operation. This paper described new techniques and algorithms of computing the integration and test orders to solve the CITO problem. New results concluded of improved edge weights that are derived from quantitative coupling measures to more precisely model the cost of stubbing, and the use of weights on nodes, allowing more information.

Maliangli *et al.*[20] introduced an idea of grouped meta-data object, which included descriptive metadata and Operative metadata. Both metadata was divided into several groups. And each group was consisted of several attributes. Each attribute describe related information GMO presented about component being tested implemented by component users. A formal reference model of grouped-metadata was presented to facilitate integration testing. Based on grouped-metadata, a change model was to generate regression testing. Liang Kong *et al.* [21] explored the application of algebraic testing method to software components. A specification language CASOCC was proposed. An automated EJB component testing tool 'CASCAT' was implemented. This method achieved a high fault detecting ability as shown by their preliminary experiment with testing a software component, which confirmed the experiments done by Doong and Frankl. Huaikon *et al.* [22], the concept of Logic Component (LC) was proposed, and a Web application was divided into LCs which was mapped into the actual physical components finally. The automaton to model each component, and use compositions of automata to model component interaction was time consuming. For each component-test-sequence, the author proposed a new automaton using composition of automata. Abstract test cases was generated from the new automata, after mapping the actions to the actual operations and adding the data of test space, the component test cases were generated automatically or semi automatically.

Patricia D. L. Machado *et al.*[23] proposed an integration testing method for component based software . This method was based on the widely used UML (Unified Modeling Language) notation, covered a complete integration testing process at a contractual component level and it was supported by the use of tools. Components and their interfaces were specified by using UML diagrams and OCL (Object Constraint Language) constraints. This method that addressed the main issues in this kind of testing for component-based software with the goal of minimizing costs and maximizing the chances to detect faults. Cost was minimized by reusing development artifacts when appropriate, defining an optimized integration order that made test harness construction easier and integration steps more feasible, and automation of tasks. Chengying Mao [24] proposed a technique to improve component's testability so as to facilitate component's unit testing and regression testing of CBS. Self-checking aspect was embedded to check the invariants which the component obeyed, and tracing aspect was introduced to collect precondition of method execution in component so as to help regression testers to pick out precise subset of test suit. AOP techniques was applied to enhance the capability of facilitating other testing activities such as integration testing. Jinfu Chen *et al.*[25] proposed a testing approach of component security (TACS) based on dynamic monitoring and detecting algorithm CSVD (component security vulnerability detecting), and their case study verified its integrity and nice operability. The shortcoming of TACS was that the algorithm was dependent on the states transition chart of component. First, the detecting algorithm CSVD improved to enhance the detection granularity. Second, some evaluation mechanisms were proposed after detecting and evaluating the security level.

Jiang Zheng *et al* .[26] presented a feasibility case study of the I-BACCI Version 4 process for regression test selection for applications that incorporate DLL components for which source code was not available. Similar to other RTS techniques, when there was a large number of changes in the new component, I-BACCI suggested a retest all regression testing strategy. The results was verified by examining the failure records of retest-all black-box testing. Current tools identified all changes; no failures would have escaped the reduced test suites. The completion of graphic user interface (GUI) software function included complicated human-machine interactions. The function testing methods usually considered interface of software and environment, while ignoring software requirement and function completion process. Wenjing Cao *et al.* [27] proposed testing method based on data flow graph. Based on software requirement and interface, this method organized function testing process with transaction processing and data flow. Comparing to the existing function testing methods, this testing method was more effective for GUI software.

Weiqun Zheng and Gary Bundell [28] introduced a new concept of Contract for Testability and developed a set of important contract oriented concepts i.e. test contract, effectual contract scope, internal/external test contract and presented useful test criteria for effective model-based testability improvement. Jing xian Gu Lei *et al.*[29] presented most of Web applications of multi-tier architectures. The development of web application that had many components, which was early stage of component-based Web application. This paper focused on the kind of web applications and constructed three dependency graphs based on structure relations and message call relations. Then the author improved the path-based integration testing method, proposed an extended MM-path approach and used this approach to find out testing paths of component-based Web application. Fernando Raposo da Camara Silva *et al.* [30] presented an approach to support component testing aiming to reduce the lack of information between component producers and component consumers. Additionally, the approach was covered by a CASE tool integrated in the development environment. A component with known value of reliability , use of a component in several systems increased the chance of errors being detected and strengthened confidence in that component. In this paper, two workflows were presented describing necessary activities conducted by producers to prepare a component tested by third party, and the activities performed by component consumers to elaborate and executed test cases to support the decision of integrating components to a system under development.

Chunyan Hou *et al* .[31] proposed an approach for component-based software reliability analysis combining the advantages of white box with black box approaches to simultaneously address system structures, and software to repair. Their approach applied testing data transformation to bridge the gap between addressing component interactions, and time domain effects. At first the models of component-based software testing process transformed the testing data to build the reliability dataset required by NHPP models. XIA Qiming *et al.* [32] combined the XML Schema outside the component with the XML validates inside the component. The component interface was extended by XML and XACML techniques, and the test-syntax model defined by XML Schema was built. According to XML Schema mutation operator, EXID validates the XML extension interface. This approach made coupling degree between test-scripts, component consistence and decreasing, thus the difficulty of creating and maintaining test-scripts was reduced. The approach was into cross-platform component testing environments.

Zhongsheng Qian [33] described a component automata based approach for generating test cases for Web applications described through a component interaction diagram. A Web application was assumed to be composed of interacting components and each component behavior was described using an automaton. The test generation process was then outlined and some coverage criteria were defined. Dirk Niebuhr and Andreas Rausch [34] implemented an approach of prototype of DAiSI. on of their components and assure good test cases while trading-off the test case execution overhead. Xiao-li LU*et al.*[35]described the features of the component-based software and metamorphic testing (MT) to alleviate the issues. The metamorphic class was used to invoke relevant component to execute test cases and use their metamorphic relations to defect faults. Test cases for the unit test phase were proposed to generate follow-up test cases for the integration test phase. It had potentials to shift the testing effort from the construction of the integration test sets to the development of metamorphic relations. The metamorphic class was invoked by relevant component to execute test cases and use their metamorphic relations to detect faults.

M. Loberbauer *et al.* [36] tested the Comparability of Plug-and-Play Components and introduced a method and a tool for testing the dynamic compatibility of component-based software systems. It was based on Plux.NET, a plug-in platform for plug-and-play composition of .NET applications. They described the Plux Compensability Test Tool (PCTT) and showed how it could be integrated into the Plux composition infrastructure. It generated a test cases according to the PCTM and executed them. Henryk Krawczyk and Adam Rek [37] presented approach to manage relationships between components and their versions. It described methods to ensure reliable and fast communication between them and also presented platforms for building and testing automation of component based applications and explained that how the component based approach could help to speed up the team work. Furqan Naseer *et al.*[38] analyzed the use of metadata in black box testing of a component and constructed some parameters on the bases of which the author evaluated the limitations of the existing approaches.

Ying Jiang *et. al.*[39] demonstrated the effectiveness of their testing approach of the syntactic and semantic specification, it was used to generate test-data through combining the function based and the error-based method. They also found that the efficiency of test data was not same as previous one after mutation and investigated the relationship between the specification elements and test data, such as the protocol specification and integration testing. As a result, the ability of specification definition was enhanced and improved the efficiency of testing method. Martin Rytter and Bo Nørregaard Jørgensen [40] proposed an approach of the required transparency, by moving fault-tolerance concerns into a meta-level. The meta-level provides clients with dynamic fault containers created as a part of reference resolution at runtime. The behavior of a dynamic fault container was dependent solely on a service-provider interface.

Aditya Pratap Singh *et al.*[41] proposed a Reliability Estimation Model for CBS to estimate the reliability through path propagation probability and component impact factor. This model incorporated the idea of path propagation to estimate overall system reliability after integration of components and the contribution of the individual components that were activated during an execution of line of path. Salma Hamza *et al.* [42] proposed object-oriented (OO) metrics to evaluate component-based applications produced with some kind of framework. Indeed, metrics became a standard in OO community. So, they were well-defined, well-known and empirically validated. Thanh-Trung Pham *et al.* [43] validated with the reporting service of a document exchange server, by modelling the reliability, conducting a reliability prediction and sensitivity analyses. Yumei Wu *et al.*[44] described the software reliability prediction models, which received the most attention in software reliability engineering, used the failure data collected in testing phases to predict the failure occurrence in the operational environment. There exists a difficult problem in software reliability modelling that the prediction capability of a model varied with failure data change. Joao M. Franco *et al.*[45] described the quantitative assessment of quality attributes on software architectures allowed to support early decisions in the design phase, certified quality requirements of stakeholders and improved software quality in future architectural changes. This provided the architects prediction and analysed availability constraints on software architectures.

V.B. Singh *et al.*[46] developed an advancement in the internet technology had eased the software development under distributed environment irrespective of geographical locations. The code-changes due to bug fixes, new features and feature improvements for a given time period were used to predict the possible code changes in the software over a long run (potential complexity of code changes. Mudasir Ahmad *et al.* [47] developed the reliability models for the internet of things. Researchers presented a new methodology for estimating hardware and software reliability given uncertain use conditions, to derive probabilistic estimates for overall system reliability. The methodology was applied to illustrative case studies: estimating the impact of temperature variation on the reliability of two component types in a typical networking product: solder joint interconnects and fans. The methodology was then extended to software applications in a networking product, capturing the effects of distinct variables: interaction between hardware and software resource consumption and the delay between software and hardware update. Sathish V. *et al.* [48] provided a Principal Component Analysis (PCA) based approach of failure prediction in industrial robots using event log information. The event logs were collected through remote service set-up from a robot controller. The proposed method reduced the dimensionality of the original data which consist of interrelated events while retaining the variation present in the data. Using PCA and multi variant statistics such as Residuals and contributions charts, that were able to detect abnormal behaviour of event pattern within 30 days before failure.

Yueshen Xu *et al.* [49] improved the prediction accuracy of reliability in a collaborative way. First, researchers estimated the failure probability of each component through two independent models extended from Matrix Factorization. For each service and user and identified the similar neighbour through similarity computation. Carsten Mueller *et al.*[50] proposed a new approach for handling the different complex algorithm. The main idea was to set up a framework for complex algorithms. A component-based framework, IEOCA (Intelligent Evaluation of Complex Algorithms) written in Java for building and studying complex algorithms like genetic algorithms and ant colony optimization was developed. Zhu Chengbang *et al.* [51] proposed a model driven QoS modelling method. This model was two level domain modelling based on MDA and generic QoS meta-model and proposed the implementation of the QoS meta-model and the QoS domain specific abstract model.

III. CONCLUSION

The systematic Literature survey investigated existing component testing techniques, understanding the behavior of components and their interactions. As seen from the year of publication of the articles, it can be understood that the research was rather dull until the year 2001. There were 8 articles published. From the year 2001 research geared up and more than 4 articles were published on an average. There exists a need to establish requirements traceability and behavior of evolving or changing components. The research also points investigate into automation of testing in CBSE. The research in the field of automated testing of components, testing at run-time, reduced time delivery, estimation, reliability of system and approaches to generate test-cases for evolving components would be beneficial for future research. However non-functional aspects in a system composed of components was analyzed and testing has a great potential for future work.

IV. FUTURE WORK

Be The study of exiting testing techniques and model for integrating component-based software systems, research will be as fundamental place to start work on techniques based on contracts, UML, software reliability models and finite state machine. Also, it was interesting to investigate on the issues like time to test, effort to be invested in testing, and the role of metrics in testing and also in component interaction models. Their studies also show evidence of CBSE and testing in the field of embedded systems. There was good chance to start research in the field of improving the reliability and simulation of repository of Commercial off the shelf components(COTS) and apply simulation methods to reduce delivery time and commit for warranty.

The presented ideas of future work in the form of following questions/points. 1) Was CBSE suitable when there are frequently changing requirements (i.e. Agile fashion)? 2) Testing tools in CBSE. 3) What is the effect of Reliability of component on the whole system. 4) Jow the testing can improve reliability of components. 5) Investigation/case study of CBSE in Software industry? 6) How to achieve common component standardization and environmental characteristics?. 7) Moreover , how the Reliability of Component affects the reliability of system and how it can be improved if the reliability of components is fixed. While informal comparisons with other techniques may be described, reporting thorough comparisons with other techniques will also help to plan as a future work.

V. REFERENCES

[1] Walid Kobrosly and Stamatis Vassiliadis, “A Survey of Software Functional Testing Techniques”, pp. 127-134, IEEE pp. 127-134, 1988.
[2] Zhenyi Jin and A. Jefferson Offutt, “ Integration Testing Based on Software Couplings” , pp. 13-23,IEEE, 1995
[3] Harald A. Stieber, “Statistical Quality Control: How to Detect Unreliable Software Components”, pp. 8-12, IEEE 1997.
[4]Elaine J.Weyuker, “Testing Component-Based Software:A Cautionary Tale “, ,pp. 54-59, IEEE
[5] Hoijin Yoon and Byoungju Choi , ” Inter class Test Technique between Black-box-class and White box-class for Component Customization Failures” , pp 162-165, IEEE, 1999.
[6] Jerry Gao, Eugene Y. Zhu, Simon Shim and Lee Chang , “ Monitoring Software components and component-Based Software”, .pp 403-412, IEEE 2000
[7] Ye Wu, Dai Pan and Mei-Hwa Chen, “Techniques for testing component Based Software. Information and software” , pp222-232, IEEE, 2001
[8] Eliane Martins, Cristina Maria Toyota and Rosileny Lie Yanagawa, “Constructing Self Testable software components”, pp151-160, IEEE, 2001.
[9] Dick Hamlet, Dave Mason and Denise Woit, “Theory of Software Reliability Based on Components”, pp361-370, IEEE 2001
[10] Marlon Vieira and Debra Richardson, “Analyzing Dependencies in large component-Based Systems” , Proceedings of the 17th IEEE International Conference on Automated Software Engineering (ASE’02), IEEE, 2002.
[11] Hans-Gerhard Gross and Nikolas Maye, “ Search-based Execution-Time Verification in Object-Oriented and Component-Based Real-Time System Development”, Proceedings of the Eighth IEEE International Workshop on Object-Oriented Real-Time Dependable Systems,Germany, IEEE, 2003.
[12] Valerie Maxville, ChiouPengLam and Jocelyn Armarego , “ Selecting Components: a Process for Context-Driven Evaluation”, Proceedings of the Tenth Asia-Pacific Software Engineering Conference (APSEC’03),Australia. IEEE, 2003.
[13] Jerry Gao and JingshaHe, “Testing Coverage Analysis for Software Component Validation”, ,IEEE, 2005.
[14] Syed A. Ghazi and Moataz A. Ahmed , “Pair-wise Test Coverage Using Genetic Algorithms”, pp.1420-1424, IEEE 2003.
[15] Gaoyan Xie,, “Decompositional Verification of Component-based Systems—A Hybrid Approach”, IEEE 2004.
[16] EgonValentini, Gerhard Fliess and Edmund Haselwanter, “A Framework for Efficient Contract-based Testing of Software Components”, Proceedings of the 29th Annual International Computer Software and Applications Conference (COMPSAC’05), IICM, Austria, IEEE 2005.
[17] Fevzi Belli and Christof J. Budnik, “Towards Self-Testing of Component-Based Software”, Proceedings of the 29th Annual International Computer Software and Applications Conference (COMPSAC’05), IEEE 2005.
[18] Sami Beydeda, “ Research in Testing COTS Components - Built-in Testing Approaches”, IEEE 2005.
[19] Aynor Abdurazik and Jeff Offutt, “ Coupling based Class Integration and Test Order, Information and Software Engineering”, ACM, 2006
[20] Ma Liangli, Wang Houxiang and Li Yongjie , “A Reference Model of Grouped-Metadata Object and a Change Model based on it Appling for Component-based Software Integration Testing”, Proceedings of the 2007 International Conference on Systems Engineering and Modeling, Wuhan, Hubei, China, IEEE 2007.
[21] Liang Kong, Hong Zhu and Bin Zhou, “Automated Testing EJB Components Based on Algebraic Specifications”, 31st Annual International Computer Software and Applications Conference(COMPSAC 2007), IEEE 2007.
[22] Huaikou Miao, Shengbo Chen, Huanzhou Liu and ZhongshengQian, “Workshop on Intelligent Information Technology Application”, School of Computer Engineering and Science, Shanghai University, Shanghai, China, IEEE 2007.
[23] Patricia D. L. Machado, Jorge C. A. Figueiredo, Emerson F. A. Lima, Ana E. V. Barbosa, Helton S.Lima, “ Component-Based Integration Testing from UML Interaction Diagrams”, pp. 2679-2686, IEEE 2007,
[24] Chengying Mao, “AOP-based Testability Improvement for Component-based Software”, 31st annual International Computer Software and Applications Conference(COMPSAC 2007), School of Software, Jiangxi University of Finance and Economics, Nanchang, China, IEEE , 2007.
[25] Jinfu Chen, Yansheng Lu, XiaodongXie and Wei Zhang, “Testing Approach of Component Security Based on Dynamic Monitoring, College of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan City, Hubei Province, China, IEEE 2007 pp. 381-386.
[26] Jiang Zheng, Laurie Williams, Brian Robinson and Karen Smiley, “Regression Test Selection for Black-box Dynamic Link Library Components”, Second International Workshop on Incorporating COTS Software into Software Systems: Tools and Techniques (IWICSS'07), IEEE 2007 .
[27] WenjingCa and ShenghongXu, “ A Software Function Testing Method Based on Data Flow Graph” , IEEE, 2008.
[28] WeiqunZheng and Gary Bundell, “Test by Contract for UML-Based Software Component Testing “, pp 377-382, IEEE 2008.
[29] JingxianGu, Lei XuBaowenXu and Hongji Yang , “An Extended MM-Path Approach to Component-based Web Application Testing”,pp 144-150, IEEE 2008.
[30] Fernando Raposo da Camara Silva, Eduardo Santana de Almeida and Silvio Romero de LemosMeira, “ A Component Testing Approach Supported by a CASE Tool”, IEEE, 2009.
[31] ChunyanHou, Gang Cui, Hongwei Liu and Xiaozong Yang , “Reliability Analysis of Component Software Based on Testing Data Transformation”, .8th IEEE/ACIS International Conference on Computer and Information Science, Harbin, China, pp. 403-412, IEEE 2009.
[32] XIA Qiming, NIE Nan, Yao Junfeng, and HE Keqing, ” XML API-based Test Framework of Extension Interface Using Software Mutation Analysis for Component”, IEEE, 2009.

[33] Zhongsheng Qian (2009), “Testing Component-based Web Applications Using Component Automata”, pp. 455-458 IEEE 2009.
[34] Dirk Niebuhr and Andreas Rausch, “ Guaranteeing Correctness of Component Bindings in Dynamic Adaptive Systems”, 35th Euromicro Conference on Software Engineering and Advanced Applications 2009. Software Systems Engineering, pp. 454-457, IEEE 2009.
[35] M. A. El-Damcese and N. S. Temraz, “ availability and reliability measures for multistate system by using markov reward model” ,RT&A, 2010.
[36] M. Loberbauer, R. Wolfinger, M. Jahn and H. Mossenbock, ” Testing the Composability of Plug-and-Play Components”, pp. 413-418, IEEE, 2010.
[37] HenrykKrawczyk and Adam Rek,” Methodology for developing Web-based applications from reusable components using open-source tools”, Proceedings of the 2nd International Conference on Information Technology, ICIT 2010, pp. 117-119,
IEEE 2010.
[38] FurqanNaseer, ShafiqurRehman and Khalid Hussain, “ Using Meta-data technique for Component based Black Box Testing”, 6th International conference on emerging Technologies (ICET) , pp. 276-281, IEEE 2010.
[39] Ying Jiang, Ying-Na Li and Xiao-Dong Fu, “The Support of Interface Specifications in Black-box Components Testing” 5th International Conference on Frontier of Computer Science and Technology , pp. 305-311, IEEE, 2010
[40] Martin Rytter and Bo NorregaardJorgensen, “Enhancing Net Beans with Transparent Fault Tolerance Using Meta-Level Architecture”. International Journal of Object Technology, vol. 9, no. 5, pages 55–73, 2010
[41] AdityaPratap Singh and PradeepTomar , ” A New Model for Reliability Estimation ofComponent-Based Software,”, pp.1431-1436, IEEE, 2012,.
[42]Salma Hamza, Salah Sadou and Regis Fleurquin , ‘ Measuring Qualities for OSGi Component-Based Applications”, pp. 25-34, IEEE, 2013 ,
[43] Thanh-Trung Pham and Xavier Defago, “ Reliability Prediction for Component-based Software Systems with Architectural-level Fault Tolerance Mechanisms”, pp. 11-20, IEEE, 2013.
[44]Yumei Wu and Risheng Yang, “ Software Reliability Modeling Based on SVM and Virtual Sample”, IEEE, 2013.
[45] Joao M. Franco, Raul Barbosa and Mario Zenha-Rela, “ Availability Evaluation of Software Architectures through Formal Methods”, pp. 282 -287, IEEE, 2014.
[46]V.B. Singh and Meera Sharma , ” Prediction of the complexity of code changes based on number of open bugs, new feature and feature improvement”, pp. 478 – 483, IEEE, 2014.
[47] MudasirAhmad and San Jose ,” Reliability Models for the Internet of Things: A Paradigm Shift”, pp. 52 – 59, IEEE, 2014.
[48] Sathish V., Sudarsan S.D. and Srini Rama Swamy , “ Event Based Robot Prognostics using Principal Component Analysis”, pp. 14 – 18, IEEE, 2014.
[49] YueshenXu, Jianwei Yin, Zizheng Wu, Dongqing He, Yan Tang, “ Reliability Prediction for Service-Oriented System via Matrix Factorization in a Collaborative Way” , pp. 125 – 130, IEEE,2014.
[50] Dr. Carsten Mueller, Andre Hofmeister and Markus Breckner , “ Component-based approach for intelligent evaluation of complex algorithms”, pp. 808 – 811, IEEE, 2014,
[51] Zhu Chengbang, Li Bing, Liu Shufen, “ A Component Quality of Service Modelling Method”, pp. 695-699, IEEE, 2014.