RESEARCH ARTICLE OPEN ACCESS

EVALUATION OF RELIABILITY FOR COMPONENT BASED SOFTWARE USING SCENARIO BASED APPORACH

K.Jagan[1], Dr.K.Subba Rao[2]
1(M.Tech Scholor-CSE, BVRIT-N, and Narsapur)
2 (Professor-CSE, BVRIT-N, and Narsapur)

Abstract:

In the Software Engineering,The reliability model, and a reliability analysis technique are very importent for component-based software. The technique is named as Scenario-Based Reliability Analysis (SBRA). Using scenarios of component interactions, we construct a probabilistic model named Component-Dependency Graph (CDG .The CDG is to represent the architectural dependency between components and possible execution paths. And it is also developed for distributed software systems.

***Index Terms*—Component-based software, component-dependency graphs (CDG), scenario-based reliability analysis (SRBA), software reliability analysis and modeling.**

I. INTRODUCTION

The Component Based Software Engineering is a specialized form of software reuse concerned with building software from existing components (including Commercial Off-The-Shelf components, COTS) by assembling them together in an interoperable manner. Achieving a highly reliable software application is a difficult task, even when high quality, pre-tested, and trusted software components are composed together . As a result, several techniques have emerged to analyze the reliability of component-based applications.
These can be categorized as:
a) System-level reliability estimation. Reliability is estimated for the application as a whole.
b) Component-based reliability estimation. The application reliability is estimated using the reliability of the individual components and their interconnection mechanisms.

The first approach treats the software system as a unit. This approach is not the most suitable for component-based applications because it does not consider compositional properties of systems, and does not accommodate the reliability growth of individual components. The limitations of system-level approaches for component-based applications. As for the second approach, two issues arise. The first is about estimating the reliability of individual components, and the second is about analyzing the reliability of the application by aggregating the reliabilities of constituting components. This paper addresses the second problem, the analysis of the reliability of a component-based system as a function of its constituents. We are concerned with reliability analysis models for systems whose design is substantially based on independent execution scenarios.

This work is motivated by the need to:

i) Study the sensitivity of the application reliability to reliabilities of components and their interfaces. This guides the process of identifying components and interfaces with critical impacts on system reliability. We can analyze the effects of replacing components with new ones, with similar/same interfaces but improved reliability.
ii) Develop a probabilistic technique for reliability analysis which is applicable at the architecture level.. Many reliability analysis techniques use test cases and fault injection. Using scenarios has the advantage of applicability in the early phases of development life cycle.

iii) Analyze the reliability of a component-based application even when the source code is not available (i.e. fault injection and seeding would not be applicable). This is frequently required by systems built of COTS components.

iv) Develop a reliability analysis technique that addresses issues related to the distributed nature of software systems, such as the complexity and hierarchical composition of subsystems. Moreover, physical distribution of components imposes the need for careful treatment of inter-component link reliabilities.

II. BACKGROUND

The Several reliability models and estimation techniques have been proposed to assess the reliability of component-based applications. Gokhale discuss the flexibility offered by discrete-event simulation to analyze component-based applications. Their approach relies on random generation of faults in components using a programmatic procedure which returns the inter-failure arrival time of a given component. The total number of failures is calculated for the application under simulation, and its reliability is estimated. This approach assumes the existence of a control flow graph of a program. The simulation approach assumes failure and repair rates for components, and uses them to generate failures in executing the application. It also assumes constant execution time per component interaction, and ignores failures in component interfaces and links (transition reliabilities). Sanyal et al. introduce Program Dependency Graphs and Fault Propagation Analysis for analytical reliability estimation of component based-applications. The approach is code-based (reverse-engineering) where dependency graphs are generated from source code, which may not be available for off-the-shelf components.

Krishnamurthy assess the reliability of component-based applications using a technique called Component Based Reliability Estimation (CBRE). The approach is based on test information and test cases. For each test case, the execution path is identified. The path reliability is calculated using the reliability of the components assuming a series connection (using the independent failure assumption and perfect interfaces between components). This approach does not consider component interface faults, although they are considerable factors in reliability analysis of component-based software. In an attempt to borrow ideas from the hardware reliability engineering community, a research group at the University of Toronto proposes that designers should follow certain disciplines in software component development. They develop a set of design and interaction rules to minimize the interaction and dependence between components. The proposed rules facilitate modeling the component-based system as Markov chains and, hence, we can apply the same reliability analysis techniques as in the case of hardware systems. A similar study of the failure dependency between components is discussed. However, such discipline is difficult to impose in practice.

The techniques discussed above are so called path-based approaches to reliability analysis of component-based software. Most path-based approaches assume that components fail independently, thus providing a pessimistic estimate of system reliability. Gokhale propose a technique which allows handling of dependent failures. The solution takes into account time-dependent representation of component reliability, and a fixed execution time per interaction. Everett describes an approach which uses the Extended Execution Time (EET) reliability growth model to arrive at a composite reliability growth model for the testing period. The proposed CDG model and SBRA algorithm could be integrated into the "superimpose component reliability" step in Everett's framework.

Schneidewind addresses the problem of assessing the reliability of distributed systems in which a set of client and server nodes remotely communicate with each other over a network. Physical distribution imposes the need to consider new factors in reliability analysis of software systems:

a) accounting for a possibility that the survivability of certain client and server components will be more critical to the system operation than others, and

b) accounting for the possibility of using redundant services across the network to allow system recovery if one of the critical nodes fails.

Goseva-Popstojanova and Trivedi present a classification of architecture-based approaches to reliability assessment of component-based software systems. They identify three classes based on the methods used to describe the architecture, and aggregate the failure behavior of components and connectors. These classes are:

a) state-based where software architectures and failure behavior are represented as a Markov chain or a semi-Markov process;

b) path-based where reliability is estimated for set of execution scenarios [33], [35]; and

c) additive models which focus on estimating the time-dependent failure intensity of the system using components failure data.

III.COMPONENT DEPENDENCY GRAPH (CDG)

The Control flow graphs are the classical method of revealing the structure, decision points, and branches in program code. We adapt the control flow graph principles to component-based applications to represent the architectural dependency between components and possible execution paths. We call this graph a Component Dependency Graph, CDG. In this section we define the graph, while the following section describes how to calculate graph attributes.

Definition 1: A Component Dependency Grap‹h "CDG": A CDG is defined by the tuple‹N,E,s,t›, whereN,E is a directed graph, is the start node,t is a termination node. N is a set of nodes in the graph,N ={n_i} , i = 1…|N|, and E is a set of directed edges in the graph, E ={e_i} , i = 1…|E|.

Definition 2: A Node "n ": n ϵ Nmodels a component C_i, and is defined by the tuple‹NC_i,RC_i,EC_i›,where:NC_i is the name of component , is the reliability of component C_i, andEC_i average execution time of the component C_i .

Definition 3: Component Reliability "RC_i ":RC_i is the probability that the component C_i executes correctly (failure free) upon invocation.

Definition 4: Average Execution Time of a Component "EC_i ":EC_i is the average execution time of a component C_i .

Definition 5: A Directed Edge " e ":e ϵ E models the control flow transfer from one component to another. It is annotated by the tuple‹T_{ij},RT_{ij},PT_{ij}› where:T_{ij} is the transition name from noden_i ton_j , denoted‹n_i, n_j›, is the transition's reliability, andPT_{ij} is the transitio‹n's execution probability.

Definition 6: Transition Reliability "RT_{ij} ": RT_{ij} Is the probability that information sent from component C_i to C_j component is delivered error-free. This probability includes possible interface errors and possible channel delivery errors, as discussed in Section IV-A-3.

Definition 7: Transition Probability " PT_{ij}": PT_{ij} the conditional probability C_j will execute next given that $C_i$currently executing. The sum of outgoing transition probabilities from any node must be unity.

Thus, a CDG is defined as follows:

CDG = ‹N, E, s, t›,

N = { n }, E = { e }, s and t are the start

and termination nodes

n = ‹NC_i, RC_i, EC_i›,

e = ‹T_{ij}, RT_{ij},PT_{ij}›,

$T_{ij} = \langle n_i, n_j \rangle$

Fig. 1 shows the CDG of a system consisting of four components.

IV.SCENARIO BASED RELIABILITY ANALYSIS (SBRA)

We propose to analyze the reliability of a component-based software application in three steps:

a) estimation of the parameters used in the reliability model;

b) construction of the component dependency graph;

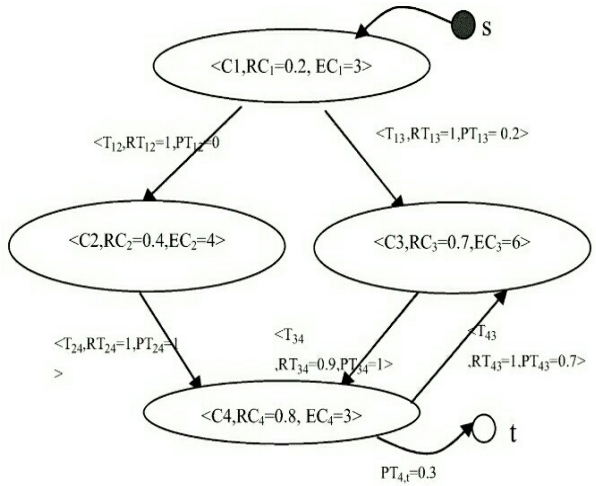

Fig. 1. A sample CDG

c) application of the algorithm for reliability and sensitivity analysis

V .CONCLUSION:

The reliability model, and a reliability analysis technique are very importent for component-based software. The technique is named as Scenario-Based Reliability Analysis (SBRA). Using scenarios of component interactions, we construct a probabilistic model named Component-Dependency Graph (CDG .The CDG is to represent the architectural dependency between components and possible execution paths. And it is also developed for distributed software systems.

This paper defines a new approach to analyze the reliability of component-based, and distributed software systems.. Component Dependency Graphs serve as reliability models for component-based systems. In the future enhancement,We define the Parameter Estmation,CDG Construction and Reliability Analysis Algorothm

VI.REFERENCES

1] A. Alayed. Component-based Approach to Software Product Line Engineering. A First Year Continuation Report Submitted to The University of Manchester, Manchester, UK, 2012.

2] K.-K. Lau, V. Ukis, P. Velasco, and Z. Wang. A component model for separation of control flow from computation in component-based systems. Electronic Notes in Theoretical Computer Science, 163:57-69, 2006.

3] K.-K. Lau, P. Velasco Elizondo, and Z. Wang. Exogenous connectors for software components. In Proc. 8th Int. SIGSOFT Symp. on Component-based Software Engineering, LNCS 3489, pages 90-106, 2005

4]. A. Wesslen et al., “Assessing the sensitivity of usage profile changes in test planning,” in Proc. 11th Int. Symp. Software Reliability Engineering (ISSRE 2000), San Jose, California, Oct. 2000, pp. 317–326.

5]. A. Whittaker, K. Rekab, and M. Thomason, “A Markov chain model for predicting the reliability of multi-build software,” J. Inform. Software Technol., vol. 42, no. 12, pp. 889–894, Sept. 2000.

6]. K. Goseva-Popstojanova and K. Trivedi, "Architecture-based approach to reliability assessment of software systems," Performance Evaluation, an International Journal, vol. 45, pp. 179–204, 2001

7]. S. Yacoub and H. Ammar, "A methodology for architectural-level risk analysis," IEEE Trans. Software Eng., vol. 28, pp. 529–547, June 2002.

8]. Cortellessa et al., "Early reliability assessment of UML based software models," in 3rd Int. Workshop on Software Performance, Rome, Italy, July 2002, pp. 302–309.

9] H. Singh et al., "A Bayesian approach to reliability prediction and assessment of component based systems," in 12th IEEE Int. Symp. Software Reliability Engineering (ISSRE '01), Hong Kong, Nov. 2001, pp. 12–21.

10] H. Jin and P. Santhanam, "An approach to higher reliability using software components," in 12th IEEE Int. Symp. Software Reliability Engineering (ISSRE '01), Hong Kong, Nov. 2001, pp. 1–11.

11] D. Hamlet et al., "Theory of software reliability based on components," in 23rd Int. Conf. Software Engineering, Toronto, Canada, May 2001.

12].Colin Atkinson, Joachim Bayer, Christian Bunse, Erik Kamsties, Oliver Laitenberger, Roland Laqua, Dirk Muthig, Barbara Paech, Jürgen Wüst, and Jörg Zettel. Component-based product line engineering with UML. AddisonWesley Longman Publishing Co., Inc., Boston, MA, USA, 2002.

13] Colin Atkinson, Joachim Bayer, and Dirk Muthig. Component-based product line development: the kobra approach. In Proceedings of the first conference on Software product lines : experience and research directions: experience and research directions, pages 289-309, Norwell, MA, USA, 2000. Kluwer Academic Publishers

14] K.-K. Lau and M. Ornaghi. Control encapsulation: A calculus for exogenous composition. In G. Lewis, I. Poernomo, and C. Hofmeister, editors, Proc. 12th Int. Symp. on Component-based Software Engineering, LNCS 5582, pages 121-139. Springer-Verlag, 2009.

15] Ivica Crnkovic. Component-based software engineering new challenges in software development. Focus Review, Software Focus, 2 2002

16] Clemens Szyperski. Component Software: Beyond Object-Oriented Programming. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2nd edition, 2002.

17] V. Cortellessa *et al.*, "Early reliability assessment of UML based soft-ware models," in *3rd Int. Workshop on Software Performance*, Rome, Italy, July 2002, pp. 302–309.