Journal of Software Engineering and Applications, **2011, 4, 181-186**
doi:10.4236/jsea.2011.43020 Published Online March 2011 (http://www.SciRP.org/journal/jsea)

Scientific Research

Software Reliability Early Prediction in Architectural Design Phase: Overview and Limitations

Lalit Kumar Singh[1], Anil Kumar Tripathi[2], Gopika Vinod[3]

[1]Department of Atomic Energy, Nuclear Power Corporation of India Limited, Mumbai, India; [2]Department of Computer Science and Engineering, Institute of Technology-Banaras Hindu University, Varanasi, India; [3]Department of Atomic Energy, Bhabha Atomic Research Center, Mumbai, India.
Email: lalit.rs.cse@itbhu.ac.in, lksingh@npcil.co.in, aktripathi.cse@itbhu.ac.in, vgopika@barc.gov.in

Received January 31st, 2011; revised March 7th, 2011; accepted March 10th, 2011.

ABSTRACT

In recent times, computer based systems are frequently used for protection and control in the various industries viz Nuclear, Electrical, Mechanical, Civil, Electronics, Medical, etc. From the operating experience of those computer based systems, it has been found that the failure of which can lead to the severe damage to equipments or environmental harm. The culprit of this accident is nobody other than our software, whose reliability has not been ensured in those conditions. Also for real time system, throughput of the system and average response time are very important constructs/ metrics of reliability. Moreover neither of the software reliability model is available which can be fitted generically for all kinds of software. So, we can ensure reliability at the early stage i.e. during design phase by architecturing the software in a better way. The objective of this paper is to provide an overview of the state-of-the-art research in the area of architecture-based software reliability analysis. We then describe the shortcomings and the limiting assump- tions underlying the prevalent research. We also propose various approaches which have the potential to address the existing limitations

Keywords*: *Software Reliability, Software Engineering, DTMC, Early Prediction

1. Introduction

The Control and Instrumentation systems are widely based on software in today's era and software is the root cause of most of today's system problems. It should be noticed that failures never occur if the software is not used but at the same time without software we can't get comfort and comfort is the necessity of everybody. The quality and longevity of a software system is determined by its architecture. Software architecture is a "first cut" at solving the problem and designing the system. The importance of the software architecture lies in earliest design decisions, addressing first design artifact & key to systematic use. Analysis of a system at the architectural level enables the choice of the right architecture for the system under consideration, thus saving major potential modifications later in the development cycle or tuning the system after deployment.

Reliability of the safety critical & safety related systems are of prime importances, which are computer based systems. So, the software that are used in these systems must be reliable, for which software professionals follow so many standards and get software verified by Independent Verification and Validation team before deploying it on the site. But during Verification & Validation, it is impossible to generate all the possible test cases and it may possible that in real life a test case might has been executed, which we had not thought during testing. So, we can never claim that my software is error free—in fact no software in this world is error free.

For this reason our software experts/scientists starts thinking to ensure the reliability of the software especially when those are going to be deployed in a safety critical systems. So, they had given/proposed so many software reliability models to ensure the reliability of the software. But unfortunately till date no model fits in all kinds of software, means there is no generic model that can be used to predict the reliability of all kinds of software. These models are analytically derived from assumptions. The emphasis is on developing the model, the interpreta-

tion of the model assumptions, and the physical meaning of the parameters. These types of reliability models are known as "software reliability growth models". [Fenton & Pfleeger] These models attempt to statically correlate defect detection data with known functions such as an exponential function. If the correlation is good, the known function can be used to predict the future behavior. The reasons that these reliability models lack the needed strength to excel in eliminating errors in software environment are:

1) The misconception of fault & failure phenomena [1]
2) Inaccurate modeling parameters
3) Difficulty in selecting the reliability models
4) Difficulty in building the software operational profile

So, different approach has been started [2-12] *i.e.* to predict the reliability based on the design parameters of software. The models based on this concept are known as "defect density" models. These models use code characteristics such as lines of code, design characteristics such as calculate the weight of a class or architectural characteristics.

The layout of the paper is organized as follows: Section 2 provides an overview of the existing techniques. It also describes the similarities and differences among those approaches. Section 3 summarizes the limitations of those approaches. Section 4 describes the proposed solutions to address the stated limitations and Section 5 summarizes the paper.

2. Overview

Software systems are developed to achieve the requirements in an automated manner. Well, need not to say that requirements can be functional as well as non-functional. The critical prerequisite for a system's success is the exhibited non-functional quality, *i.e.* how a system does. Every software is composed of one or more software modules. Scientists/Researchers proposed many approaches. These approaches are more or less similar in the sense that quantitative methods for reliability assessment depend on the availability of system usage information—*i.e.* a system's operational profile. The operational profile information is combined with the non-probabilistic behavior models in order to obtain probabilistic models which can be used for reliability analysis. Error propagation can also take place so modeling approach to analyze the impact of error prorogation on reliability is also done [2].

So, these approaches have the common model as represented in **Figure 1**.

But the approach to generate Probabilistic Models and to estimate the software reliability differs from one approach to other approach [3]. Probabilistic behavior of a software system is often modeled and analyzed with discrete-time Markov chains. DTMC [4] consists of a set of states; each state has corresponding probabilities of transitioning to other states. DTMCs embody a basic way of modeling, and reasoning about probabilistic behavior; further demands, including the need to more faithfully represent software systems, enabled a proliferation of probabilistic automata formalisms [5]. As an example DTMC modeling of Robot has been shown in **Figure 2**. Using DTMC modeling there are two main paradigms for modeling probabilistic behaviors: the generative, which can generate some actions and the reactive paradigm, which can react to the external actions. In this case calculating the system-wide reliability is equivalent to computing the steady state probabilities of DTMC. To illustrate this approach, consider DTMC model of an example Robot, which is depicted in **Figure 2**. The architecture consists of *Controller*, *Sensor*, *Follower*, and *GUI* components. Several difficulties arise when deriving transition proba-

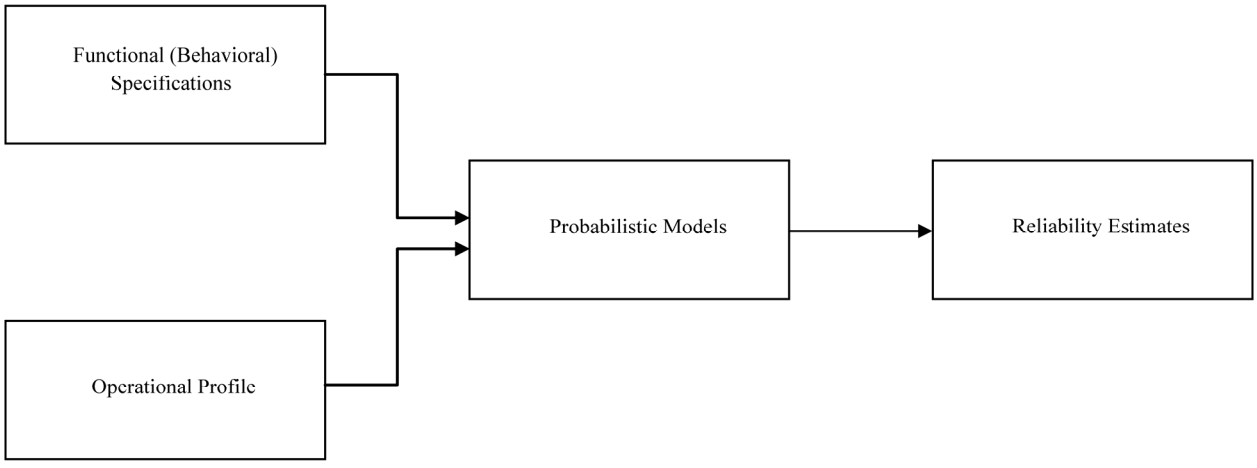

Figure 1. Existing model for reliability prediction based on architecture.

bilities from an operational profile:

1) GUI can invoke different Controller's operations depending on the navigation model.

2) GUI and Follower may be concurrent to each other.

Some researchers also proposed Probabilistic model, based on the control flow graph as shown in **Figure 3**, where $p_{1,2}$ is the probability of going from node 1 to node 2. The enumeration of paths could be conducted algorithmically, experimentally or by simulation. The reliability of each path is obtained as a product of the reliabilities of the components along path. For example application software shown in **Figure 3**, possible execution path is 1, 3, 5, 6 and its reliability is obtained by R_1, R_3, R_5, R_6. The application reliability can be estimated by averaging path reliabilities. But the main drawback is that this approach doesn't give accurate reliability due to looping effects, like- in the path 1, 2, 4, $2^{1\cdots*}$, 6. The subpath 2, 4, 2 can occur infinite number of times. The classification of architecture-based software reliability models can be understood by **Figure 4** [6].

3. Limitations

In this section we discuss the limitations of the prevalent state-based architecture-based analysis techniques. The limitations of the existing approaches can be classified into—1) modeling, 2) analysis 3) parameter estimation 4) validation and 5) optimization. Usually modeling limitations are due to the assumptions we made to ensure model expansibility, which may lead to unreliable estimation. Analysis limitations are due to lacking of analysis techniques. Parameter Estimation limitations are due to non consideration of different software artifacts. Validation limitations are due to paying little effort. Optimization limitations are due to non consideration of complex interactions between components in the architectural design.

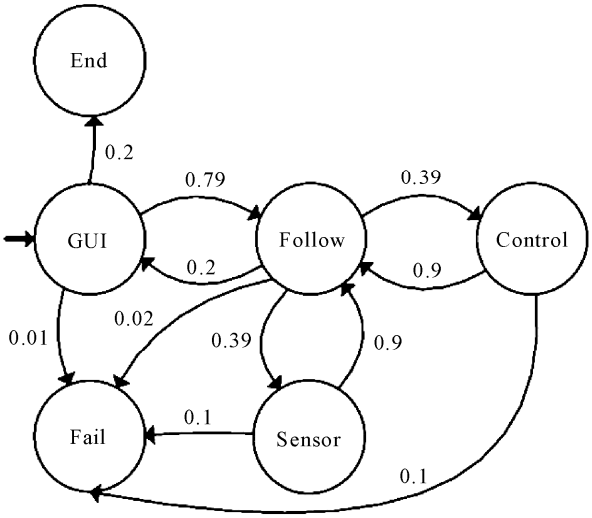

Figure 2. DTMC model of an example Robot.

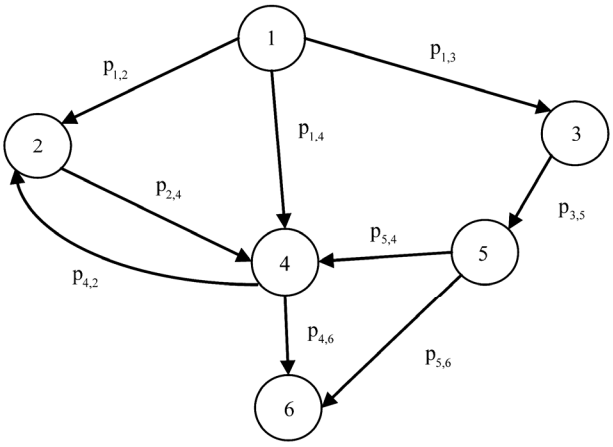

Figure 3. Probabilistic control flow graph of an example application software.

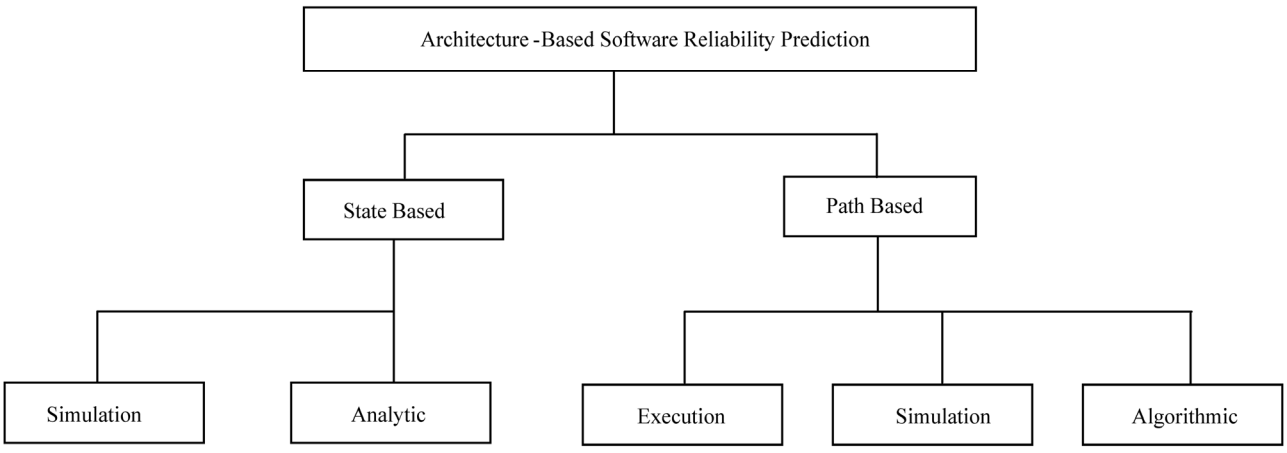

Figure 4. Classification of architecture-based software reliability models.

3.1. Modeling Limitations

1) A practice in which an engineer combines an operational profile and a non probabilistic specifications to directly produce an analysis-enabled generative model is tedious, non-intuitive and error prone.

2) The operational profile information that the existing approaches assume available is often just a subset of the available information.

3) Support for discovery and modeling of error states is not clear or accurate

4) Existing DTMC based models assume that at a time the application can be in one state only which is not valid for today's complex systems.

5) Also the failure of one component can pass on its impact on other components as well which has not been taken care in anyone of the approaches.

6) None of the approach has taken into consideration, the nature of the interface between the components, as there is a possibility that components may be distributed with the advanced technologies like RMI, RPC, etc.

7) The architectural style of different components of the same application software may be different, for which we suspect to fit the common reliability approach on all the components. Also dynamically *i.e.* when application operates its architecture also changes dynamically.

3.2. Analysis Limitations

In some approaches Hidden Markov model has been used when it is difficult to have the surety of next probabilistic transition. But in this case the transition matrix and observation probability matrix (which represents the probability of observing event in a particular state) has been initialized randomly, which may not be accurate.

3.3. Parameter Estimation Limitations

The system-level model can be analyzed using traditional DTMC analysis [7], whose complexity is $O(n^3)$, where n is the number of states. Generally large complex software systems have thousands of states, which could be very expensive to solve the DTMC model.

3.4. Validation Limitations

The less effort has been paid to validate the predicted reliability, based on architectural design with the estimated reliability, just before product release to ensure the correctness of the predicted methodology so that, it can be applied to the future projects.

3.5. Optimization Limitations

Reliability prediction based on architecture can optimized if we success to optimize the software architecture. Sometimes it is noticed that architects design the software in a complex manner, full of tight coupling and low cohesion, which is a poor quality attributes of an architecture [8]. For resource allocation optimization see [9]

So, in this section we have seen the overview of the existing approaches to predict the software reliability in architecture phase. We have also compared the approaches, identified the similarities and differences. We also identified the various limitations which needs to address.

4. Proposed Solutions to Address Limitations

We have seen many limitations of the existing approaches which are crucial for reliability prediction. We proposed the solutions to address the above limitations.

As we have seen the two major problems when we illustrated DTMC model of an example Robot (**Figure 2**) [10,11]. The solution can be sought in more intuitive way by:

1) Specifying systems behavior in terms of scenarios for which use case diagram can help a lot [12].

2) Try to analyze the transition probabilities between different scenarios. Sequence diagrams, swimlane diagrams, collaboration diagram can help in this regard

3) Specify the component failure probabilities. The above limitations, based on classification, can be addressed with the help of the following respective approaches:

4.1. Modeling Limitations

1) We feel that an operational profile and non probabilistic specifications can be more intuitively combined by handling output actions and those actions which are controlled by other components or external environments. Output actions are basically controlled actions. So we can ease the generation of analysis-enabled generative model as well as it will be less likely to be error prone

2) Operational profile, a most important construct to create DTMC model. Hybrid information must be taken into consideration for deriving the operational profile. Though, it is impossible to identify accurate operational profile but by considering various hybrid sources of information, more near operational profile can be identified. The suggested information sources are: i) Domain expert; ii) Similar components/software that are already being used somewhere iii) simulation, mandatory in case of new type of application; can be used for existing type as well; iv) Requirements specifications; v) architectural models of the system at the highest level of detail; vi) data log of the existing running applications. [13]

3) Support for discovery and modeling of error states is not clear or accurate. The main cause could be the inconsistencies in the architectural design. For which, mapping of interaction protocols, dynamic behavior and static behavior can serve the purpose [14].

4) The solution is given above in 3 steps at the begin-

ning of this section.

5) To represent the architecture of concurrent applications, a high-level specification mechanism, such as a Stochastic Reward Net [15], may be used.

6) The methodology proposed by Littlewood to use the constant failure rates to represent interface failures can be used.

7) There are various architectural styles available viz client-server, layered, centric, etc. Also a single application can have many components which again may have different architectural style. For this we propose the solution to convert into layered architectural style because it has been seen that almost every module can be designed by layered architecture. Then we can make the use of Jalote's method to solve the same.

4.2. Analysis Limitations

Architectural models can be used to derive the transition matrix and observation probability more intuitively.

4.3. Parameter Estimation Limitations

After generating the DTMC model of the while application and if the number of states exceeds 100, we shall try to break the DTMC model into DTMC submodels and can solve the individual submodels. The final solution can be sought by integrating the solution of individual submodels. The major issue of breaking can be done in the following steps: 1) be cautious to decide which states should be kept in one group. We must try the most coupled (to each other) states in a single group unless the coupled states are larger than 100; 2) properly define the interface states which has to be introduced between breaked states; 3) solve each DTMC submodels traditionally; 4) Again be cautious while integrating the solutions, have to exclude the effect of the interface states, as being introduced in 2)

4.4. Validation Limitations

The results obtained from evaluation process may be validated conceptually, which is not quantitative approach. This can be done by consulting with the stake-holders.

4.5. Optimization Limitations

To simplify the architectural design, the architects can make the use of Data Flow Diagrams through which simple architectural design can be derived by identifying the transaction center and transformation center [Pressman].

5. Conclusions

In this paper we first introduced the need of the early prediction of the software reliability. We then provided an overview of the existing approaches of software reliability early prediction. We also describe the communalities and differences of those approaches. We also stated various limitations of the existing approaches. We then proposed approaches that could help in addressing the stated limitations. Note that we only proposed the approaches, while providing the exact solutions using those approaches may needs separate papers for each limitation; which we plan to take it as a part of our future work.

REFERENCES

[1] S. Gokhale, W. E. Wong, K. S. Trivedi and J. R. Horgan, "An Analytic Approach to Architecture-Based Software Reliability Prediction," *Proceedings of International Performance and Dependability Symposium*, Durham, September 1998, pp. 13-22.

[2] V. Cortellessa and V. A. Grassi, "A Modeling Approach to Analyze the Impact of Error Propagation on Reliability of Component-Based Systems," *Proceedings of the Component-Based Software Engineering*, Medford, Vol. 4608, 9-11 July 2007, pp. 140-156. doi:10.1007/978-3-540-73551-9_10

[3] I. Krka, L. Golubchik and N. Medvidovic, "Probabilistic Automata for Architecture-Based Reliability Assessment," QUOVADIS' 10, Cape Town.

[4] W. J. Stewart, "Numerical Solution of Markov Chains," CRC Press, USA, 1991.

[5] L. Cheung *et al.*, "Early Prediction of Software Component Reliability," *ACM/IEEE* 30*th International Conference on Software Engineering*, Leipzig, 10-18 May 2008, pp. 111-128.

[6] S. Gokhale, "Architecture-Based Software Reliability Analysis: Overview and Limitations," *IEEE Transactions on Dependable and Secure Computing*, Vol. 4, No. 1, Jan 2007, pp. 32-40. doi:10.1109/TDSC.2007.4

[7] S. S. Gokhale and K. S. Trivedi, "Reliability Prediction and Sensitivity Analysis Based on Software Architecture," *Proceedings of International Symposium on Software Reliability Engineering*, Annapolis, November 2002, pp. 64-75.

[8] Pressman, "Software Engineering, A Practitioner's Approach," 6th Edition, McGraw-Hill International Edition, New York.

[9] J. Lo, S. Kuo, M. R. Lyu and C. Huang, "Optimal Resource Allocation and Reliability Analysis for Component-Based Software Applications," *Proceedings of* 26*th Annual International Computer Software and Applications Conference*, Oxford, August 2002, pp. 7-12.

[10] B. Boehm, J. Bhuta, D. Garlan, E. Gradman, L. G. Huang, A. Lam, R. Madachy, N. Medvidovic, K. Meyer, S. Meyers, G. Perez, K. Reinholtz, R. Roshandel and N. Rouquette, "Using Empirical Testbeds to Accelerate Technology Maturity and Transition: The SCRover Experience," *Proceedings of the* 2004 *International Symposium on Empirical Software Engineering*, IEEE Computer Society, Washington DC, 19-20 August 2004, pp. 117-126. doi:10.1109/ISESE.2004.1334899

[11] B. Littlewood, "A Semi-Markov Model for Markov

Structured Software," *International Conference on Reliable Software*, New York, Vol. 10, No. 6, June 1975, pp. 204-207.

[12] S. Yacoub, B. Cukic and H. H. Ammar, "A Scenario Based Reliability Analysis Approach for Component Based Software," *IEEE Transactions on Reliability*, Vol. 53, No. 4, December 2004, pp. 465-480.

[13] J. D. Musa, "Operational Profiles in Software-Reliability Engineering," *IEEE Software*, Vol. 10, No. 2, 1993. doi:10.1109/52.199724

[14] K. S. Trivedi, "Analytical Models for Architectural-Based Software Reliability Prediction: A Unification Framework," *IEEE Transactions on Reliability*, Vol. 55, No. 4, December 2006, pp. 578-590.

[15] A. Puliafito, *et al.*, "The Evolution of Stochastic Petri-nets," *Proceedings of World Congress Systems Simulation*, Singapore, September 1997, pp. 3-15.

