A Comparative Analysis of Various Existing Software Reliability and Availability Prediction Methods

Bindhyachal Kumar Singh, Arun Solanki & Abdur Rahman

School of Information and Communication Technology
Gautam Buddha University
Greater Noida-201310
Uttar Pradesh, India.
Email: bindhyachalsiwan@gmail.com

Abstract

Software reliability engineering is emphasizing on engineering techniques for developing and maintaining software systems whose reliability and availability can be quantitatively predict and measured. Software reliability models are very useful to estimate the probability of the software fail along the time. It can be used to predict the reliability of a system or the numbers of latent defects of the software product when it is deliver to the customers. The number of defects recovered during various life cycle stages of development models for a project conforms to a numerical distribution. In this paper, we take review of various reliability prediction methods, which have been successfully used for predicting reliability and availability of a software system and show a comparative analysis of different existing prediction models. This paper also provides a framework for comparisons of software reliability and availability prediction methods.

Keywords: Reliability Prediction Model, Software Reliability Engineering, Safety, Maintainability, Availability

1. INTRODUCTION

Software systems are increasingly entering consumers' everyday life. These systems are often highly complicated and distributed to different platforms over wired or wireless networks. To satisfy the consumers' needs, these systems must demonstrate high reliability and availability; thus, they must function correctly and without interruption. In complex software systems, reliability is the most important aspect of software quality. Software reliability assessment is thus a key technology for reducing software costs and producing highly reliable software. A quantitative measurement of software reliability is important for managing software development; it is needed in order to assess software performance and to minimize development and maintenance costs. Reliability is defined as the probability of the failure-free operation of a software system for a specified period of time in a specified environment [1]. Availability is measured as the probability of a software service or system being available when needed. Reliability and availability are often defined as attributes of dependability, which is "the ability to deliver service that can justifiably be trusted [2].

Several measures are traditionally used for reliability and availability, such as mean time to

failure (MTTF), mean time to repair (MTTR) and failure rate.

Mean Time between Failure (MTBF) = MTTF+MTTR

Architecture can be defined as the system structure comprising the components, their externally visible properties and their relationships among each other [3]. Reliability and availability prediction from the architectural descriptions is a challenging task for two main reasons:

I. Reliability is strongly dependent on how the system will be used. Since reliability and availability are execution qualities, the impact of faults on reliability differs depending on how the system is used, i.e. how often the faulty part of the system is executed.

II. The reliability of software architecture depends on the reliability of individual components, component interactions, and the execution environment. The reliability of a component depends on its internal capabilities, e.g. implementation technology, size, and complexity, information about which might be unavailable, or not yet exist, while architecting.

Several analysis/prediction methods have been developed during recent decades for different types of purposes and by different communities. Consequently, they have different definitions and measures for reliability, architecture, inputs, outputs, notations, assumptions, users, etc.

2. A FRAMEWORK FOR RELIABILITY AND AVAILABILITY PREDICTION METHODS

The framework provides in this paper can be used as a basis for our method comparison. The given framework reveals the characteristics required for the analysis methods to be implemented for the evaluation of their suitability for architecture level prediction. The elements of the categories give rise to some specific requirements for the reliability and availability analysis methods that are applicable at the architecture level. The reliability prediction of a component is problematic, as it is affected by several factors, such as the component's implementation technology and configuration. Due to large-scale requirements, and rather often three-layer characterization and a highly complicated distribution to several platforms, the architecture modelling of software systems is challenging.

Table 1: Framework for Reliability and Availability Analysis

S. No.	Type	Elements	Questions
1.	**User**	1. Resources 2. Target Group 3. Skills Required 4. Expected Gains 5. Expected Losses	What skills are required for using the method? What are the gains of using the method? Who is the intended user of the method? How much target assigned for Reliability & Availability? What are the losses of using these methods?
2.	**Valid -ation**	1. Accuracy of prediction 2. R & A. requireme	Does the method define how to trace requirements into the architecture?

		nts Traceability 3. Process maturity levels of the method	Is it Analyze the Accuracy of prediction How close are the expected values to the actual values?
3.	**Environment**	1. Scope of applicability 2. Architecture-specificity 3. Goal	Is the method applicable to the different layers of S/W Is the method limited to application domain? What is the goal of the analysis method?
4.	**Stuffing**	1. Variability 2. System usage 3. Language 4. Tool support	Does the method provide a special model with which the analysis is Performed? Is the variation of architecture considered in the analysis? What notation is used in architecture descriptions? Are there any tools that support the method?

3. COMPARATIVE ANALYSIS OF R & A PREDICTION METHOD

In this section, we compare various software reliability and availability approaches who predict these tow parameters of quality assurance.

3.1 Criteria for Comparison

We define the criteria for the selection of approaches for analysis of prediction methods and concentrate only on the methods that fulfil these criteria. Also introduce a number of different methods and approaches on reliability and availability prediction and compare the most interesting and promising ones based on the framework. According to these criteria, an analysis approach has to

- Concentrate only on software Reliability and Availability
- Take the user centric approach on software product and its analysis.
- Provide a clear applicable analysis method
- Be based on architectural models

Operational profiles and usage profiles, which are usually abstractions from component execution details, are commonly modelled as Markov chains [5], [6]. Markov chains are generally Finite State Machines (FSMs) that are extended with inter-module transition probabilities as the user profile. Use cases and scenarios are a means for requirements engineering to capture system requirements [8]. Use cases are used to define usage scenarios for different system users, thereby defining the external requirements on system capabilities. A scenario is a brief description of a single interaction of a stakeholder with a system. While use cases focus on runtime behaviour with the stakeholder as the user, scenarios also encompass other interactions with the system, such as a maintainer carrying out modification. Message Sequence Charts (MSC) and Sequence Diagrams (SD) are widely accepted notations for scenario-based specifications.

3.2 Comparison of the Selected Methods

We select a various methods and compare those methods in the aspect of software reliability and prediction. Cheung [5] provides a user-oriented software reliability model, according to which the reliability of a system can be computed as a function of both the deterministic properties of the structure of a system and the stochastic properties of the utilization and failure of its components. Basically, Cheung's model is a Markov reliability model that models the composite structure of a system as a control flow graph of a program. The approach of Reussner et al. [1] assumes that components rely on other components of the environment, which furthermore use the properties of underlying hardware. The aim of this method is to predict component reliability through the c*ompositional analysis* of usage profiles and the reliability of environment components. The *path-based approaches*, such as [14], [13] and [15] focus on running the software for various inputs. For each run, the resulting execution path is specified in terms of sequences involving components and connectors. The reliability of the software is a weighted average of the reliabilities of all the paths [26]. The Krishnamurthy and Mathur model [14] estimates the reliability of a sequence of components executed in each test run and subsequently calculates the average of all the test runs. Gokhale and Trivedi [15] also propose a path-based approach to architecture-based software reliability prediction, removing the assumption of independence among the successive executions of the components by proposing a solution based on the failure intensities of components. The model presented in [13] is a *scenario-based* probabilistic model, which is applicable for high-level designs. This model is specific to component-based software whose analysis is based on execution scenarios. The Scenario- Based Reliability Analysis (SBRA) method provides the Component Dependency Graph (CDG) model, which represents in turn the components, component reliabilities, link and interface reliabilities, transitions, and transition probabilities. Rodrigues et al. [10] present a scenario-based approach on reliability prediction, in which the more fine-grained system architecture model is synthesized for computing a reliability prediction. The approach is based on scenario specifications and Cheung's user-oriented software reliability model [5]. The approach utilizes a high-level message sequence chart (HMSC), which is annotated with scenario transition probabilities derived from the operational profile of the system. An extension to the Bayesian Approach has been proposed by Cortellessa et al. [17]. While the approach uses the same UML extensions as the Bayesian approach, it also extends the approach by annotating the deployment diagram. The annotation of the deployment diagram with the probabilities of failure over the connectors among sites enables the reliability model to embed the communication failures. Zarras and Issarny [21] propose a reliability modelling method that describes the architecture based on the behaviour and reliability aspects of the system. The architecture is first described, after which success criteria, i.e. abstract descriptions of the behaviour expected by the system, are defined through the definition of use case diagrams. The failure rate, MTTF, and reliability of each of the architectural elements are approximated, and described with the *signal class* that describes the failures generated by a particular architectural element. Finally, the overall reliability of the system is assessed using the Reliability Block Diagram (RBD), which can be derived directly from the collaboration diagram. The approach of Grassi [22] directly focuses on service reliability, and is therefore examined further according to our framework. The approach exploits a unified service model that helps to model

and analyze different architectural alternatives, where the characteristics of high and low level services are taken into account. The approach is based on the idea that a set of components requires and subsequently provides services. The approach of Wang et al. [25] especially discussed architectural styles. The approach provides a model for computing the reliability of heterogeneous systems consisting of various architectural styles. System reliability is analyzed based on the reliabilities of components and connectors. The operational profile is taken into account as transition probabilities between the components.

4. RESULTS ON COMPARATIVE ANALYSIS

Although there is a large area of literature review on software reliability and availability, as well as other quality attributes such as maintainability, usability, safety e.t.c. have just recently begun to be addressed at the architecture level by methods, techniques and notations. All of selected methods require some additional work, mostly regarding the development of an analysis model or application of mathematical models and algorithms. It is obvious that approaches closer to UML require less additional work as UML being a widely used standard, and therefore, are more familiar to architects working in industry than the approaches that require a separate analysis model. It is also obvious that more tool support is needed in order to make reliability prediction a fluent part of software development. In recent days, there are several tools available that support at least the analysis that is based on Markov chains [60]. We could not find any method that would also consider variability in the analysis; therefore, no method is applicable for software family engineering. None of the available methods recognizes the variability in reliability requirements, or in architecture descriptions. In several approaches, component reliability was assumed available. There are a number of approaches that especially analyze the reliability of components such as [29], [30]. Except the approach of [1], the analysis approaches studied above do not analyze component reliability, or do not consider the effect of a component's internal behaviour on its reliability.

5. CONCLUSION

In this paper, a framework is defined for comparing existing reliability and availability analysis methods from the software architecture point of view. The comparison of the methods revealed that none of the studied methods alone could provide adequate support for predicting reliability and availability from software architecture. In addition, there was no proof of the maturity of the methods as they were not validated or used in the industry. The methods seemed to focus on analyzing systems by computing, for example, the probability of failure, the probability of repair or some other measures. The main benefit of an integrated environment is that it enables the achievement of a better traceability of reliability and availability requirements, and therefore, a better applicability of the methods for large software products in the industry.

Furthermore there exist some methods that can be applied in the industry as soon as their shortcomings have been removed. The prediction of reliability and availability provides benefits that are visible in both product quality and production efficiency, as long as the prediction is fluently integrated with software architecture design.

REFERENCES

[1] Reussner, R.H., Schmidt, H.W., Poernomo, I.H. "Reliability prediction for component-based

software architectures", *J. Systems Softw*. 66(3), pp. 241–252 (2003)

[2] Molter,G. "Integrating SAAMin Domain-centric and Reuse based development process" *In: Proceedings of the 2nd Nordic Workshop on Software Architecture* (1999)

[3] Bass, L., Clements, P., Kazman, R. "Software Architecture in Practice" *Addison-Wesley, Reading*, 452 p (1998)

[4] Kazman, R., et al. "The architecture tradeoff analysis method" *In: The 4th IEEE International Conference on Engineering of Complex Computer Systems* (1998)

[5] Cheung, R.C. "A user-oriented software reliability model" *IEEE Trans. Softw. Eng.* 6(2), 118–125 (1980)

[6] Musa, J.D. "Operational profiles in software-reliability engineering" *IEEE Softw*. 10(2), 14–32 (1993)

[7] Whittaker, J.A., Thomason, M.G. "A Markov chain model for statistical software testing" *IEEE Trans. Software. Eng.* 20(10), 812–824 (1994)

[8] Jacobson I. "Object-Oriented Software Engineering: A Use Case Driven Approach" *Addison-Wesley, ACM Press*, 400 p (1992)

[9] Runeson, P., Regnell, B. "Derivation of an integrated operational profile and use case model" *In: Proceedings. The Ninth International Symposium on Software Reliability Engineering* (1998)

[10] Rodrigues, G.N., Rosenblum, D.S., Uchitel, S. "Using scenarios to predict the reliability of concurrent component-based software systems" *In: 8th International Conference on Fundamental Approaches to Software Engineering, FASE* 2005.

[11] Thomason, M.G.,Whittaker, J.A. "Rare failure-state in a Markov chain model for software reliability" *In: Proceedings of the 10th International Symposium on Software Reliability Engineering. IEEE, Boca Raton,* (1999)

[12] Gokhale, S., Trivedi, K.S. "Reliability prediction and sensitivity analysis based on software architecture" *In: Proceedings of the 3rd International Symposium on Software Reliability Engineering (ISSRE 02). IEEE Computer Society, Annapolis,* (2002)

[13] Yacoub, S., Cukic, B., Ammar, H. "Scenario-based reliability analysis of component-based software" *In: Proceedings of 10th International Symposium on Software Reliability Engineering (ISSRE'99)* (1999)

[14] Krishnamurthy, S., Mathur, A.P. "On the estimation of reliability of a software system using reliabilities of its components" *In: Proceedings of the 8th International Symposium on Software Reliability Engineering (ISSRE97)* (1997)

[15] Gokhale, S.S., Trivedi, K.S. "Dependency characterization in path-based approaches to architecture-based software reliability prediction." *In: Proceedings of the IEEE Workshop on Application-Specific Software Engineering Technology, ASSET-98* (1998)

[16] Leangsuksun, C., Song, H., Shen, L. "Reliability modelling using UML" *In: Proceeding of the 2003 International Conference on Software Engineering Research and Practice. Las Vegas* (2003)

[17] Cortellessa, V., Singh, H., Cukic, B. "Early reliability assessment of UML based software models" *In: Third International Workshop on Software and Performance. Rome* (2002)

[18] Cortellessa, V., Pompei, A. "Towards a UML Profile for QoS: A Contribution in the Reliability Domain" *In: Proceedings of the Fourth International Workshop on Software and Performance. ACM Press* (2004)

[19] Rodrigues, G.N., Roberts, G., Emmerich, W., Skene, J. "Reliability support for the model driven architecture" *In: Proceedings of the ICSE Workshop on Software Architecture for Dependable Systems. Portland* (2003)

[20] Rodrigues, G.N. "A model driven approach for software systems reliability" *In: 26th International Conference on Software Engineering (ICSE'04). Edinburgh* (2004)

[21] Zarras, A., Issarny, V. "Assessing software reliability at the architectural level" *In: Proceedings of the 4th ACM SIGSOFT International Software Architecture Workshop .ACM, Ireland* (2000)

[22] Grassi, V. "Architecture-based dependability prediction for service-oriented computing" *In: Proceedings of the Twin Workshops on Architecting Dependable Systems, International Conference on Software Engineering (ICSE 2004). Springer, Edinburgh,* (2004)

[23] Greenfield, J. "UML Profile for EJB" *in Technical report. Rational Software Corp* (2001)

[24] Laprie, J.C., Kanoun, K: "X-ware reliability and availability modelling" *IEEE Trans. Software. Eng*. 18(2), 130–147 (1992)

[25] Wang, W.-L., Wu, Y., Chen, M.-H. "An architecture-based software reliability model" *In: Pacific Rim International Symposium on Dependable Computing. IEEE, HongKong* (1999)

[26] Shooman, M. "Structural models for software reliability prediction" *In: Proceedings of the 2nd International Conference on Software Engineering* (1976)

[28] Leangsuksun, C., et al. "Availability prediction and modelling of high availability OSCAR cluster" *In: IEEE International Conference on Cluster Computing. Hong Kong* (2003)

[29] Everett, W. "Software component reliability analysis" *In: IEEE Symposium on Application Specific Systems and Software Engineering and Technology. Richardson* (1999)

[30] McGregor, J.D., Stafford, J.A., Cho, I.-H. "Measuring component reliability" *In: Proceedings of the 6th ICSE Workshop on Component-Based Software Engineering. IEEE, Portland* (2003)