

Calculating Architectural Reliability via Modeling and Analysis

Roshanak Roshandel

*Computer Science Department
University of Southern California
Los Angeles, CA 90089-0781 U.S.A.
roshande@usc.edu*

Abstract

We present a software architecture-based approach to compositional estimation of system's reliability. Our approach is applicable to early stages of development when the implementation artifacts are not yet available, and exact execution profile is unknown. The uncertainty of the execution profile is modeled using stochastic processes with unknown parameters. The compositional approach calculates overall reliability of the system as a function of the reliability of its constituent components and their (complex) interactions. Sensitivity analysis to identify critical components and interactions will be provided.

1. Research problem and importance

Reliability is defined as the probability that a system will perform its intended functionality under specified design limits. *Software* reliability techniques are aimed at reducing or eliminating failures of software systems. Existing software reliability techniques are typically rooted in the field of reliability engineering, and particularly hardware reliability. Such approaches leverage extensive existing experience and provide advanced mathematical formalisms for building software reliability models. They complement *testing* by providing estimates of a program's ability to operate without failure. However, they are not always geared to today's complex software systems, development processes, and their inherent challenges.

As the complexity of software systems increases, it becomes necessary to model them in terms of coarse-grain building blocks. The field of software architecture addresses this issue via high-level abstractions for representing the structure, behavior, and key properties of a software system. Architectural decisions directly affect aspects of system dependability. Identifying and mitigating architectural problems early in the development process helps to increase dependability of a system in a cost-effective manner. To achieve this goal, reliability and other quality attributes must be "built into" the software system throughout the development process, including during the architectural design phase. Therefore, building reliable software systems requires understanding reliability at the architectural level. However, while some recent approaches have started to quantify reliability at the level

of architectural models, they still rely on system implementation to provide either (1) a behavioral model of the system, or (2) each individual component's reliability.

Our work focuses on the structural and behavioral aspects of a software system's architecture, with the goal of addressing the following research question: *Can we use the architectural model of a software system to estimate meaningfully the reliability of individual components, and consequently the reliability of the entire system? What are the benefits obtained thereby?*

2. Related work

Modeling, estimating, and analyzing software reliability—*during testing*—is a discipline with over 30 years of history. Many reliability models have been proposed. Software Reliability Growth Models (SRGMs) are used to both predict and estimate software reliability during testing using statistical approaches [2,5,7,8]. These are *black-box* approaches: they are applied to the software system as a whole and largely ignore its internal structure.

Another category of software reliability modeling techniques is *white-box*: they consider a system's internal structure in reliability estimation. These approaches directly leverage the reliability of individual components and their configuration in order to calculate the system's overall reliability [3,4,6]. They usually assume that the individual component reliability is known or can be obtained via SRGM approaches. White-box techniques are further classified into *path-based* and *state-based* [4]: path-based models compute software reliability based on the system's possible execution paths; state-based models use the control flow graph to represent the system's internal structure and estimate its reliability analytically.

All of the above approaches are based on implementation artifacts and measure the system's reliability during testing. Even those approaches assumed to be applicable in other development phases rely on estimates of the code size [1]. When architectural, existing approaches consider only the structure of the system. The only exceptions are Reussner et al. [10] and Wang et al. [16], both offering compositional approaches to reliability modeling. Reussner et al. build architectural reliability models based on both structural and behavioral specifications of a system. Their parametrized reliability estimation technique

assumes the reliability of individual component services to be known. Similarly, Wang et al. leverage architectural configuration, while focusing on architectural styles for building a prediction model that is mostly concerned with sequential control flow across components in a system. Both approaches assume that component behaviors exhibit the *Markov property* [9], and build Markov models of component interactions as the basis for calculating system reliability. However, neither approach considers the effect of a component’s internal behavior on its reliability. Instead, they rely on the availability of a running system to obtain the frequency of component service invocations. In addition, both approaches assume that each state underlying the Markov model corresponds to an observable event. This may not always be true when modeling complex systems.

3. Research Hypotheses

Our research is based on the following hypotheses.

Hypothesis 1. In order to estimate software reliability at the architectural level, we need rich architectural models as well as reliability models that do not rely on a running system’s operational profile. We hypothesize there exist conditions under which models of architectural structure and behavior can be used to obtain a meaningful estimate of system reliability.

Hypothesis 2. A component’s external behavior is traditionally modeled using *interaction protocols*, i.e., valid sequences of the component’s interaction with its environment. Different paths permitted by a protocol model may contribute differently to a system’s overall reliability. We hypothesize that a stochastic model constructed based on the interaction protocol model can be used to calculate reliability of individual components. The results can then be composed into a global interaction model to calculate the reliability of the overall system. We hypothesize that the global interaction model can be used to calculate the effect of different combinations of components to the overall reliability of the system.

Hypothesis 3. Different cost values are associated with different classes of defects introduced during architectural design. Consequently, different classes of errors can affect the system and reduce its overall reliability in different ways. We hypothesize that our stochastic reliability estimation model can be parametrized for frequency/severity of different defect types, which then may be used to identify defects that are more critical and cost-effective to fix.

4. Approach

We propose a three-part solution to the problem of modeling and quantifying architecture-level reliability of software systems: (1) We will use a multi-view architectural model (*the Quartet*), that can be analyzed for possible architectural defects. It also can serve as a basis for generating implementation-level artifacts. (2) We will estimate

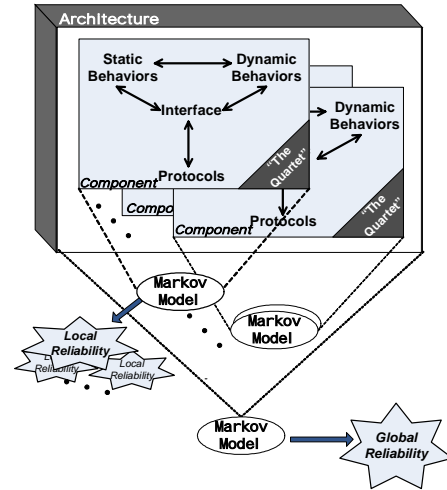


Figure 1. Our approach to local and global reliability modeling

the reliability of a component (*local reliability*) using a stochastic model based on the Quartet, and as a function of the consistency of a component’s internal behavioral models. (3) We will leverage the local reliability estimate and components interaction protocols to estimate the overall system reliability (*global reliability*) in a compositional manner. A high-level graphical view of our approach is depicted in Figure 1. We now explain each step in more detail.

The Quartet. We advocate using a multi-view modeling approach, the Quartet, to comprehensively model the properties of components in a software system: (1) *Interface* models the points at which a component interacts with other components in a system; (2) *Static behavior* models the functionality of a component discretely, i.e., at particular “snapshots” during the system’s execution; (3) *Dynamic behavior* models a continuous view of how a component arrives at different states in its execution; (4) *Interaction protocol* models an external view of the component and its legal interactions with other components in the system.

The conceptual dependencies among the models both within and across components in a system will be the basis for architectural analysis [11,12]. Extensions to our existing architectural modeling environment [14] will automate such analysis. The analysis results will be used by our stochastic model to estimate reliability.

Local Reliability. These automated architectural analyses serve as a basis for estimating component-level reliability. Different classes of architectural defects may affect component reliability in different ways. We have developed a classification of architectural defects [13], which will be parameterized to account for severity/cost values associated with the defects’ mitigation. This will constitute a cost-framework to be used by the stochastic model to quantify local reliability values.

The component *interaction protocol* view provided by the Quartet is the basis for a Hidden Markov Model

(HMM) [9] that will be used to estimate local reliability. HMM is used because in the protocol model, a correspondence between states and observable events may not exist. HMM will allow several algorithms, such as Viterbi and Baum-Welch [9], to be used to estimate the probability of transitions in the model, as well as identify the most/least reliable sequences of transitions. Along with the parameterized cost framework, this Markov model can then be used to rank the architectural defects and prescribe the most cost-effective way to approach defect mitigation.

Global Reliability. We propose a compositional approach to estimating a system's overall reliability based on the reliabilities of its constituent components. A series of *concurrent* state machines, each representing an individual component's behavior, will be used to build an augmented Markov model. This model is further extended to include event/action pairs representing complex interactions (the *glue*) among the components. Using the Markov model, components can be ranked to prescribe an order for mitigating faults, while maximizing overall reliability.

5. Evaluation

The proposed approach will be evaluated by applying our architectural modeling, analysis, and reliability estimation to the SCRover project [15]. SCRover is a real-world application built using NASA/JPL's Mission Data System framework. Extensive documentation, runtime analysis, and empirical evaluation is applied to SCRover by a separate research group. Hypothesis 1 will be evaluated by applying our reliability analysis technique to SCRover and comparing our results with their runtime reliability calculation and empirical evaluation.

We will formally demonstrate the properties of the stochastic models to evaluate our hypothesis 2. In particular, we will demonstrate that system reliability depends on the specific combination of components, and provide justification of the defect mitigation strategies based on the Markov principles and the parametrized cost-framework.

To further evaluate hypotheses 2 and 3, we will build an architectural monitoring and simulation environment, and will use it to simulate multiple classes of defects for given interaction protocol models of components, leverage the parameterized cost-framework, identify components that have the greatest effect on the system's reliability, and suggest defect mitigation strategies.

6. Conclusion and Contribution

Despite the maturity of software reliability techniques, architecture-based reliability modeling has not received much attention. On the other hand, the field of software architecture offers sophisticated modeling and analysis capabilities that often lack quantification and measurement. We are confident that our approach will begin to close the gap between architectural specification and its effect on software reliability. In particular, contributions of our work may be summarized as follows: (1) Multi-view

approach to architectural modeling and mechanisms to ensure the intra- and inter-consistency among the views; (2) A formal reliability model to calculate both component-level and system-level reliability of a given software system based on its architectural specification; (3) Identification of critical defects whose mitigation is most cost-effective in increasing a system's overall reliability.

7. References

- [1] Dalal, S.R., Software Reliability Models: A Selective Survey and New Directions, *Handbook of Reliability Engineering*, edited by H. Pham, Springer, 2003.
- [2] Goel A.L., Okumoto K., Time-Dependent Error-Detection Rate Models for Software Reliability and Other Performance Measures, *IEEE Trans. on Reliability*, 28(3):206-211, 1979.
- [3] Gokhale S., Philip T., Marinou P., Trivedi K., Unification of finite-failure nonhomogenous Poisson process models through test coverage, in *Proceedings of the 7th IEEE International Symposium on Software Reliability Engineering (ISSRE-96)*, November. 1996.
- [4] Goseva-Popstojanova K., Mathur A.P., Trivedi K.S., Comparison of Architecture-Based Software Reliability Models, in *Proceedings of the 12th IEEE International Symposium on Software Reliability Engineering (ISSRE-2001)*, 2001.
- [5] Jelinski, Z. and Moranda, P. B., Software Reliability Research, *Statistical Computer Performance Evaluation*, edited by W. Freigerger, Academic Press, 1972.
- [6] Krishnamurthy S., Mathur A.P., On the estimation of reliability of a software system using reliability of its components, in *Proc. of the 8th IEEE International Symposium on Software Reliability Engineering*, November 1997.
- [7] Littlewood, B.A., and Verrall, J.L., A Bayesian Reliability Growth Model for Computer Software, *Applied Statistics, Volume 22*, pp. 332-346, 1973
- [8] Musa J.D., and Okumoto K., Logarithmic Poisson Execution Time Model for Software Reliability Measurement, in *Proceedings of. Compsac 1984*, pp. 230-238, 1984.
- [9] Rabiner L.R., A Tutorial on Hidden Markov Models, in *Proceedings of the IEEE*, vol. 77, pp. 257-286, 1989
- [10] Reussner R., Schmidt H., Poernomo I., Reliability prediction for component-based software architectures, In *Journal of Systems and Software*, 66(3), Elsevier Science Inc, 2003.
- [11] Roshandel R., Medvidovic N., Modeling Multiple Aspects of Software Components, in *Proceeding of Workshop on Specification and Verification of Component-Based Systems, ESEC-FSE03*, Helsinki, Finland, September 2003.
- [12] Roshandel R., Medvidovic N., Multi-View Software Component Modeling for Dependability, in *Architecting Dependable Systems II*, Lecture Notes in Computer Science. Rogerio de Lemos, Cristina Gacek, Alexander Romanovsky, (Editors.), 2004 (to appear).
- [13] Roshandel R., Schmerl B., Medvidovic N., Garlan D., and Zhang D., Using Multiple Views to Model and Analyze Software Architecture: An Experience Report, *USC Technical Report USC-CSE-2003-508*, September 2003.
- [14] Roshandel R., van der Hoek A., Mikic-Rakic M., Medvidovic N., Mae - A System Model and Environment for Managing Architectural Evolution, Submitted to *ACM Transactions on Software Engineering and Methodology (In review)*, October 2002.
- [15] SCRover Project: <http://cse.usc.edu/hdcp/iscr>
- [16] Wang W., Wu Y., Chen M., An architecture-based software reliability model, in *Proc. of Pacific Rim International Symposium on Dependable Computing*, 1999.