Software Architecture Reliability Analysis using Failure Scenarios[*]

Bedir Tekinerdoğan, Hasan Sözer, Mehmet Akşit
Department of Computer Science, University of Twente,
P.O. Box 217 7500 AE Enschede, The Netherlands
{ bedir, sozerh, aksit }@cs.utwente.nl

Abstract

We propose a Software Architecture Reliability Analysis (SARA) approach that benefits from both reliability engineering and scenario-based software architecture analysis to provide an early reliability analysis of the software architecture. SARA makes use of failure scenarios that are prioritized with respect to the user-perception in order to provide a severity analysis for the software architecture and the individual components.

1. Introduction

Software Architecture Reliability Analysis (SARA) is an approach which integrates scenario-based approaches with conventional reliability analysis techniques. SARA defines a failure scenario model that is based on the established Failure Modes and Effects Analysis (FMEA) method in the reliability engineering domain. Failure scenarios are systematically derived and expressed using this model. The developed failure scenarios are utilized to derive a Fault Tree Set (FTS) in which failures are prioritized based on severity from the perspective of the user. Severity analysis is provided for the top-level architecture and the most relevant fault categories are identified for the individual components. The method results in a failure analysis report that can be used for improving reliability of the software architecture with respect to user-perceived failures.

[*] This work has been carried out as part of the TRADER project under the responsibility of the Embedded Systems Institute. This project is partially supported by the Netherlands Ministry of Economic Affairs under the Bsik program.

2. Reliability Analysis

The steps of SARA are presented in Figure 1. Here, the rectangles represent the steps and the arrows represent the control flow.

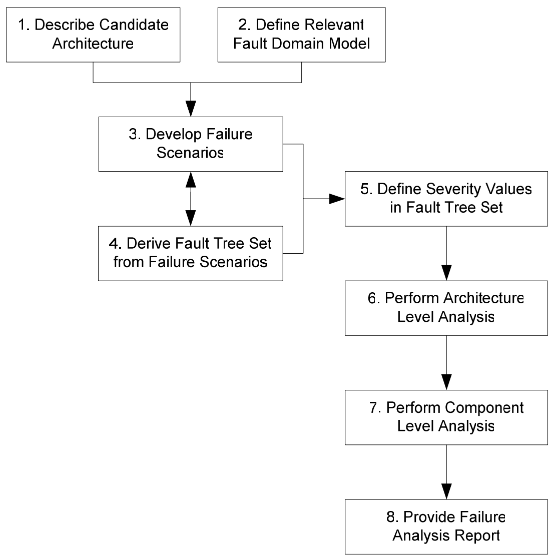
1. Describe Candidate Architecture
2. Define Relevant Fault Domain Model
3. Develop Failure Scenarios
4. Derive Fault Tree Set from Failure Scenarios
5. Define Severity Values in Fault Tree Set
6. Perform Architecture Level Analysis
7. Perform Component Level Analysis
8. Provide Failure Analysis Report

Figure 1. *Software Architecture Reliability Analysis*

Similar to existing software architecture analysis methods, SARA starts with defining alternative architectures. The candidate architecture includes the architectural components and their relationships. The method does not presume to provide a particular architectural view. In the mean time, definition of a relative fault domain model is also necessary since all potential faults will not be relevant for a given reliability analysis project. The fault domain model is utilized to derive failure scenarios which are represented by means of a failure scenario model. In this model, each scenario is represented by means of the tuple <*failure id*, *component acronym*, *failure mode*, *failure cause*, *failure effect*>. For instance, <F9, DDI, "Data cannot be interpreted", "Reception of out-of-spec signals", "Provide wrong/incomplete information"> defines a failure scenario corresponding to Data Decoder & Interpreter (DDI) component. The failure cause (i.e. "Reception of out-of-spec signals") is categorized in accordance with the fault domain model.

Failure scenarios that are developed are connected to each other. That is, failure of a component triggers failure of another component. To make the connections explicit, we construct fault trees. A fault tree is a model for representing the cause-effect relations of faults, in which the root node represents a system failure. Since a failure can be caused by a set of faults, the nodes of the tree are interconnected with logic gates characterizing the propagation behavior. We consider multiple system failures which lead to a set of fault trees. We term this as a Fault Tree Set (FTS).

Once the FTS is identified, we define the severity degrees. In conventional fault tree analysis, fault trees are used in order to calculate the probability that a failure would take place, based on the probabilities of fault occurrences ([3]). The severity is defined as a concept related to faults denoting how severe a fault is (e.g. faulty component can be repaired or not). In our model, we take a user-centric approach and define severity based on the user-perception. System failures that we consider are not restricted to complete crash-down of the system and they would not upset the user in the same way. As a diversion from the usual approach, we assign severity values to intermediate failures and faults based on severities of system failures.

After calculating severity values, we perform architecture level analysis in which we pinpoint sensitive points of the architecture with respect to reliability. For all components, we analyze associated failures and their severities. In Figure 2, result of architectural level analysis is given in which 26 components are compared in terms of weighted failures impacting them. In component level analysis, we analyze the categories of faults that impact a component in accordance with the fault domain model. In Figure 3, percentage of permanent and transient faults that impact a component is analyzed. Failure analysis report summarizes the analysis results and provides hints for improvements.

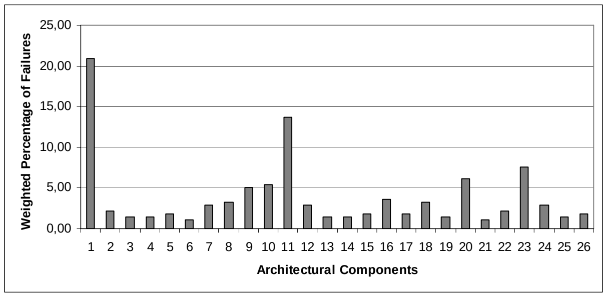

Figure 2. Architecture Level Analysis

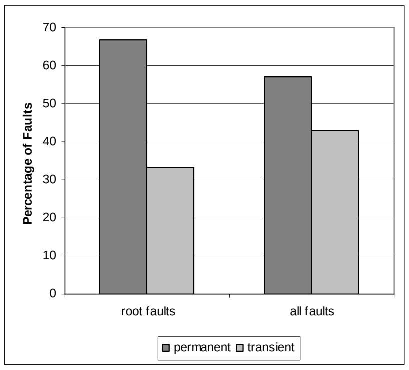

Figure 3. Component Level Analysis

3. Conclusion

We have introduced SARA, software architecture reliability analysis method that has been developed after a study of both software architecture analysis methods and reliability engineering techniques. The overall scenario elicitation and prioritization is inspired from the work on software architecture analysis methods ([2]). The definition of a fault domain model, utilization of fault trees and failure model are inspired from the reliability engineering domain ([1], [3]). The usage of a failure model is beneficial for deriving scenarios in a systematic way. In fact, we believe that it is necessary to define quality attribute models to provide a meaningful and feasible scenario-based analysis.

References

[1] A. Avizienis, J.-C. Laprie, B. Randell, Fundamental Concepts of Dependability, LAAS Report No 01145, LAAS-CNRS, France, April 2001.

[2] L.Dobrica & E.Niemela, A survey on software architecture analysis methods, IEEE Trans. on Software Engineering, Vol. 28, No. 7, pp.638-654, July 2002.

[3] M. R. Lyu, Editor, *Handbook of Software Reliability Engineering*, McGraw-Hill, New York, NY, 1996.