Sensitivity Analysis for a Scenario-Based Reliability Prediction Model

Genaína N. Rodrigues[1], David S. Rosenblum[1] and Sebastian Uchitel[2]

London Software Systems

Department of Computer Science[1]
University College London
Gower Street
WC1E 6BT
United Kingdom
{g.rodrigues, d.rosenblum}@cs.ucl.ac.uk

Department of Computing[2]
Imperial College London
180 Queen's Gate
SW7 2RH
United Kingdom
su2@doc.ic.ac.uk

ABSTRACT

As a popular means for capturing behavioural requirements, scenarios show how components interact to provide system-level functionality. If component reliability information is available, scenarios can be used to perform early system reliability assessment. In previous work we presented an automated approach for predicting software system reliability that extends a scenario specification to model (1) *the probability of component failure*, and (2) *scenario transition probabilities*. Probabilistic behaviour models of the system are then synthesized from the extended scenario specification. From the system behaviour model, reliability prediction can be computed. This paper complements our previous work and presents a sensitivity analysis that supports reasoning about how component reliability and usage profiles impact on the overall system reliability. For this purpose, we present how the system reliability varies as a function of the *components reliabilities* and the *scenario transition probabilities*. Taking into account the concurrent nature of component-based software systems, we also analyse the effect of implied scenarios prevention into the sensitivity analysis of our reliability prediction technique.

Categories and Subject Descriptors

D.2.1 [**Software Engineering**]: Requirements/Specifications—*elicitation methods (scenarios), methodologies*; D.2.2 [**Software Engineering**]: Design Tools and Techniques—*state diagrams*; D.2.4 [**Software Engineering**]: Software/Program Verification—*model checking, reliability, statistical methods*; D.2.10 [**Software Engineering**]: Design—*methodologies*

General Terms

Design, Languages, Reliability, Verification

WADS '05 May 17, 2005, St. Louis, MO, USA.

Keywords

architecture models; labelled transition systems; software reliability prediction; scenario specification; sensitivity analysis

1. INTRODUCTION

Software reliability engineering is an important aspect of many system development efforts, and consequently there has been a great deal of research in this area [6, 4]. One important activity included in software reliability engineering is *reliability prediction* [4]. A promising compositional approach to predicting reliability of component-based systems early in the lifecycle is to base the prediction on scenarios of system usage.

Scenarios have been widely adopted as a way to capture system behavioural requirements. *Message Sequence Charts* (MSCs) are a widely accepted notation for scenario-based specification. MSCs are classified into two types: Basic MSCs (BMSCs) and High-level MSCs (HMSC). The BMSCs are equivalent to the UML Sequence Diagrams (SDs), which model sequential message exchange between components. HMSCs are a widely used notation to compose BMSCs through three fundamental constructs: *vertical composition* (where two BMSCs are composed sequentially), *alternative composition* (defining that the system could alternatively choose one of the BMSCs to follow) and *iterative composition* (which composes a sequence of BMSCs). An HMSC is a directed graph, whose nodes refer to BMSCs and whose edges indicate the acceptable ordering of the BMSCs. HMSCs allow stakeholders to reuse scenarios within a specification and to introduce sequences, loops and alternatives of BMSCs. The semantics of an HMSC is the set of sequences of interactions that follow some maximal path through the HMSC.

In previous work, we presented a scenario-based approach to reliability prediction of concurrent systems by synthesizing architecture model from scenario specification [8]. The contribution of this paper is to show how our reliability prediction technique can provide guidance towards enhancing the reliability of the software system. We complement our technique with sensitivity analysis to identify components and usage profiles with greater impact on the system reliability. We use the Boiler Control system of our previous work as a way to exemplify our sensitivity analysis. For this purpose, we present how the system reliability is sensitive to the (1) *components reliabilities* and (2) *scenario transition probabilities*. These two analyses can help us to identify components and scenarios transitions that could threat the reliability of the software

system. Taking into account the concurrent nature of component-based software systems, we also analyse in this paper what effects the prevention of undesirable implied scenarios cause to our reliability prediction technique.

Succinctly, sensitivity analysis of software reliability can be performed in two different ways: *analytically*, where a general function can be derived for all systems, or *experimentally*, where results are obtained through measurement. Cheung indicates analytically how to improve the system reliability by differentiating the system reliability with respect to the reliability of the program modules [2]. Siegrist presents a more fine-grained analytical sensitivity analysis by making the partial derivative of system reliability with respect to the reliability of the system states [9]. The sensitivity analysis of the scenario-based method of Yacoub et al. [12] involves experimentally analyzing the system reliability as a function of (1) the component reliability, (2) the reliability of transition between components, and (3) the scenario usage profile based on a numerical solution (rather than an analytical one like Cheung's). We follow the experimental approach by presenting plots of system reliability whose curves are interpolated over sampled values for the elements to which our prediction technique is sensitive.

Throughout this paper we use a variant of the Boiler Control System example presented by Uchitel et al. [11]. The Boiler Control system in Figure 1 consists of four components: *Sensor*, *Control*, *Database* and *Actuator*. In the upper-left corner of the figure, we depict the HMSC specification of the Boiler, which composes five BMSCs: *Initialise*, *Register*, *Analyse*, *Terminate* and *End*. Note that the variables appearing in curly brackets in the figure are an extension to MSCs that we briefly explain in Section 2. The paper is

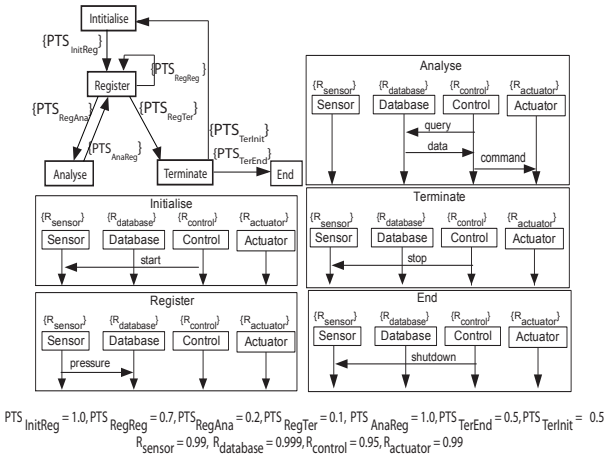

Figure 1: The Message Sequence Chart Specification for the Boiler Control System, with Example Probability Values.

structured as follows: In Section 2, we present some background on our reliability prediction technique. In Section 3 we present sensitivity analysis for our technique showing how the system reliability can be described as a function of the components and the transitions between the scenarios. In Section 4 we analyse the impact that implied scenarios have in our reliability prediction technique. In Section 5 we compare the reliability prediction curve prior to and after accounting for implied scenarios. Finally, in Section 6 we present our conclusions and discuss directions for future work.

2. BACKGROUND

In this section we present the overall principles behind our method to predict software system reliability. The method is based on the annotation of a scenario specification with probabilistic properties and the use of a probabilistic labelled transition system (LTS) synthesised from the scenario specification for the software reliability prediction. We refer the reader to our earlier paper for a more detailed explanation of our model [8].

2.1 Reliability Analysis Using Scenarios

The method comprises five major steps: (1) annotation of the scenarios, (2) synthesis of the probabilistic LTS, (3) construction of the stochastic matrix, (4) system reliability prediction, and (5) implied scenario detection.

In the first step, we annotate the scenarios (i.e., the HMSC and BMSCs) with two kinds of probabilities, *the probability of transitions between scenarios* PTS_{ij} and *the reliability of the components* R_C.

The transition probability PTS_{ij} is the probability that the system will exhibit the behaviour specified in scenario S_j after executing scenario S_i. This information would be normally derived from an operational profile for the system [5]. Thus, from scenario S_i, the sum of the probabilities PTS_{ij} for all successor scenarios S_j is equal to one. As the PTS_{ij} relates to the transition between scenarios, these probabilities are annotated on the corresponding edges of the HMSC, as shown on the HMSC of Figure 1. For the purposes of illustrating our method on the Boiler example, we use the values depicted in Figure 1 for the PTS_{ij}. The values for the PTS_{ij} are based on the assumption that the system executes the scenario *Register* (which causes sensor readings to be entered into the database) far more frequently than the scenarios *Analyse* and *Terminate*, and that when it does execute *Terminate* there is an equal probability of reinitialising and shutting down.

The component reliabilities R_C are annotated on the BMSCs, as also shown in Figure 1. Without loss of generality, we use coarse-grained, single values for the overall component reliabilities; in general, we could associate reliabilities with individual messages and/or segments of component timelines. The values in Figure 1 for the reliability of the components reflect the assumption that the *Database* is a highly reliable, mature commercial software product, that the *Sensor* and *Actuator* are components whose hardware interface to the sensed/actuated phenomena will eventually fail, and that *Control* is a complex software subsystem that still contains latent faults.

The second step of our method is to synthesise a probabilistic LTS from the annotated scenario specification. This step is an extension of the synthesis approach of Uchitel et al. [10]. Our extension exploits recent probabilistic extensions to the LTS formalism [1] and involves enhancements to the LTS synthesis of Uchitel et al. [10]. The enhancements have the effect of mapping the probability annotations of the scenario specification into probability weights for transitions in the synthesised architecture model. The probability weights are computed correctly in the process of reducing each component LTS to its deterministic, minimal form.

Concluding the process of the probabilistic LTS synthesis, the system architecture model is constructed as the parallel composition of the LTSs synthesized for each component. The probability weights of the composed LTS are computed according to the notion of *generative parallel composition* defined by D'Argenio et al. [3]. At the end of this step, it follows that for each node of the synthesized architecture model, $\forall i$ such that $1 \leq i \leq n$, $\sum_{j=1}^{n} PA_{ij} = 1$, where n is the number of states in the LTS architecture model and PA_{ij} is the probability of transition between state S_i and S_j of the composed LTS; $PA_{ij} = 0$ if the transition (S_i, S_j) does not exist.

In the third and fourth steps of our reliability prediction method,

the architecture model synthesised in the second step is interpreted as a Markov model, and we apply the method of Cheung [2] to compute the reliability prediction. In particular, the transition probability weights of the architecture model are mapped into a square transition matrix whose row entries sum to one. So as to conform to Cheung's model, we have to make sure that there is only one initial and one final scenario in the specification. This is the reason why we include the scenario *End* in our MSC specification as depicted in Figure 1.

2.2 Implied Scenarios

It has been shown that given a scenario specification, it may be impossible to build a set of components that communicate exclusively through the interfaces described and that exhibit only the specified traces when running in parallel [11]. The additional unspecified traces that are exhibited by the composed system are called *implied scenarios* and are the result of specifying the behaviour of a system from a global perspective yet expecting the behaviour to be provided by the components, which have only a local system view. The Boiler Control System of Figure 1 has implied scenarios, and

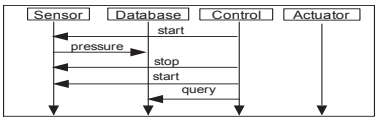

Figure 2: Implied Scenario Detected.

Figure 2 shows one of them. From the specification we see that the Boiler Control System is expected to exhibit a trace *"start, pressure, query, data, command ..."* and that component *Control* interacts with *Database* only through messages *query* and *data*. However, in the implied scenario of Figure 2 a *query* is being performed immediately after *start*.

From the reliability prediction point of view, the existence of an implied scenario means that the Boiler produces a trace that reveals a mismatch between behaviour and architecture. In that case, the model can exhibit behaviour (an implied scenairo) that has not yet been validated and that, depending on whether it describes intended or unintended system behaviour, can impact system reliability. If we decide that the occurrence of the trace is desirable, a new BMSC containing the trace is added and placed appropriately in the HMSC composition. But if we consider the occurrence of the trace in Figure 2 as undesirable, the Boiler architecture model should be constrained in such a way that the implied scenario cannot occur. In both situations, the reliability must be recalculated. We can use the approach described by Uchitel et.al. to build such a constraint [11]. From here on, we refer to the model where we apply those constraints as the *Constrained Model*, while the unconstrained model we refer to as the *Architecture Model*.

3. SENSITIVITY ANALYSIS

In this section we illustrate some sensitivity analyses that can be performed for our reliability prediction technique. We carry out two different analyses of our technique: (1) as a function of the components' reliability and (2) as a function of the transition probability between scenarios. The analyses are exemplified using the Boiler Control system.

3.1 System Reliability as a Function of Component Reliability

This analysis consists in varying the system reliability as a function of the components' reliabilities with the purpose of identifying components that have the greatest impact on the reliability of the software system. The method consists of varying the reliability of one component at a time and fixing the others to 1. The transition probability values are those for PTS_{ij} presented in Figure 1, where i and j represent respectively the *from* and *to* scenarios of the transition. Figure 3 shows the graphs of the reliability of the

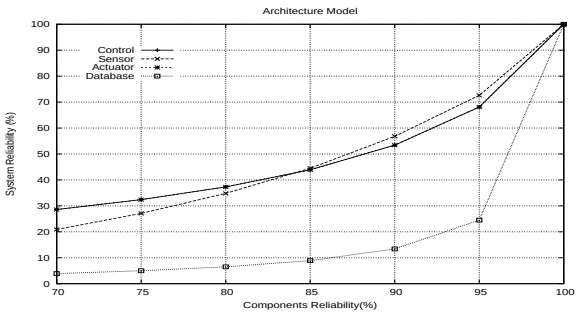

Figure 3: The System Reliability of the Architecture Model as a Function of the Component Reliabilities.

system architecture model as a function of the component reliability. Note that the component *Database* has a large impact on the system reliability, in such a way that the system reliability drops quickly *Database* reliability decreases below 100%. We can initially attribute this to the fact that the system reliability is directly proportional to the number of requests that each component processes. But not only, otherwise *Sensor* would have the strongest impact on the system reliability.

We consider this result due to the higher probability of transitions to scenarios where *Database* takes part than to scenarios where *Sensor* takes part. The *Database* is executed in scenarios *Register* and scenario *Analyse*. According to the Figure 1, the outgoing transitions of scenario *Register* have chances of following three different paths: (1) 70% chance that scenario *Register* will execute again, (2) 20% chance that scenario *Analyse* will be executed and (3) 10% chance that scenario *Terminate* will be executed. This gives component *Database* 100% chance that it is executed, considering successful the transition from scenario *Initialise* to scenario *Register*, followed by 70% chance from the execution of the *Register* loop transition and another 20% chance of transition from scenario *Register* to scenario *Analyse*. On the other hand, the *Sensor* participates in scenarios *Initialise*, *Terminate* and *End*. Chances that the *Sensor* will be invoked correspond to 100% for *Initialise* execution, 10% for the execution of scenario *Terminate* and 50% for the execution of scenario *End*. Compared to the results for the *Database*, we can figure out why *Database* has a higher impact on the reliability of the Boiler compared to the *Sensor*.

Components *Control* and *Actuator* identically show less impact on the system reliability. This is because they are always invoked in the same scenario, *Analyse*, the same number of times, once. Therefore, they are expected to equally have less impact on the overall system reliability.

3.2 System Reliability as a Function of Transition Probability

This analysis consists in varying the system reliability as a function of the scenario transition probabilities. Considering scenarios

with multiple outgoing transitions, we want to find out if scenario transition probabilities have a significant influence on our reliability predictions. In case that influence is significant, we want to find out which of those transitions has higher impact on system reliability. Using the Boiler system as an example, we analyse the impact of outgoing transitions from scenarios *Register* and *Terminate*, as the other scenarios have only one outgoing transition with unitary probability transition. To run this experiment, we keep the component reliabilities values in Figure 1. Taking one transition

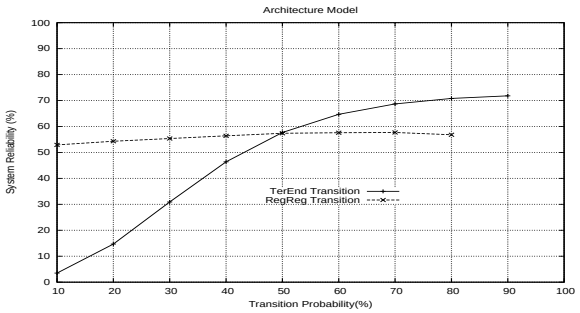

Figure 4: The System Reliability of the Architecture Model as a Function of the Transition Probabilities

of scenario *Register* and scenario *Terminate* is enough to analyse the sensitivity of the system reliability as a function of the transition probability. This is due to the fact that outgoing transitions of a scenario sum to a unity. Varying one transition, we vary the remaining outgoing transitions of the same scenario proportionally, so that transitions sum to 1. For instance, consider the outgoing transitions of scenario *Register* in Figure 1. For instance, if we change PTS_{RegReg} from 0.7 to 0.1, we allocate the remaining 0.9 to PTS_{RegAna} and PTS_{RegTer} in a way that preserves the ratio between them (0.6 and 0.3, respectively). For further information on probabilistic composition of concurrent processes, we refer the reader to D'Argenio et.al [3].

The results depicted in Figure 4 show that outgoing transitions from scenario *Terminate* have more impact on the overall system reliability than outgoing transitions from scenario *Register*. This impact is a result of the role that scenario *Terminate* plays in the whole Boiler system. Intuitively, this can be reasoned by the fact that the higher the probability of transition from *Terminate* to *Initialise*, the higher the chance that the whole system will fail, as both scenarios comprise the end and the beginning of an iteration through the system. Conversely, the higher the chance the transition from *Terminate* to *End*, the higher the system reliability, meaning fewer loops will happen from *Terminate* to *Initialise* and therefore fewer chances for the failure of the system.

4. THE IMPACT OF IMPLIED SCENARIOS ON THE SYSTEM RELIABILITY

As we identified in section 2, the Boiler Control System presents an implied scenario that is considered undesirable for the system specification. From the reliability prediction point of view, it means that the Boiler produces a trace that reveals a mismatch between behaviour and architecture. Preventing the occurrence of those traces, we conduct the experiment in the same way we applied in Section 3. We then obtain the *Constrained Model* and analyse the system reliability as follows. The results are depicted in Figures 5 and 6.

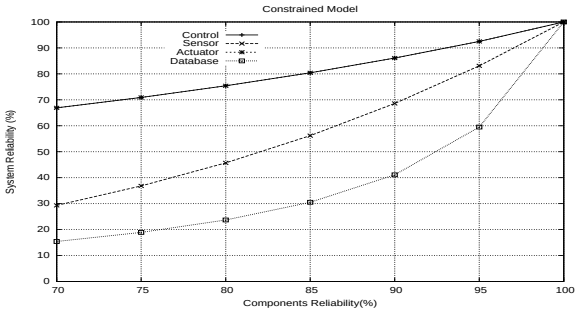

Figure 5: The System Reliability of the Constrained Model as a Function of the Component Reliabilities.

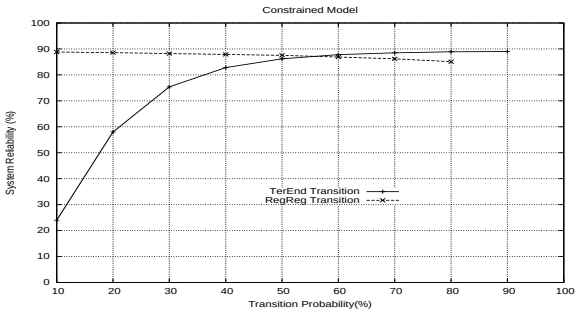

Figure 6: The System Reliability of the Constrained Model as a Function of the Transition Probabilities.

Figure 5 shows the graphs for the system reliability of the Constrained Model as a function of the component reliabilities. Comparing to previous results presented in Figure 3, shows that the system reliability as a function of each component increases when the Constrained Model is applied for the Boiler system. The Constrained Model also has an impact on the system reliability as a function of the transition probability. Results depicted in Figure 6 compared to those depicted in Figure 4, show a considerable increase on the system reliability when the Constrained Model is applied for the Boiler system. Therefore, the implied scenarios show a considerable influence on software reliability prediction.

5. COMPARING THE RELIABILITY PREDICTION CURVES

In previous sections, we compute reliability predictions in the infinite extreme over all possible executions, as provided by Cheung model [2] and our previous work [8]. In this section, we compute reliability predictions as a function of the number of scenario executions.

To measure the system reliability after a certain number of system executions, we start from the Cheung definition that the system failure probability, E, is the probability of reaching state F (faulty termination) from the initial state N_1:

$$E = P^n(N_1, F) \quad (1)$$

P represents the stochastic matrix with all the state transitions of

the synthesized LTS, including the transitions to the absorbing states C (correct termination) or F, (faulty termination). $P^n(i, j)$ is the probability that starting from state i, the chain reaches state j at or before the n^{th} step. As a result of Equation 1, we have a matrix

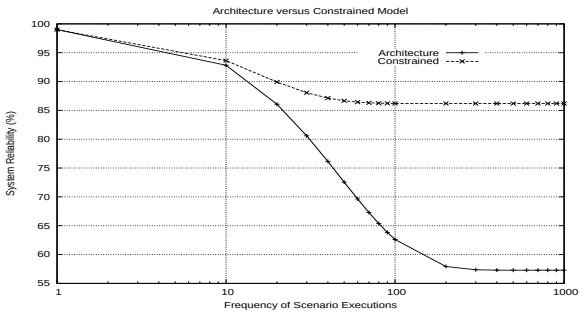

Figure 7: The System Reliability as a Function of the Frequency of Scenario Executions.

that, after n steps, results in the probability that execution traverses from state N_1 to the final faulty state F. Three steps have to be carried out in order to obtain the graphs in Figure 7: (1) Multiply P with itself for discrete number of steps n: P^n; (2) Obtain from P^n the probability of failure of the system E: the probability of reaching the absorbing fault state F from the initial state N_1; (3) Calculate the reliability of the system as $R = 1 - E$.

The analysis of Figure 7 shows that the greater the number of scenarios executed, the more noticeable is the difference between the reliability predicted for the Architecture Model and the Constrained Model. Furthermore, the reliability flattens out at around 300 scenario executions for the Architecture Model and 70 for the Constrained Model, as the likelihood of avoiding the *End* scenario becomes minuscule after those points.

6. CONCLUSION AND FUTURE WORK

In this paper, we have presented sensitivity analysis of our scenario-based technique for predicting software system reliability, taking into account the component structure exhibited in the scenarios and the concurrent nature of component-based systems.

For future work, we will investigate more deeply the underlying nature of the implied scenarios and the ways they affect reliability, as the results in this paper indicated a considerable influence of implied scenarios on software reliability prediction. Also, we plan to apply our approach on case studies of larger, more realistic systems in order to evaluate its scalability and the accuracy of the predictions it produces. Additionally, we will carry on previous work for the purpose of model-driven development, where the ultimate purpose is to define reliability properties in the software design and generating code for different platforms that preserves those properties [7].

Acknowledgment

David Rosenblum holds a Wolfson Research Merit Award from the Royal Society. Sebastian Uchitel was partially funded by EPSRC grant READS GR/S03270/01. Genaína Rodrigues was funded by CAPES, under grant number 108201-9. We would like to thank Rami Bahsoon for his useful comments during the realization of this work.

7. REFERENCES

[1] T. Ayles, A. Field, J. Magee, and A. Bennett. Adding performance evaluation to the LTSA tool (tool demonstration). In *Proc. 13th Performance Tools*, September 2003.

[2] R. C. Cheung. A User-Oriented Software Reliability Model. In *IEEE Transactions on Software Engineering*, volume 6(2), pages 118–125. IEEE, Mar. 1980.

[3] P. R. D'Argenio, H. Hermanns, and J.-P. Katoen. On generative parallel composition. In C. Baier, M. Huth, M. Kwiatkowska, and M. Ryan, editors, *Electronic Notes in Theoretical Computer Science*, volume 22. Elsevier, 2000.

[4] M. R. Lyu. *Handbook of Software Reliability Engineering*. IEEE Computer Society Press and McGraw-Hill, 1996.

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

[6] J. D. Musa, A. Iannino, and K. Okumoto. *Software reliability: measurement, prediction, application*. McGraw-Hill, Inc., 1987.

[7] G. Rodrigues, G. Roberts, and W. Emmerich. Reliability Support for the Model Driven Architecture. In R. de Lemos, C. Gacek, and A. Romanovsky, editors, *Architecting Dependable Systems II –LNCS 3069*. Springer Verlag, 2004.

[8] G. Rodrigues, D. Rosenblum, and S. Uchitel. Using Scenarios to Predict the Reliability of Concurrent Component-Based Software Systems. In M. Cerioli, editor, *FASE 2005 – LNCS 3442*, pages 111–126. Springer, 2005.

[9] K. Siegrist. Reliability of systems with markov transfer of control. *IEEE Trans. Softw. Eng.*, 14(7):1049–1053, 1988.

[10] S. Uchitel, J. Kramer, and J.Magee. Synthesis on Behavioral Models from Scenarios. In *IEEE Transactions on Software Engineering*, volume 29(2), pages 99–115. IEEE, Feb. 2003.

[11] S. Uchitel, J. Kramer, and J.Magee. Incremental Elaboration of Scenarios-Based Specifications and Behavior Models Using Implied Scenarios. In *ACM Transactions on Software Engineering and Methodologies*, volume 13(1), pages 37–85. ACM Press, Jan. 2004.

[12] S. M. Yacoub, B. Cukic, and H. H. Ammar. Scenario-Based Reliability Analysis of Component-Based Software. In *Proc. of the* 10^{th} *ISSRE, Boca Raton, FL, USA*. IEEE, Nov. 1999.