

*2009 International Symposium on Computing, Communication, and Control (ISCCC 2009)
Proc .of CSIT vol.1 (2011) © (2011) IACSIT Press, Singapore*

UML Support for Reliability Evaluation

Martin Jedlicka¹, Oliver Moravcik¹, Peter Schreiber¹ and Pavol Tanuska¹⁺

¹ Faculty of Materials Science and Technology in Trnava, Slovak University of Technology, Bratislava, Slovak Republic

Abstract. Today's software systems are developed and targeted for satisfying sometimes very critical functions. Reliability is considered to be one of the most important nonfunctional quality attribute of such software systems. The aim of reliability estimation in early stages of software development process – analysis and design – should reduce the future costs for possible failure repairing through increasing the reliability before the construction of the software system. Because, the Unified Modeling Language (UML) becomes the standard for software system's specification, the last works done in architecture based reliability estimation and assessment use UML as the base for software architecture specification. In this paper, we discuss the existing approaches with critical overview and outline the directions for future research.

Keywords: Software Reliability, Software Reliability Assessment, Unified Modelling Language

1. Introduction

The role of satisfying of nonfunctional requirements is at least important as it is for functional requirements. The nonfunctional requirements are often evaluated in the end of the whole software development process which might cause the additional costs and also the inevitable requirement for redesigning the whole system architecture. Therefore, the evaluation of nonfunctional requirements at software architecture level should be the reason which could not be.

One of the most important non-functional requirements, which cannot be missed out when designing software system, is the reliability. The software reliability is the probability that a software system is performing successfully its required functions for the duration of a specific mission profile. [5]. In the past years there were introduced several approaches to evaluate reliability of software systems. [5] The [7] presents that evaluating the reliability at the architecture level is possible.

2. UML And Reliability

Unified Modelling Language (UML) becomes the standard for software system's specification. UML provides tools for modeling the software system architecture which is not dependent on programming language, programming techniques and human factor.

UML provides several types of diagram to model the software behavior and architecture. For the reliability evaluation purposes these following diagrams were used in the past:

- Use Case diagram,
- Sequence diagram,
- Deployment diagram,
- Statechart Diagram.

⁺ Corresponding author. Tel.: +421 918 646 061; fax: +421 33 5511758.
E-mail address: pavol.tanuska@stuba.sk.

2.1. Approach of Cortellessa et al.

The approach [2] provides quite complex approach which evaluates the reliability of the whole system using Use Case diagrams, Sequence diagrams and Deployment diagrams.

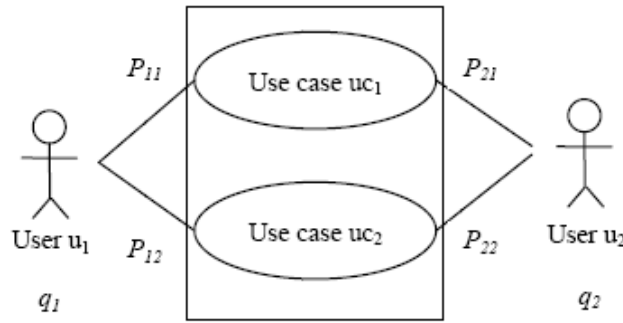


Fig. 1. Annotated Use Case Diagram

The approach presents the Use Case diagram is annotated as follows:

- q_1 and q_2 represent the probabilities of requesting services from system for users u_1 and u_2
- P_{11} and P_{12} represent that user u_1 requests the functionality f_1 represented by use case uc_1 or f_2 represented by use case uc_2 .

Then the probability of executing general use case x is:

$$P(x) = \sum_{i=1}^m q_i \cdot P_{ix}$$

where m is the number of user types.

Then there is the assumption that for each use case there are specified all relevant Sequence Diagrams representing main scenarios each use case. These sequence diagrams have the non-uniform probability distribution according to following equation:

$$P(k_j) = p(j) \cdot f_j(k)$$

where $f_j(k)$ is the frequency off the k -th overall the sequence diagrams referring to the j -th use case and parameter $P(k_j)$ represents the probability of a scenario execution.

In sequence diagram, the approach focuses on the interval where the component is busy. For example, in Figure 2 the interval, when component C_3 is busy, is between the moment when interaction l_{11} enters the component and interaction l_{22} leaves the component. The total number of busy periods off component C_3 is 2.

Then the estimate of the probability of failure θ_{ij} of the component C_i in the scenario j represents the next equation:

$$\theta_{ij} = 1 - (1 - \theta_{ij})^{bp_{ij}}$$

where bp_{ij} the number of busy periods that component C_i has in the scenario j .

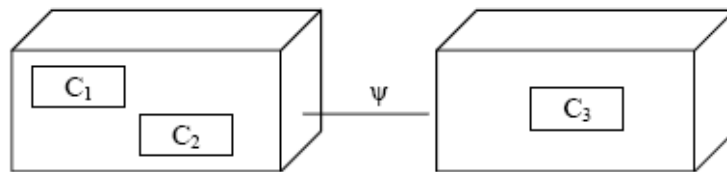


Fig. 3. Annotated Deployment Diagram

The deployment diagram in Figure 3 includes the same components from the sequence diagram. The subject of failure probability ψ_i is every communication between components (l, m) through the connector i . and $|Interact(l, m, j)|$ represents the number of interactions between components l and m from the scenario (sequence diagram) j . e.g. there exist 2 interactions exchanged between C_2 and C_3 but only 1 interaction between C_1 and C_2 .

Then the contribute ψ_{lmj} to the reliability of communication between all these components over the connector i in the scenario j is:

$$\theta_S = 1 - \sum_{j=1}^K p_j \left(\prod_{i=1}^N (1 - \theta_{ij})^{bp_{ij}} \cdot \prod_{(l,i)} (1 - \psi_{lij})^{|Interact(l,i,j)|} \right)$$

2.2. Approach of Rodrigues et al

This approach can also be used in early stages of software development lifecycle for component-based systems. As basic modeling techniques, the scenarios and a Message Sequence Charts (MSC) are used [4]. The framework of the approach shows the Figure 5.

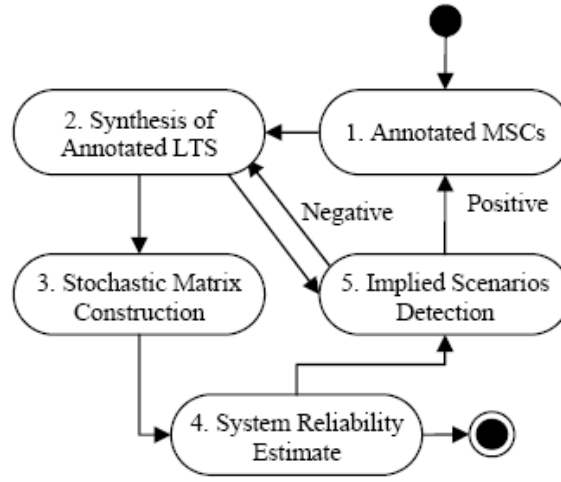


Fig. 5. Framework

First step is annotation the MSCs with two kind probabilities:

- Probability of transitions between scenarios PTS_{ij}
- Reliability of the components R_C .

PTS is simply the probability that scenario S_j will be executed after executing scenario S_i which value comes from an operational profile for the system (Musa, 1993). It is annotated on High-level Message Sequence Charts (HMSC).

The component reliability R_C is the probability that component C will successfully perform its service which is invoked by message from another component. The execution time of the service is not significant for component reliability assessment. Component reliability is annotated on Basic Message Sequence Charts (BMSC).

Second step is to made synthesis of Labelled Transition System (LTS) from annotated scenarios. This step consists of several sub-steps to construct the architecture model of the system.

In the third step the architecture model is interpreted as Markov model. First the mapping of the probability weights from architecture model into a square transition matrix is constructed.

In the fourth step the Cheung approach [1] is used to determine the reliability prediction of the whole system.

In last step, the implied scenarios are searching. These scenarios can be found as the result of the fact that in the system there can exist such set of components that don't communicate exclusively through the

interfaces described and that can exhibit via unspecified traces when running in parallel [6]. Founding of these scenarios has impact on step 1 and 2 of the approach's framework.

3. Conclusions

When comparing these 2 approaches, the first approach [2] evaluates the reliability of the system as whole, while the second one concern only the reliability of a component of the system. Both approaches take into account just little operational profile.

The present research made in this area shows that UML modelling techniques are applicable for reliability assessment of software.

We made also some thesis and assumptions:

- Sequence diagrams will be the core modeling technique for system architecture and behavior
- We assume that developing of new UML extensions will be necessary for reliability assessment
- Our new approach will combine traditional reliability techniques with UML modeling
- The whole process of our reliability assessment approach will be automatic as much as possible
- The new tool which will cover our approach will be developed
- The approach will be developed with focus on control systems software

In this paper we discuss the area of software reliability and its support in UML. We also introduce the today's main approaches and concepts in this area make the short evaluation of them and outlook the basis of our future work in this area.

4. References

- [1] R. C. Cheung. A User-Oriented Software Reliability Model. In: *IEEE Transactions on Software Engineering*, vol. 6(2), 1980, pp. 118–125.
- [2] V. Cortellessa, H. Singh, B. Cukic. Early reliability assessment of UML based software models. In: *Proceedings of the 3rd international workshop on Software and performance*, Rome, Italy, 2002.
- [3] J.D. Musa. Operational profiles in software-reliability engineering. In: *IEEE Software*, vol. 10, no. 2, 1993, pp. 14–32.
- [4] G.N.Rodrigues, D.S.Rosenblum, S. Uchitel. Reliability prediction in model-driven development. In: *Proceedings of the 8th international conference MoDELS*, pp. 339–354, Montego Bay, Jamaica, 2005.
- [5] M. Shooman. *Software Engineering - Design/Reliability/Management*. McGraw-Hill, 1983.
- [6] S.Uchitel, J.Kramer, J.Magee. Incremental “Elaboration of Scenario-Based Specifications and Behavior Models Using Implied Scenarios. In: *ACM Transactions on Software Engineering and Methodologies*, 2004, vol. 13, pp. 37–85.
- [7] A. Zarras, V. Issarny. Assessing Software Reliability at the Architectural Level. In: *Proceedings of the 4th International Software Architecture Workshop (ISAW-4)*, Ireland, 2000.