

Reliability Modeling Using UML

Chokchai Leangsuksun[♠], Hertong Song, Lixin Shen
Computer Science
College of Engineering & Science
Louisiana Tech University
Ruston, LA 71270, USA
Email: {box, hso001, lsh007}@latech.edu

Abstract

System reliability has become an increasingly important benchmark in measuring service continuity. As part of many service level agreements, system performance is gauged by how long it provides service without downtime. It is essential to understand and estimate system reliability during the system design and development phase. Most reliability modeling techniques are based on reliability block diagrams, fault trees, Markov chains and various other elements. Such formalisms may not be familiar to system architects, product managers and software developers. They normally use other high level notations such as Unified Modeling Language (UML) [10] to describe the behaviors and aspects of computing systems. Consequently, there is inevitably a gap between the design process and reliability modeling that system designers are facing. This paper proposes a method, modeling framework and proof-of-concept tool that will attempt to fill the gap. We will then demonstrate our tool that uses UML technology to model two-tier computer system's reliability, obtain the failure/repair rate specified in the UML model, and calculate the system's reliability using the well-known SHARPE tool [8].

Keywords: UML, High Availability, OSCAR, Software Engineering, Multi-tiered Architecture

1. Introduction

System reliability has become an increasingly important factor in evaluating computer systems. In our modern society, computer systems have tremendously impacted us, in telecommunications, financial on-line trades, nuclear plants, automobiles, spacecraft, and other aspects of daily life. System malfunctions may have catastrophic results, causing system downs, financial loss and even human lives.

Reliability is measured by the system's mean time to fail (MTTF). The mean time to failure $MTTF$ of a system is the expected time until the occurrence of the first system failure. Given the system reliability $R(t)$, the $MTTF$ can be computed as,

$$MTTF = \int_0^{\infty} R(t) dt$$

Most reliability modeling approaches are based on statistical methods such as reliability block diagrams, fault trees, and Markov chains. SHARPE tool [8] is based on the aforementioned formalisms. On the other hand, the industrial and academic society has adopted UML a de facto standard modeling language to describe system behaviors and aspects. [5] Most industrial software architects, product managers and software developers often use UML to describe their software design and communicate with each other. However, they may not be familiar with the statistical methods needed for reliability modeling. This situation may result in a disconnection in the design and reliability modeling/estimation process.

In this paper, we propose a method, modeling framework and proof-of-concept tool that attempts to fill the gap. We will then implement and demonstrate our framework concept with a simple tool which uses UML deployment diagram to model a two-tier computing system. By specifying the failure rate and repair rate for components, the tool will be able to process the output file generated from the UML model, then uses this information to construct the statistical fault trees and Markov Chain models, and finally use the SHARPE tool to calculate the system's reliability.

In this initial approach, only the two-tier computing system has been addressed. We are only considering the hardware failure for each entity. The UML deployment diagram for the computing system is mapped to the Faults Trees and the Markov Chain model. Additional reliability modeling and software component failure considerations will be studied in future works.

Figure 1 illustrates our proposed UML-based design and reliability modeling framework. Step (1) UML notations articulate a system design and describe

[♠] This research has been supported by Center for Entrepreneurship and Information Technology (CEnIT), Louisiana Tech University.

component-wise reliability information (e.g. stereotype notion is used to describe the component reliability). (2) Obtains an equivalent XMI representation [11] from the UML model in step 1. (3) Maps the model to the statistical reliability model. (4) Calculates the system's reliability.

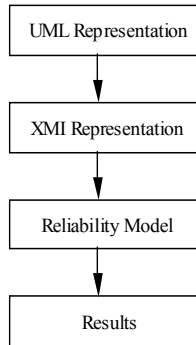


Figure 1 Framework of the UML modeling tool

The remainder of this paper is structured as follows: section 2 shows how to use UML to model system reliability. Section 3 describes passing information from the UML to the SHARPE tool. Sections 4 and 5 show the details of constructing the fault trees and the Markov chain models. Thoughts regarding modeling more sophisticated systems are given in the future works section.

2. Use UML to model system's reliability

We consider using UML to model two-tier computer systems' reliabilities. A two-tier system can be divided into two categories, the front end and back end or head and computing nodes in a cluster environment. An online or e-commerce system, for example, consists of the front end web servers which handle incoming requests from the outside world and business intelligence or database servers which serve as the backend and provide business logic or data repository. We assume (1) each component in the system works independently, and that a failure of one component will not cause the failure of another component; (2) only the hardware failure for each node is considered. The system can be modeled using the UML deployment diagram and is mapped to the Faults Trees and Markov Chains model. Our framework can model more complex (general) multi-tier systems; however, it requires two or more-step approaches whereas we group two-tier sub systems as a new front-end or back-end and then apply the similar technique to calculate the system reliability. More sophisticated automatic system modeling, mapping UML to other reliability models and using UML to describe systems reliability behaviors elegantly will be studied in the future.

Currently, Gentleware's Poseidon [12] for UML community edition's tool is used to create the UML model. By specifying the failure rate and repair rate for each node as tagged values, we will be able to calculate the reliability or the availability of the system either manually or using existing tools. In our approach, SHARPE is used to perform the calculations. A SHARPE configuration file can be generated from the UML model output file in XMI format. We feed the XMI output file to our UML/reliability mapping tool which in turn creates the SHARPE configuration file. Once we run SHARPE with the given input, we then modify the fault trees or the Markov chain models with the SHARPE GUI if needed.

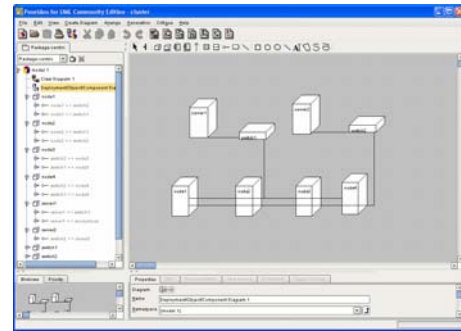


Figure 2 A two-tier system's deployment diagram

Figure 2 demonstrates a system that consists of two servers and four computing nodes in the Poseidon UML tool. Servers are named server 1, server 2 and so on. Nodes are named node 1, node 2 and so on. Servers are connected to switches, which in turn are connected to computing nodes. The failure rate and the repair rate for each node are specified as a tagged name-value pair. For example, if we want to define a failure rate of a server to 0.01 %, we will create tagged name-value pair; name=failure rate, value=0.01, respectively.

Once the model is saved, the Poseidon for UML tool generates a zip formatted file named *zargo*. Inside the *zargo* there is a XMI file that contains the description for the UML model.

3. Parsing reliability information

After finishing a system modeling, an XMI file can be obtained from the output *zargo* file. Failure/repair rates for each node and the relationships between nodes are embedded in this XMI file. These information will then be further processed by parsing the XMI file. Figure 3 shows the workflow from UML modeling to obtain the reliability results.

With this approach, the information such as nodes names, failure/repair rates, and dependency relationships obtained from the XMI file will be transformed into a

SHARPE configuration file. System designers can then use a reliability modeling tool, such as SHARPE, to calculate system reliability. We envision that our future work will include an integrated tool that will take an input from UML model and then calculate system reliability.

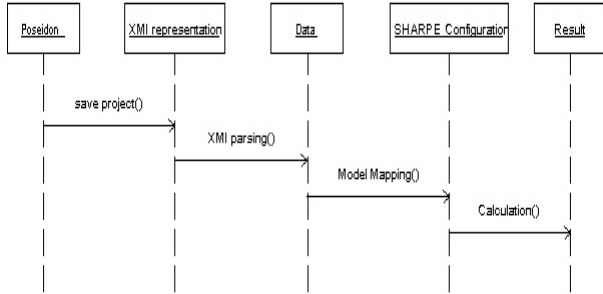


Figure 3 UML based transformation workflow

We have developed a simple tool for proofing of concepts that can create such configuration files from an XMI file. Currently, our tool supports a mapping from UML to two target reliability models, namely Fault tree and Markov-chain. In the next sections, we describe the model methods and mapping algorithms.

4. Construct the Fault Tree Model

Fault Trees is a non-state space reliability modeling method; it represents the probability of failure of the system. It uses Boolean gates to represent the operational dependency of the system on its components. The failure time distribution $F_G(t)$ is computed as:

$$F_G(t) = \begin{cases} \prod_{i=1}^n F_i(t) & \text{AND gate} \\ 1 - \prod_{i=1}^n F_i(t) & \text{OR gate} \end{cases}$$

where $F_i(t)$ is the failure time distribution of component i . The corresponding reliability of each component is $R(t) = 1 - F(t)$.

The mapping from UML representation to fault trees is straight forward at our initial approach. Assume that the system can still provide service if at least a server and a node are working. So the servers (front end) can be bound into a group using an AND gate in the faults tree model, and the nodes (backend) can be bounded into another group using another AND gate. Finally the two servers and nodes gates are bound into an OR gate, and the reliability of this OR gate will be the reliability of the system. Figure 4 shows the faults tree model mapping from the UML model of the previous example given in Figure 1.

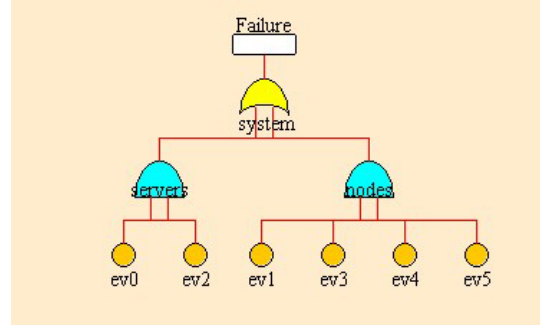


Figure 4 Faults tree model

Once the faults tree configuration file is created, SHARPE tool is used to open the file. Any modifications can be done on the faults tree diagram by editing the GUI, in order to have a more accurate reliability model if there is a need. The reliability for the previous example calculated by SHARPE based on the fault trees model is 50.456% after 1000 hours, assuming the failure rate for each node is 0.001/hr.

5. Construct the Continuous Time Markov Chain Model

No state space modeling methods such as Fault Trees assume stochastic independence between system components. Many intricate system dependencies cannot be adequately represented by these methods. On the other hand, the state space methods such as continuous Markov chains model can be used to handle these kinds of system dependencies.

The mapping from UML representation is as follows. We assume (1) the nodes are independent to each other, meaning that one node failure does not cause other node failures. (2) Once the system is in failure state, the system will not cause more failures. Subsequently, the overall states of the Markov chain are the combinations of servers and working nodes. If we allow servers on the column, working nodes on the row, then the number of available nodes will be reduced sequentially by row and column. Let s denote the number of servers and n denote the number of nodes, then the first row of the transition states would be $(s, n), (s, n-1), \Lambda (s, 0)$, the second row would be $(s-1, n), (s-1, n-1), \Lambda (s-1, 1)$, and the last row would be $(0, n), (0, n-1), \Lambda (0, 1)$. Figure 5 shows the Markov chain states mapping from the previous example described in figure 1.

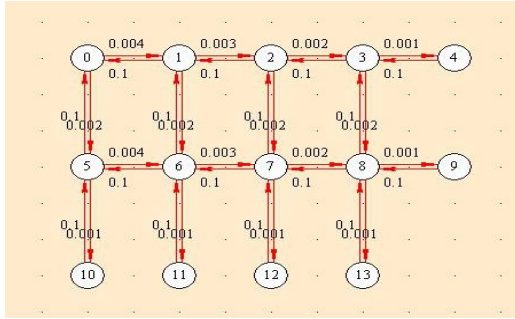


Figure 5 Markov chain model

Once the configuration file is created, SHARPE can open this file and calculate a corresponding system. From an example in the previous section, we obtain system availability, 99.98% after 1000 hours from our framework with SHARPE tool. We assume the failure rate and recover rate for each node is 0.001/hr, 0.1/hr respectively.

6. Future work

We have demonstrated our framework that enables a design notation in UML and calculates the two-tier computer system's reliability. Our method is to transform information in a given UML model into the faults tree and Markov chains model. However, our initial approach only considers hardware failure for each node. We will extend our future to address the follows:

- We will consider using the UML to denote the relationships and dependencies between the components of systems.
- Our framework will include software components in the UML model to reflect the system's reliability.
- We will consider using UML to model systems' dynamic aspects such as fault detection and recovery etc.
- We will consider an integrated modeling framework for multi-tiered computer systems.
- SHARPE is currently used to calculate the system's reliability. We develop a well-integrated tool that automatically transforms UML model and calculates reliability in a single step.

7. Conclusion

Reliability is a necessity that is no longer regarded as a luxury feature. It is a challenging area and service providers are continuously imposing higher expectations on platforms and service availability. System reliability is and continues to be an important part of system design and development. The most popular reliability modeling formalisms are based on block diagrams, faults tree, Markov chains etc. In the software industry, however,

most software architects and developers may not be familiar with these formalisms. We propose a method and framework to model a two-tier computer system using UML and then transform the UML model into reliability models. We successfully demonstrated our framework concepts through a simple tool construction and used it to calculate reliability from a given UML model. We envision that our framework will extend the usefulness of UML and enable architects/designer to obtain system reliability with ease. In the future work, we will consider using UML to model dependencies of components, dynamic aspects of systems and multi-tier computing systems.

References

- [1] Zarras and V. Issarny. UML-Based Modeling of Software Reliability. Proc. of the 1st ICSE Workshop on Describing Software Architecture with UML, May 2001, Toronto, Canada. pp. 36- 40. 2001.
- [2] Zarras and V. Issarny. Assessing Software Reliability at the Architecture Level. *Proceedings of the 4th International Software Architecture Workshop*, Limerick, Ireland, June 2000.
- [3] P. Kahkipuro. A UML Based Performance Modeling Framework for Object Oriented Distributed Systems. In *Proceedings of UML'99*, Oct. 1999.
- [4] R. Pooley. Software Engineering and Performance: A Road-map. In *Proceedings of the 22 International Conference on Software Engineering (ICSE 2000)- The Future of Software Engineering*, pages 189-200. ACM-IEEE-SIGSOFT, June 2000.
- [5] G. Booch, J. Rumbaugh, I. Jacobson. The Unified Modeling Language User Guide. Addison-Wesley. 1999.
- [6] W. Vogels and etc. The Design and Architecture of the Microsoft Cluster Service. FTCS' 98.
- [7] K. S. Trivedi. Probability and Statistics with Reliability, Queuing, and Computer Science Applications. Prentice-Hall. 1982.
- [8] K. S. Trivedi, R. Sahner "Reliability Modeling using SHARPE" *IEEE Transactions on Reliability*, Vol. R-36, No.2, June 1987, pp186-193.
- [9] OSCAR Online document. <http://openclustergroup.org/>
- [10] UML Online document. <http://www.omg.org/uml/>
- [11] XMI Online Document. <http://www.omg.org/xml/>
- [12] <http://www.gentleware.com/>