UML Based Severity Analysis Methodology

A. Hassan, West Virginia University
K. Goseva-Popstojanova, Ph. D., West Virginia University
H. Ammar, Ph. D., West Virginia University

Key Words: Risk Assessment, Severity Analysis, UML, Hazard Analysis

SUMMARY & CONCLUSIONS

According to the NASA Standard on software safety [1], risk is a function of the anticipated frequency of occurrence of an undesired event, the potential severity of resulting consequences, and the uncertainties associated with the frequency and severity.

The authors in [2] developed a methodology for risk assessment of software architectures based on the Unified Modeling Language (UML) [3]. The methodology estimates the probability of software components/connectors failures by measuring the complexity/coupling of the UML dynamic specifications [4]. Severity is estimated based on the MIL_STD_1629A [5] and using the classical technique of Failure Mode and Effect Analysis (FMEA). In this paper we address the problem of assessing the severity based on UML artifacts and using the cost of failures of software components and connectors as well as failures of system execution scenarios. We propose a severity assessment methodology which is performed combining three different hazard analysis techniques: Functional Failure Analysis (FFA), Failure Mode and Effect Analysis (FMEA), and Fault Tree Analysis (FTA). The methodology integrates these techniques in order to assess the severity of failures of system scenarios and the severity of failures of each architectural element (component/connector) early in the software analysis and design phases. FFA is used as a top-down approach based on system scenarios to identify the system level failures. FMEA is used as a bottom-up appraoch based on the detailed view of the system to identify the possible causes of component/connector failures. Finally, FTA correlates the results of FMEA and FFA. This process of estimating severity can be automated in development environments supporting UML by annotating the hazard analysis results and the cost of failure information in the UML diagrams. We use this methodology for reliability-based risk assessment [2], performance-based risk assessment [6], and requirement-based risk assessment of software systems [7].

1. INTRODUCTION

Severity assessment is a procedure by which the severity of failures of software architectural element (component/connector) is estimated and ranked accordingly to the consequences of these failures. In the US military standard [5], severity considers the worst case consequence of a failure, determined by the degree of injury, property damage, system damage, and mission loss that could eventually occur.

Considering the severity of software failures will help in allocating development and testing resources. Some software components may be tested more intensively than others based on the severity of failure weighted by the probability of failure. In [2], we proposed an architectural level software risk assessment methodology. This methodology combined the probability of software failure with the severity of this failure to estimate the risk factor of software architectural element early on the software design phase. The probability of failure is estimated based on the work done in [4], while severity is estimated using FMEA. Sherer in [8] proposed a methodology to estimate the consequences of software failure caused by faults in different software modules. Sherer used the hazard analysis technique FTA and software operational profile to estimate the cost of failure for every software module. This is a complex process to be automated and it was applied at the code level. Yacoub et al [9] used FMEA to assess the severity of software (components/connectors) failure.

Using one hazard analysis technique for severity analysis usually fails to offer a coherent and complete picture of the ways in which low-level component failures contribute to hazardous malfunctions of the system. Hazard analysis techniques assume different design representations, which reflect different levels of abstraction in the system design. While, for example, FFA requires only abstract functional descriptions, HAZard and OPerability analysis (HAZOP) [10] and FMEA require architectural designs of increasing detail and complexity. As shown by Allenby and Kelly in [11] it is not enough to only use one severity analysis technique in complex systems. Often a combination of more than one technique should be used to get a more complete understanding of the system. The suitability of UML to be the specification language for severity analysis using more than one classical hazard analysis method was briefly discussed in [12]. This survey paper paves the way to the work presented here. In this paper we propose a methodology for severity analysis of software systems at the early phases of development based on UML. The proposed methodology enables the integrated severity assessment of complex software systems from system level hazards to the low level hazards of component/connector failure modes. This paper is organized as follows. The proposed methodology is presented in Section 2 and illustrated at case study in Section 3.

2. THE PROPOSED SEVERITY ANALYSIS METHODOLOGY

The proposed severity analysis process (Figure 1) starts with FFA, which uses system level scenario diagrams (Figure 3) as an input to identify all system level hazards [13]. This high level FFA analysis provides a comprehensive view of the ways in which the system could fail. System level failures arise as a result of failures or malfunctions of lower level components/connectors. Therefore we apply FMEA as the second step of the process, at the level of components and connectors using UML sequence diagrams to determine their failure modes and cost of failure for each failure mode.

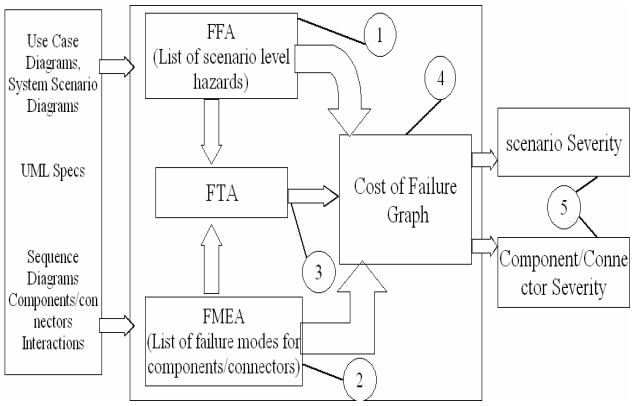

Figure 1. Schematic diagram of severity analysis methodology

We use FTA as a third step to define a relationship between failures of individual architecture elements (component/connector) and a failure of the system. The system hazards identified in step 1 are used as top events in the fault tree, while the basic events are the failure modes identified in step 2. The step 4 of this process is to develop the cost of failure graph [8] to estimate cost of failure for each execution scenario and every component/connector in the scenario. The final step is to map estimated cost of failure of scenario and each component/connector to a severity rank using cost severity graph which is introduced in [14]. The steps of the methodology for a given scenario are summarized as follows:

1. Identify system hazards (states of the system that can contribute to accidents and mishaps) by performing FFA [15].
2. Identify components/connectors failure modes performing FMEA [5].
3. Construct a detailed cause and effect model that records how failures propagate from components/connectors level through the system level by using FTA. (This step combines the outputs from step 1 and step 2.)
4. Develop the cost of failure graph to estimate cost of failure of a given execution scenario and each component/connector in this scenario [8].
5. Estimate the severity of each component/connector and system scenario using cost of failure graph [8] and cost severity graph [14].

2.1 Functional Failure Analysis (FFA)

Figure 2 shows a UML use case diagram for a system *S*, where actors Act_1 performs two use cases Uc_1 and Uc_2 through associations Ass_1 and Ass_2, and actor Act_2 performs the use case Uc_1 through Ass_3. Figure 3 shows a high level annotated system sequence diagram which describes one scenario of the use case Uc_1 showing the interactions between actors Act_1, Act_2 and the system through input and output events. Events like E_{11s}, E_{2s1}, are the events between the system *S* and Act_1. Event E_{1s2}, is the event between the System *S* and the Act_2. The system states are S_1, S_2, *and* S_3, which are the states of the system after receiving or sending an event to the external actors (Act_1, Act_2). The input events (E_{11s},) in Figure 3 represent external events that stimulate responses from the system. The output events (E_{2s1}, E_{1s2}) represent the externally observable behavior of the system.

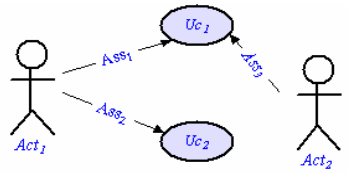

Figure 2 A use case diagram for a system *S*

The process starts with applying FFA on the system scenario diagram shown in Figure 3. We perform FFA using guide words defined in [15] to identify possible failure modes for each event between the system and the actors (E_{11s}, E_{1s2}, E_{2s1}). The events are systematically examined for potential hazards, which include the loss of event, the unintended delivery of event, and event malfunctions. The analysis considers each event in turn and decides whether or not these *hypothetical* failure modes are credible and, if they are, what the consequences might be. This gives a clear view of how the failure of these events could contribute to hazards and accidents of the scenario. The input to FFA is a list of events from the system level scenario, list of guide words, and cost of failure for every class of failure. The output of FFA is a tabulated form (see Table 2).

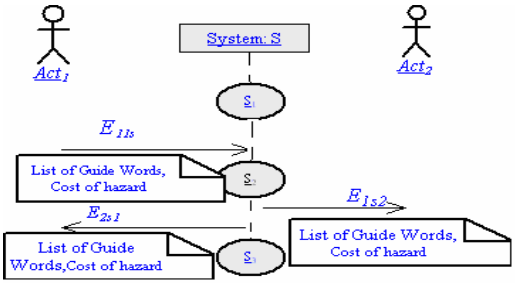

Figure 3 The sequence diagram of use case UC_1 for system *S*

2.2 Components/connectors Failure Modes

FMEA examines how component/connector could fail considering component/connector malfunctions. It generates a failure model for the components/connectors under examination; it is essentially a tabular process [14]. During specific scenario components interact with each other by exchanging messages. Each of these interactions links a component that requests an operation with a component that performs the operation. All interactions and component behavior are captured in sequence diagrams. FMEA is applied for each component/connector within the sequence diagram.

The behavior of each component could be captured with the component state diagram during this scenario. The component changes its states through interactions based on message exchange. A hazard occurs from an unwanted interactions (or events). Each of the unwanted events in the sequence is either due to a message being sent incorrectly by the sender component, or the message not being acted on correctly by the receiver component, or the connector acting incorrectly.

These events can be generated by sender or receiver state transitions. Therefore, faults in component state or transitions can give reasons for a component/connector failure [16]. It is necessary to confirm that under correct behavior of components/connectors, the system does not allow the occurrence of hazards. That is, if the components in the system correctly generate the intended messages, are in the correct state, and connectors transmit correct messages, then the system is safe. This means that no failure will happen to components/connectors. In order to identify possible faulty behaviors for the components, we apply FMEA to the states of the components. We identify hazards associated with each component, detail all possible failure modes, and identify their resulting effect on the system. The output of this process is in a tabulated form (see Table 3).

A connector is defined as the interface between two components [9]. The connector transmits the messages between the components. By applying FMEA on connectors using the messages transmitted through these connectors, we can identify connectors failure modes and the effect of these failures on the system. We identify hazards associated with each connector, detail all possible failure modes, and identify their resulting effect on the system.

2.3 Fault Tree Analysis

FTA is a top-down method used to identify failure causes [17]. FTA is primarily used for analyzing causes of hazards, not identifying hazards. The process of analyzing causes is documented in one or more fault trees. FTA depicts logical interrelationships of the basic events that may lead to a particular undesired event. FTA is used for addressing low level failure conditions (basic events) and their potential effect for causing the top level hazards (top events) [13]. Failure of components/connectors (low level) will propagate to the system level (higher level). We use FTA to map system level hazards (output from FFA) to components/connectors failure modes (output from FMEA). The top events of the fault trees are the system level hazards and the basic events are the components/connectors failure modes.

2.4 Cost of Failure Graph

Kmenta in [14] described failure scenarios as "*undesired cause-effect chains of events, from the initiating cause to end effect, including all intermediate effects*". Each failure scenario happens with some probability and results in negative consequences. FTA is considered as a cause effect model in [18] consisting of many cause effect chains with probabilities for each cause and effect. These cause-effect chains relate the system level hazard identified from FFA to components/connectors failure modes identified from FMEA. Considering these cause-effect chains as failure scenarios for the system, we could estimate the cost of failure of each component/connector based on these failure scenarios.

Cost is an accepted measure of consequences [19]. Expected cost is used extensively in the fields of risk analysis, economics, insurance, and decision theory. Kmenta and Ishii [14] proposed an adaptation of FMEA considering the consequences of the failures in terms of costs. Cost is a universal language understood by engineers without ambiguity. We develop cost of failure graph proposed in [8] for every component/connector and scenario to estimate cost of failure of every component/connector and scenario. For a specific component/connector, there is more than one failure scenario. The expected cost of failure for component/connector is the sum of all costs over these scenarios weighted by the probability of each failure scenario.

We develop component/connector cost of failure graph [8] to estimate the component/connector and scenario cost of failure using annotated UML sequence diagrams representing the interactions of components and using FFA and FTA analysis. During the execution of a system scenario S_x there are many hazards for this scenario. These hazards and their consequences are identified in step 1 using FFA technique. In step 3 we estimate the probability of each of these hazards. The expected cost of failure of system scenario may be estimated by summing the expected use of the scenario, weighted by the expected consequences of all hazards that may be result from the usage of this scenario. Using the probability of usage of the scenario [2], probability of hazards, and cost of these hazards for this scenario (results from step 1 and step 3) we estimate the cost of failure of this scenario using the cost of failure (see Figure 9).

Definitions:

- ${}^{x}Cost_i(M)$ is the cost of failure of (component/connector) i in a given failure mode M in a given system scenario S_x,
- ${}^{x}p_i(M)$ Is the probability of component/connector) i being in failure mode M in a given system scenario S_x.
- ${}^{x}p(H)$ Is the probability of system level hazard H for a given system scenario S_x,
- ${}^{x}Cost(H)$ is the cost of failure for a given system hazard H in a given scenario S_x.
- ${}^{x}p(S)$ is the probability of execution of a given scenario S_x.

Total expected cost of failure of (component/connector) i in a given system scenario S_x is as follows:

$$ {}^{x}TotalCost_i = \sum_{k=1}^{k=H} {}^{x}p(k) \sum_{j=1}^{j=M} {}^{x}p_i(j) \bullet {}^{x}Cost(j) \quad (1) $$

The total expected cost of failure of a given scenario S_x is estimated as follows:

$$ {}^{x}TotalCost(S) = {}^{x}p(S) \sum_{k=1}^{k=H} {}^{x}p(k) \bullet {}^{x}Cost(k) \quad (2) $$

2.5 System Scenario and Components/Connectors Severity

In [14] Kmenta proposed cost of failure metrics, which he used to estimate consequences of failure and map cost of failure on a 0.1-1.0 severity rank. Using cost-severity graphs (Figure 10) we map the expected cost of failure of

component/connector, as well as system scenario to severity rank. The cost-severity curve is dependent on the application domain. For example, in health care the cost of out patient care would have a severity rank between 0.1-0.3, whereas in patient care would have a severity rank between 0.4-0.6, followed by the severity of intensive care from 0.7-1. In general, the cost-severity relationship is nonlinear.

3. CASE STUDY

We have selected a case study of a cardiac pacemaker device to illustrate the proposed methodology

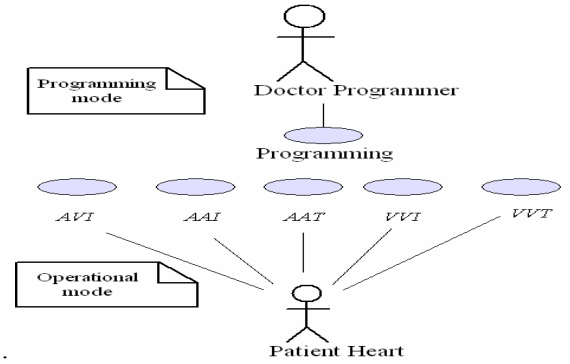

Figure 4 Pacemaker use case diagram

A cardiac pacemaker is an implanted device that assists cardiac functions when the underlying pathologies make the intrinsic heart beat low [20]. An error in the software operation of the device can cause loss of a patient's life. This is an example of a critical real-time application. We use the UML real-time notion to model the pacemaker. The use case diagram of a pacemaker is shown in Figure 4. A pacemaker can be programmed to operate in one of five operational modes (*AVI, AAI, AAT, VVI, and VVT*) depending on which part of the heart is to be sensed and which part is to be paced. It runs in either a programming mode or in one of five operational modes. The pacemaker application is modeled as six use cases and two actors namely *doctor programmer* and *patient's heart*. Each use case is realized by one sequence diagram (i.e., scenario). Domain experts determine probabilities of occurrence of use cases and the scenarios within each use case. This can be done in a similar fashion as the estimation of the operational profile in the field of software reliability [21]. According to [20] the inhibit modes are more frequently used than the triggered mode. Also, the programming mode is executed significantly less frequently than the regular usage of the pacemaker in any of its operational modes. Hence, we assume the probabilities for programming use case and the five operational use cases as given in Table 1 [2].

3.1 FFA Analysis

Figure 5 shows system level scenario diagram for the *AVI* mode of operation. The system received *Programming Command* event from the *programmer* actor to operate in *AVI* mode. To monitor the heart the system receives *VSense* event from the heart actor and handles it. The system begins pacing the heart by sending signals (*Pace* event) to the heart actor. Table 2 shows part of the FFA table obtained using FFA with guide words as explained in section 2.

Use case	*Programming*	*AVI*	*AAI*	*VVI*	*AAT*	*VVT*
Probability	0.01	0.29	0.20	0.20	0.15	0.15

Table 1 Probabilities of the use cases executions

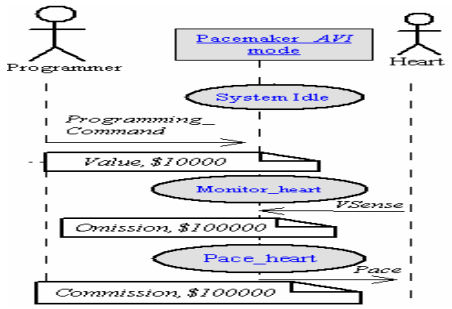

Figure 5 System Scenario Diagram of *AVI operation* mode

3.2 FMEA Analysis

In the sequence diagram shown in Figure 6, the *VT* component monitors the heart. When a heart beat is not sensed, the *AR* component paces the heart and a refractory period is then in effect. Table 3 presents a part of the FMEA table for *AR* component. Applying FMEA on every component by tracing states and transitions for every component from its state diagram we come up with the FMEA result. Also, we apply FMEA for every connector by tracing all messages transmitted over the connector. Due to space limitations, we show the results concerning the *AR* component only.

3.3 FTA Analysis

Figure 7 shows the fault tree of the top event "*Commission*" of "*Pace*" hazard as a function of components/connectors failure modes. FTA (step 3) combines the results from FFA (step 1) and FMEA (step 2) to map the *Commission "Pace*" hazard to its basic failure modes. Using the probabilities of the basic events, which are determined in step 2, we estimate the probability of top level events.

3.4 Component/connector and scenario cost of failure graph

The first level of the *AR* component cost of failure graph shown in Figure 8 is the top events of all fault trees with their probabilities. Every component/connector could contribute to these hazards during the execution of the scenario. The component/connector contribution to these hazards results from the component/connector failure modes. To estimate the cost of failure for each component/connector in a scenario we develop a cost of failure graph which combines the cost of component/connector failures for all relevant hazards.

During the intended use of the *AVI* scenario there are several system level hazards. The output of the FFA is the list these of possible hazards. Every hazard is represented by a top event in a single fault tree. As shown in Figure 5, the *AVI* scenario event *Programming_Command* is used to initialize the system (*Programmer* actor programs the pacemaker to work in *AVI* mode). Event *VScense* is used to monitor the heart (pacemaker receive signal from *Heart* actor). Also *Pace* event is used to pace the heart (pacemaker pace the heart). The

probability of usage of *AVI* scenario is given in Table 1. Using this probability of usage with the results from step 1 (list of system level hazards, cost of hazards) and results from step 3 (probability of the system level hazards), we could estimate the cost of failure of the scenario. To implement this we use the cost of failure graph in Figure 9 and equation (2).

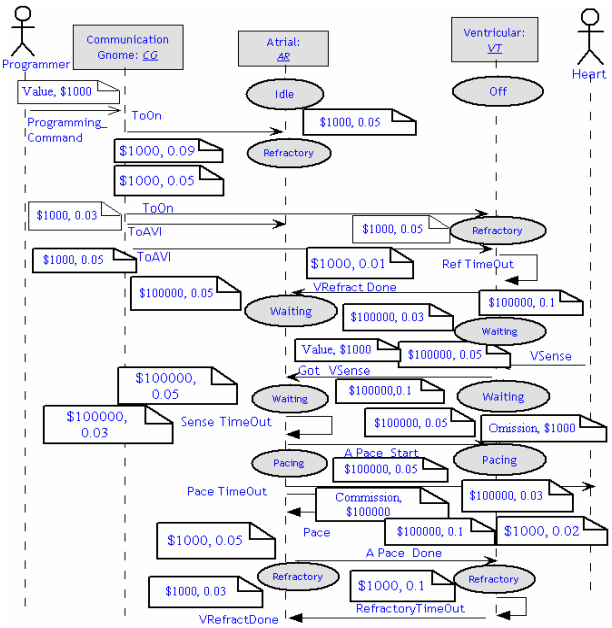

Figure 6 Sequence diagram of the *AVI* scenario

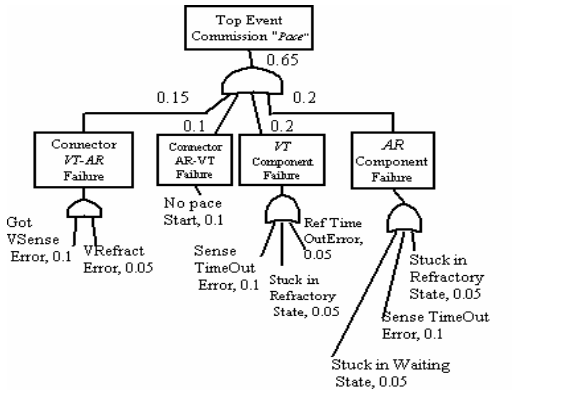

Figure 7 Commission "*Pace*" Fault Tree

3.5 Components/connectors and scenario severity

In the final step of the methodology, we use a cost severity-graph (Figure 10) to determine the severity rank for each component/connector, as well as the scenario. For the *AVI* scenario this is done by extending point A in y axis which gives the total cost of failure of the scenario, to meet the cost-severity curve at point B.

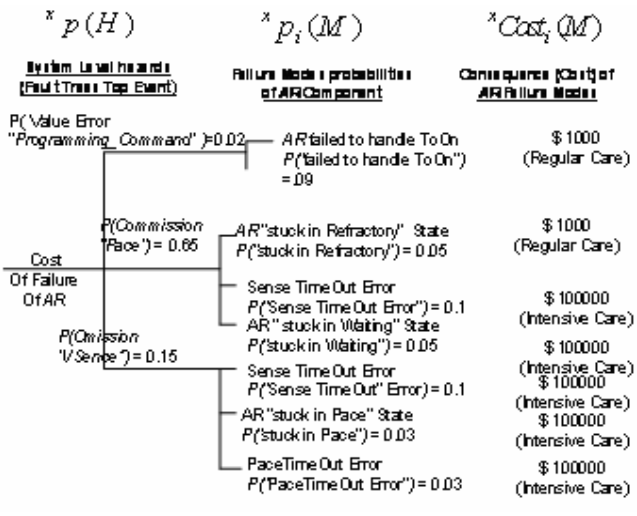

Figure 8 Cost of failure graph of the *AR* component

We extend point B to meet the severity scale in the x axis at point C. Point C gives the severity value associated with the scenario failure. Table 4 shows the results of the final step of the methodology after mapping the cost of failure of each component/connector to severity rank. Next, we map the estimated cost of failure of *AVI* scenario to severity rank using cost-severity graph, which gives 0.95 as a severity rank for *AVI* scenario. The results from Table 4 show that the *VT* and *AR* components are components with the highest severity in the *AVI* scenario. This result is intuitive since these two components are the most active and the most critical components that directly control the operation of the heart during the scenario.

The *CG* component, on the other hand, controls the programming operation and it is monitored by the physician before the device is put into operation. Also from Table 4, we identify that the connection between the *VT* and *AR* components (*AR-VT*, *VT-AR* connectors) are the highest severity connectors. This result is also intuitive in the context of the pacemaker example since these connectors deliver critical messages controlling the heart operation such as sensing and pacing. Results for *AVI* scenario show that the *AVI* scenario is a high severity scenario because it is controlling the pacing operation of the heart. The worst consequence of failure of this scenario could lead to patient's death.

Event Name	Class of failure	Failure	Effects on System	Cost of failure($)	Comments
VSense	*Omission*	Timer not set correctly	No pace is generated for the heart, patient will require intensive care	1000,00	Timers does not work well
Pace	*Commission*	Pacing hardware device malfunctioning	Heart is always paced while patient condition requires only pacing the heart when no pulse is detected	1000,00	Sensor failed to sense the heart.

Table 2. Part of the FFA for *AVI* scenario presented in Figure 5

Component	Failure Modes	Effect on the system	Cause of failure	Cost of Failure $
AR	*ToOn Value Error*	The component will not work and there is no pace of the heart	The component does not receive signal from CG	1000
-	*Sense TimeOut Error*	Heart operation is irregular because it receives no pacing	The component sensor does not work well.	100000

Table 3. Part of the FMEA for *AR* component

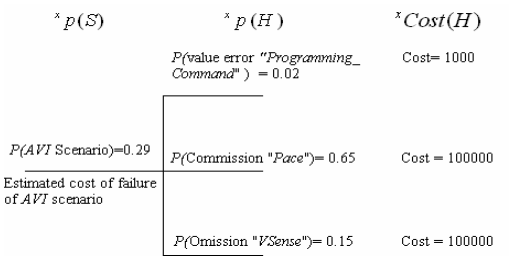

Figure 9. Cost of failure graph of *AVI* scenario

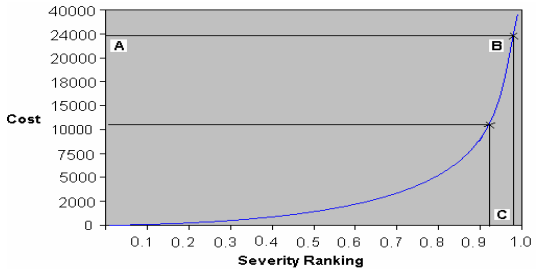

Figure 10. Cost-severity graph

Component/Connector name	Severity
CG-AR	0.50
CG-VT	0.50
AR-VT	0.94
VT-AR	0.95
CG	0.50
AR	0.96
VT	0.95

Table 4. Components/connectors severity in *AVI* scenario

The severity assessment is part of any risk assessment methodology. Thus, the work presented in this paper can be used for estimating severity of component/connector failures in reliability-based risk assessment methodology [2], and severity of scenario failures in performance-based risk assessment [6], and requirements-based risk assessment [7].

ACKNOWLEDGEMENT

This work is funded in part by grants to West Virginia University Research Corp. from the National Science Foundation Information Technology Research (ITR) Program grant number CCR-0082574, and from the NASA Office of Safety and Mission Assurance (OSMA) Software Assurance Research Program (SARP) managed through the NASA Independent Verification and Validation (IV&V) Facility, Fairmont, West Virginia.

REFERENCES

1. NASA-STD-8719.13A,"Software Safety NASA Technical Standard", SEP., 15, 1997.
2. K. Goseva-Popstojanova, A. Hassan, A. Guedem, W. Abdelmoez, D. Nassar, H. Ammar, A. Mili, "Architectural-Level Risk Analysis using UML", *IEEE Transactions on Software Engineering*, Vol. 29, No. 10, 2003, pp 946-960.
3. OMG Unified Modeling Language Specification Version 1.4, September 2001, http://www.uml.org.
4. Ahmed Hassan, Walid M. Abdelmoez, Rania M. Elnaggar, Hany H. Ammar, "An Approach to Measure the Quality of Software Designs from UML Specifications," *7th Int'l conference on information systems, analysis and synthesis ISAS*, July. 2001, pp 559-564.
5. US Military Standard, "Procedures for Performing Failure Mode Effects and Criticality Analysis", *US MIL_STD_1629A / Notice 2*, Nov. 1984.
6. V. Cortellessa1, K. Goseva-Popstojanova, K. Appukutty, A. Guedem, A. Hassan, R. Elnaggar, W. Abdelmoez, and H. H. Ammar, "Performance-based Risk Analysis of UML Models", *submitted for publication.*
7. K. Appukutty, Hany H. Ammar, Katerina Goseva Popstajanova, "Software Requirement Risk Assessment Using UML", The 3rd ACS/IEEE International Conference on Computer Systems and Applications (*AICCSA),* January ' 05.
8. Susan A. Sherer, "Methodology for the Assessment of Software Risk", Ph.D. Diss., Wharton School, University of Pennsylvania, 1988.
9. Sherif M. Yacoub, Hany H. Ammar, "A Methodology for Architectural-Level Reliability Risk Analysis ", *IEEE Transactions on Software Engineering, Vol.* 28, No 6, 2002, pp 529 - 547.
10. McDermid, J.A., Nicholson M., Pumfrey D.J., Fenelon P., "Experience with the application of HAZOP to computer-based systems", *IEEE, COMPASS '95: Proceedings of the Tenth Annual Conference on Computer Assurance*, 1995, Gaithersburg, MD, pp 37-48.
11. Karen Allenby, Tim Kelly, "Deriving Safety Requirements Using Scenarios", *5th IEEE Int'l Symposium on Requirements Engineering*, August 27-31, 2001, Toronto, Canada, pp 228-236.
12. A. Hassan, K. Goseva-Popstojanova, H. Ammar, "Methodology for Architecture Level Hazard Analysis, A Survey", *ACS/IEEE Intl. Conference on Computer Systems and Applications (AICCSA'03*), Tunis, Tunisia, July 14-18, 2003, pp 68-70.
13. Per Johannessen, Christian Grante, Anders Alminger, Ulrik Eklund Jan Torin, "Hazard Analysis in Object Oriented Design of Dependable Systems", *Proceeding of the 2001 Int'l conference on Dependable Systems and Networks,* 2001, Goteborg, Sweden, pp 507-512.

14. Steven Kmenta, "Scenario- Based FMEA: A Life Cycle Cost Perspective", *Proceedings of DETC 2000, ASME Design Engineering Technical Conferences*, September 10 - 14, 2000, Baltimore, Maryland, pp 270-281.
15. 15.Yiannis Papadopoulos, John A. McDermid, "Hierarchically Performed Hazard Origin and Propagation Studies", *18th Intl Conference on Computer Safety, Reliability and Security (SAFECOMP'99)*, 1999, Lecture Notes in Computer Science, 1698, pp 139-152.
16. P. A. Lindsay and J. A. McDermid. "A systematic approach to software safety integrity levels", *Proceedings of 16th International Conference on Computer Safety, Reliability and Security (SAFECOMP'97)*, 1997, pp 70-82.
17. Kirsten M. Hansen, Anders P. Ravn, Victoria Stavridou, "From Safety Analysis to Software Requirements", *IEEE Transaction on Software Engineering, Vol.* 24, No. 7, July 1998, pp 573-584.
18. Shaoying Liu, "Verifying Formal Specifications Using Fault Tree Analysis", *Int'l Symposium on Principles of Software Evolution*, Nov. 1-2, 2000, Kanazawa, Japan, pp 271-280.
19. Gilchrist, W. "Modeling failure modes and effects", *International Journal of Quality and Reliability Management,* Vol. 10, No. 5, 1993, pp 16-23.
20. B. P. Douglass, "Real-Time UML: Developing Efficient Objects for Embedded Systems", 2nd Edition, Published by Addison Wesley, ISBN: 0201657848, 2000
21. J. D. Musa. "Operational profiles in software reliability engineering", *IEEE Software*, Vol. 10, No. 2, , Mar. 1993, pp 14–32.

BIOGRAPHIES

Hany Ammar, Professor,
Lane Dept. of Computer Sc. and Electrical Engineering
West Virginia University, 739 Engineering Sc. Building,
Morgantown, WV 26506 USA

E-mail: Hany.Ammar@mail.wvu.edu

Hany Ammar is a Professor of Computer Engineering in the Department of Computer Science and Electrical Engineering at West Virginia University. Dr. Ammar has published over 80 articles in prestigious journals and conference proceedings. Dr. Ammar has been teaching in the area of Software Engineering since 1987. Dr. Ammar has been recently a Principal Investigator on a number of research projects on software verification and validation. Dr. Ammar is a member of the IEEE and the ACM professional organizations.

Katerina Goseva-Popstojanova, Assistant Professor
Lane Dept of Computer Science and Electrical Engineering
West Virginia University, Morgantown, WV, 26506 USA

E-mail: katerina@csee.wvu.edu

Katerina Goseva-Popstojanova is an Assistant Professor in the Lane Department of Computer Science and Electrical Engineering at West Virginia University, Morgantown, WV. Her research interests include software reliability engineering, dependability, performance and performability assessment, and computer security and survivability. She has published over 40 journal and conference articles on these topics. She is a Senior Member of the IEEE, and member of the ACM.

Ahmed Hassan
Lane Dept. of Computer Sc. and Electrical Engineering
West Virginia University, 219 Engineering Research building
Morgantown, WV 26506 USA

E-mail: hassan@csee.wvu.edu

Ahmed Hassan is a Ph. D student in West Virginia University, WV, USA. Hassan is a Graduate Research Assistant working with Dr. Hany Ammar, WVU. His research interests are software hazard analysis, software metrics, and software risk assessment. Hassan is a member of IEEE, and ACM.