Transactions of the Korean Nuclear Society Spring Meeting
Jeju, Korea, May 7-8, 2015

Quantification of Safety-Critical Software Test Uncertainty

M. Khalaquzzaman, Jaehyun Cho*, Seung Jun Lee, Wondea Jung
Integrated Safety Assessment Division, Korea Atomic Energy Research Institute
1045 Daedeok-daero, Yuseong-gu, Daejeon, 305-353, Korea
E-mail: kzaman74@gmail.com; sjlee@kaeri.re.kr; wdjung@kaeri.re.kr
**Corresponding author: chojh@kaeri.re.kr*

1. Introduction

Software failure probability is quantified based on test results. Usually, the types of testing performed for software reliability estimation are the debug testing, operational testing, so on. The operational profile-based testing is recognized to be more effective and economical [4] than other testing options. In operational profile-based testing, the test cases are prioritized based on the probability of uses [2, 18]. An input profile-based testing follows the operational profile in which the most probable input must be tested first, and the least probable input comes in the last.

The input-profile based testing for quantification of the failure probability of NPP safety critical software was proposed in [1]. The method, conservatively assumes that the failure probability of a software for the untested inputs is 1, and the failure probability turns in 0 for successful testing of all test cases. However, in reality the chance of failure exists due to the test uncertainty.

Some studies have been carried out to identify the test attributes that affect the test quality. Cao discussed the testing effort, testing coverage, and testing environment [3]. Management of the test uncertainties was discussed in [5, 6]. BBN based modeling for evaluating the software engineering uncertainty was proposed in [7, 8].

In this study, the test uncertainty has been considered to estimate the software failure probability because the software testing process is considered to be inherently uncertain. This paper discusses the software test uncertainty quantification employing a Bayesian belief network. An example for quantification of RPS software failure probability considering the uncertainties is presented.

2. Uncertainties in Software Testing

Software test activities are performed in a computing environment to identify the discrepancies between the actual capability and the desired capability of a software [9, 7]. These activities are mainly categorized into two groups, test planning and test enactment. Although both categories of tasks are known to be error prone, the test enactment is identified to be inherently uncertain because the numbers of test cases are usually limited and the test environment is not ideal [8].

The uncertainties introduced by the elements of the automated test activities are the intended system to be tested, the test platform, the test cases, the tools, and the administrator [10]. Because uncertainties emerge from multiple factors relevant to the test activities and environment, they should be taken into account for quantification of the software test reliability. The major causes of test uncertainty are discussed below.

- *Test planning activities*

Software test planning is usually carried out at the early stage of the software development phase. A software test plan includes all of the necessary testing activities, allocation of roles, responsibilities, and resources including the overall schedule [16]. Test planning is error prone because the tasks are mostly human intensive. The identified test planning tasks from which the uncertainties are emerged are artifacts, planned test activities, and the plans themselves.

- *Selection of Test Cases*

"Test cases are a set of test inputs, executions and expected results developed for their objectives such as to exercise a particular program path or to verify compliance with a specific requirement" [11]. The test case selection is the activity of choosing a finite set of elements to be tested out of a typically infinite number of test elements. A test case is to determine the appropriateness of the software features [12, 8, 13].

Test cases should be described in a simple manner so that the team members do not become confused during the execution and finding any defects. The effective test case developing techniques are a gradual learning process. Extensive experience in addition to in-depth understanding of the program structure is desired to achieve the skills of test case development. Thus, plenty of domain knowledge, technology knowledge, and testing practices are essential for designing and generating effective test cases [13].

It can be recognized that the uncertainty emerged from the development of test cases are influenced by (i) the selection of finite test cases among an infinite number of test elements [8], and (ii) the shortcomings/limitations of test case designers.

- *Test Execution*

Each of the selected test cases is manifested through the execution of the software by employing test systems and examining the test results to see whether a test passed or failed. The proper execution of testing is essential for achieving the objectives of software testing. The test environment should be identical to the live environment for an effective execution of the test cases [3].

- *Software Test Environment*

The software test environment plays a vital role in achieving the goal of testing. For an effective testing, the test environment should be identical to the live environment [3]. However, in reality, the test environment mostly differs from the actual operating environment.

The degree of difference between the software test environment and actual operating environment increases the uncertainty in successful testing. Although uncertainty control is a major goal in software quality assurance, the full control of execution uncertainty remains unfeasible for the complex units of the software under testing. In particular, the aleatoric uncertainties cannot be precisely determined. The epistemic uncertainties can be resolved by spending adequate effort [6]. The aleatoric uncertainties are related to natural variability and the epistemic uncertainties are related to the lack of knowledge [14]. It becomes very difficult to rely on the test results when significant uncertainty exists. The development of sophisticated oracles is considered to be a complementary way of dealing with uncertainty.

- *Software Test Resources*

Software test resources play an important role to achieve a quality testing. The major components of software test resources include the test equipment and human resources.

- Test Tools

The development or selection of an appropriate tool for testing of the software is essential to effectively conduct the testing. The major factors relevant to the test tools that introduce uncertainty are incompatibility and complexity of the tools for the intended software testing, test platform, and recognition of technology.

- Competent staff

The number of persons available and their education, knowledge, experience, training and expertise play key roles for an effective test design and execution of the test cases. The deficiency in the qualification of the testing staff can introduce a significant level of uncertainty over successful testing software.

3. Uncertainty Quantification

The proper assessment of test uncertainty leads to a better estimation of the software failure probability. Quantifying the test uncertainties through an evaluation of the entire testing process, test platform, the tools, test resources and environment is necessary. The BBN, through this study, has been proposed for software test uncertainty quantification more explicitly considering the underlying factors that influence the test quality. Research reports, regulatory standards and guidelines are reviewed to identify influential attributes that can affect the software test quality. For instance, the US NRC report BTP-7-14 (2007) [15] discusses the characteristics of the software development life cycle process and can be used for identifying important attributes. The major attributes related to the testing are presented in Table 1. For a consideration of the set of weighting factors, it will be worthwhile to develop the model reflecting the impacts of the attributes differently because in reality the attributes have different characteristics and influence the quality of testing.

Table 1 Attributes of software test quality

Attributes	Weighting Factor
Test Activities	
- Planning	w_{a1}
- Management	w_{a2}
- Measurement	w_{a3}
- Test case development	w_{a4}
- Test execution	w_{a5}
- Result checking	w_{a6}
Test Resources and Environment	
- Tools (hardware and software)	w_t
- Environment	w_e
Human Resource	
- Education	w_{ed}
- Knowledge	w_k
- Training	w_t
- Experience	w_{ex}

Example

A BBN model consisting of the software test quality attributes was constructed and the test uncertainty was quantified based on the assumed input data. The example model consisting of two interconnected subnets are shown in Fig. 1 – Fig. 3. In this calculation, it was assumed that the 10% deviation of the attributes from the desired level would lead to a complete test failure. Our model shown in Fig. 1 represents the structure of a BBN for an evaluation of the test activities and Fig. 2 represents the structure of a BBN for the test resources and test environment. The structure of the net to estimate overall uncertainty is shown in Fig. 3.

The commercial BBN tool, AginaRisk, was employed for constructing the example network in which the attributes were represented as ranked nodes. Ranked nodes are used for measuring the states of discrete variables on a subjective scale, i.e., low, medium, and high. The states of ranked nodes are expressed on an ordinal scale of 0-1[19, 17].

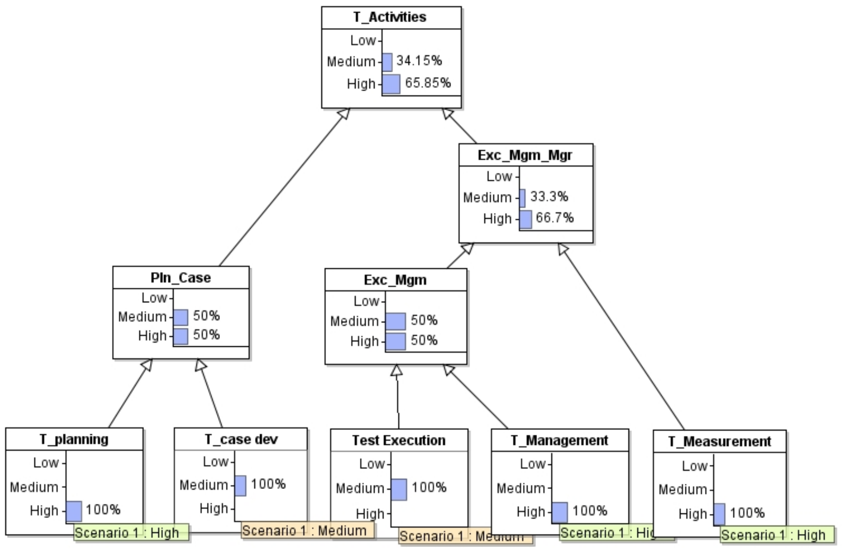

Fig. 1 Model for assessing of software test activities

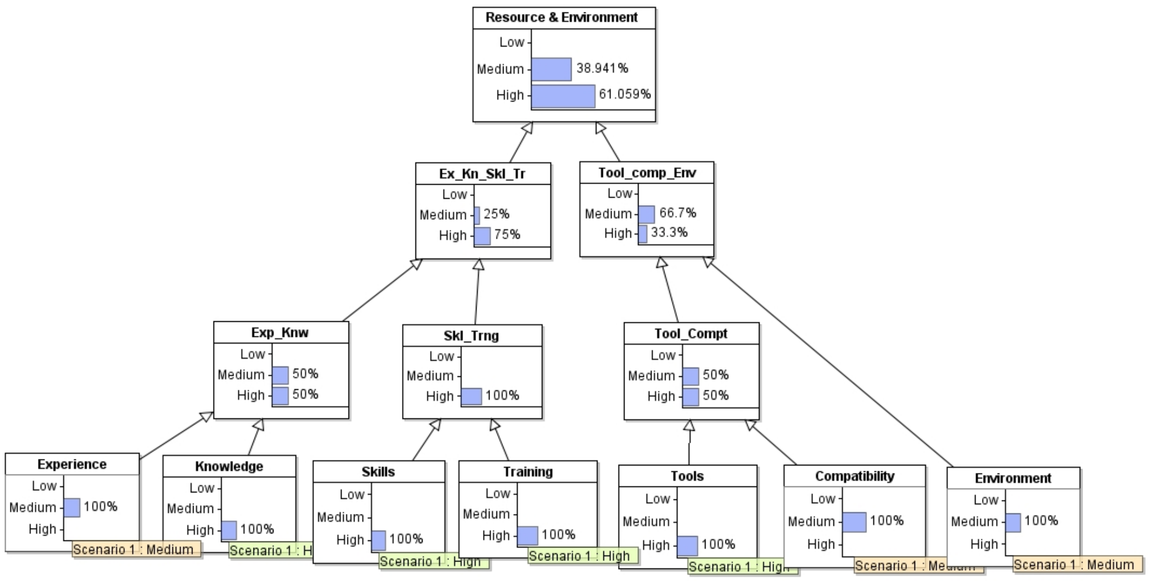

Fig. 2 Model for evaluation of software test resources and environment

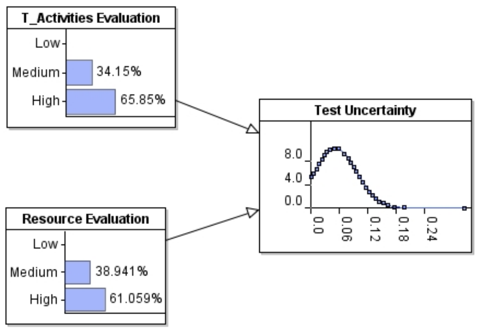

Fig. 3 Overall test uncertainty

4. Software Failure Probability Estimation

The probability of software failure after n^{th} testing has been estimated. Software test cases are prepared using specific input values. If the probability of a specific input is p, then the probability of software failure (φ) for the successful testing of the first test case can be estimated considering the test uncertainty (σ_1) as follows

$$\varphi_1 = 1-(1-\sigma_1)\, p_1 \ldots\ldots\ldots\ldots\ldots\ldots\ldots \text{(1)}$$

Similarly, the equation for a software failure probability after a series of successful testing can be derived as presented in Table 2.

Table 2 Software reliability after n^{th} test

n^{th} Test	Input Probability	Software failure probability after n^{th} test
0	-	1
1	p_1	$\varphi_1 =[1- p_1-\sigma_1 p_1]$ ……..(1)
2	p_2	$\varphi_2 = [1- p_1 - p_2 + \sigma_1 p_1 + \sigma_2 p_2]$…(2)
…….	……..	……………………..
k	p_k	$\varphi_i = [1- p_1 -p_2 - \ldots + \sigma_1 p_1 + \sigma_2 p_2 + \ldots + \sigma_k p_k]$……(3)
…….	…………	………………………
.	.	
n	p_n	$\varphi_n = [1- p_1 - p_2 - \ldots + \sigma_1 p_1 + \sigma_2 p_1 + \ldots + \sigma_n p_n]$……(4)
σ- test uncertainty, *φ - software failure probability after certain test, p - software input probability*		

After completion of the n^{th} test, the ultimate failure probability, φ_n can be expressed in the following form.

$$\varphi_n = 1 - \sum_{i=1}^{n} p_i + \sum_{i=1}^{n} \sigma_i p_i \ldots\ldots\ldots\ldots\ldots\ldots\ldots \text{(5)}$$

In Eq. (5), the second term in the right-hand side is the coverage of test cases that tends unity for a very large number of test cases. The third term expresses the accumulated test uncertainties, which seems to be non-zero because an ideal test environment is infeasible.

5. Discussions

The quality of software test processes and the test environment play vital roles in software failure probability quantification based on test results. Thus, the entire test process along with the test environment should be thoroughly evaluated and taken into account in software reliability quantification. This study proposes the BBN model for software failure probability estimation considering the test uncertainties. Hence, the following points important to discuss.

- A software test uncertainty estimation can be performed simply through expert judgment, using matrices, or statistical tools like a Bayesian belief network. However, the use of a BBN model is emphasized since the approach is capable of assessing the underlying causes establishing relations between the causes and effects.

- The BBN structure can be constructed using a converging connection or diverging connection. In our example, a converging connection has been chosen because the parent nodes are conditionally independent. The intermediate nodes (the nodes between the root nodes and leaf node) are used in the structure to make the node probability table (NPT) editing task easier. The NPT of the root nodes can be built through a qualitative assessment (e.g. High, Medium, and Low) by experts.

- The attributes have a different level of influence on the test uncertainty. Thus, different weighting factors should be considered for each of the attributes while preparing the node probability table of a BBN. BBN parameters and the structure of the network should be prepared by appropriately reflecting the influences of all test activities, test platform, and environment. The scale of the weighting factors should be determined by experts.

- The main challenge is determining the prior probability distribution of the BBN; however, expert elicitation along with historical test failure data related to safety and non-safety software can be used for a prior probability estimation.

- The approach for a software failure probability estimation presented in section 4 considers the test uncertainty and probability of inputs. If only a fraction of test cases are executed then a factor relevant to the test coverage should be included in the calculation.

6. Conclusions

A reliability estimation of software is very important for a probabilistic safety analysis of a digital safety critical system of NPPs. This study focused on the estimation of the probability of a software failure that considers the uncertainty in software testing. In our study, BBN has been employed as an example model for software test uncertainty quantification. Although it can be argued that the direct expert elicitation of test uncertainty is much simpler than BBN estimation, however the BBN approach provides more insights and a basis for uncertainty estimation. Our study is expected to provide an option for a reliability estimation of safety critical and non-safety software in the nuclear power industry.

Acknowledgement

This work was supported by Nuclear Research & Development Program of the National Research Foundation of Korea grant, funded by the Korean government, Ministry of Science, ICT & Future Planning (Grant Code: 2012M2A8A4025991).

REFERENCES

[1] H. G. Kang, H. G. Lim, H. J. Lee, M. C. Kim, S. C. Jang, "Input-profile-based software failure probability quantification for safety signal generation systems," Reliability Engineering and System Safety, 94, pp. 1542–1546, 2009.

[2] Musa JD, The operational profile in software reliability engineering: an overview. In: Third International Symposium on Software Reliability Engineering, October 1992.

[3] Cao, P., Tang, G., Zhang, Y., Luo, Z., 2014. Qualitative Evaluation of Software Reliability Considering Many Uncertainty Factors. Ecosystem Assessment and Fuzzy Systems Management, Advances in Intelligent Systems and Computing 254, DOI: 10.1007/978-3-319-03449-2_20, Springer International Publishing Switzerland 2014. Page 199-205.

[4] Li, N. Malaiya, Y.K., "On Input Profile selection for software testing," In: Proceedings of 5th International Symposium on Software Reliability Engineering (ISSRE' 94), IEEE computer society, Los Alamitos, CA, pp. 196-205, 1994.

[5] Rees, K., "Managing the uncertainties of software testing: a Bayesian approach," Proceedings 14th Advances in Reliability Technology Symposium, Manchester, November 2000.

[6] Elbaum, S., Rosenblum, D.S., "Known Unknowns: Testing in the Presence of Uncertainty," FSE'14 , Hong Kong, China, November 16–22, 2014.

[7] Ziv, H., Richardson, D.J., Khosch, R., The Uncertainty Principles in Software Engineering. Technical Report 96-33, University of California, Irvine, CA, USA, August 1996.

[8] Ziv, H., Richardson, D.J., 1997. Constructing Bayesian-network Models of Software Testing and Maintenance Uncertainties. The Proceedings of International Conference on Software Maintenance, September 1997, IEEE.

[9] A. L. Goel. "Software reliability models: Assumptions, limitations, and applicability," IEEE Transactions on Software Engineering, SE-11(12):1411-1423, 1985.

[10] Naik, K. and Tripathy, P., Software Testing and Quality Assurance, Theory and Practice. John Wiley & Sons, Inc. Hoboken, New Jersey, 2008.

[11] IEEE, 2012. IEEE Standard for System and Software Verification and Validation. IEEE Std. 1012-2012.

[12] Cem Kaner, J.D., "What Is a Good Test Case?" Florida Institute of Technology Department of Computer Sciences, May-2003.

[13] Singh, R., 2014. Test case generation for object-oriented systems: A review. The Proceedings of Fourth International Conference on Communication Systems and Network Technologies, 2014 IEEE. P 981-989, DOI 10.1109/CSNT.2014.201.

[14] Urbina, A. and Mahadevan, 2010. Quantification of Aleatoric and Epistemic Uncertainty in Computational Models of Complex Systems. Proceedings of the IMAC-XXVIII February 1–4, 2010, Jacksonville, Florida USA.

[15] BTP-7-14, 2007. Guidance on software review reviews for digital computer-based instrumentation and control systems. NUREG-0800, Standard Review Plan: Branch Technical Position 7-14, Revision 5. U.S. Nuclear Regulatory Commission, 2007

[16] Eickelmann, N.S. and Richardson, D.J., 1996. An Evaluation of Software Test Environment Architectures. The proceedings of ICSE-18, IEEE.

[17] Fenton N. E., Neil, M., Risk Assessment and Decision Analysis with Bayesian Network, 2012.

[18] H. Koziolek, "Operational Profiles for software reliability," Seminar 'Dependability Engineering", Carl von Ossietzky University of Oldenburg, Germany, July 2005.

[19] Fenton, N. E., Neil, M., Caballero, J.G., 2007, "Using Ranked Nodes to Model Qualitative Judgments in Bayesian Networks," IEEE Transactions on Knowledge and Data Engineering, Vol. 19, No. 10.