High level issues in reliability quantification of safety-critical software

KIM Man Cheol

Integrated Safety Assessment Division, Korea Atomic Energy Research Institute, Daejeon, Korea (charleskim@kaeri.re.kr)

Abstract: For the purpose of developing a consensus method for the reliability assessment of safety-critical digital instrumentation and control systems in nuclear power plants, several high level issues in reliability assessment of the safety-critical software based on Bayesian belief network modeling and statistical testing are discussed. Related to the Bayesian belief network modeling, the relation between the assessment approach and the sources of evidence, the relation between qualitative evidence and quantitative evidence, how to consider qualitative evidence, and the cause-consequence relation are discussed. Related to the statistical testing, the need of the consideration of context-specific software failure probabilities and the inability to perform a huge number of tests in the real world are discussed. The discussions in this paper are expected to provide a common basis for future discussions on the reliability assessment of safety-critical software.

Keyword: probabilistic safety assessment/probabilistic risk assessment; digital instrumentation and control; software reliability; Bayesian belief network; statistical testing

1 Introduction

The quantification of software reliability has been one of the areas that have received a lot of attention in the field of probabilistic safety assessment (PSA)/probabilistic risk assessment (PRA) of digital instrumentation and control (I&C) systems. In an effort to develop a technically sound method for software reliability quantification, a research on Bayesian belief network (BBN) modeling and statistical testing for quantifying software reliability is ongoing. The objective of the research is to obtain insights into the feasibility, practicality and usefulness of developing digital system models for inclusion in PSAs/PRAs of nuclear power plants (NPPs), specifically with respect to incorporating software failures into the models.

With the consideration that we do not have consensus methods for quantifying the reliability of safety-critical software in NPPs, a workshop involving experts with knowledge of software reliability and/or NPP PSA/PRA was held in May 2009. At the workshop, experts established a philosophical basis for modeling software failures in a reliability model[1]. Based on the philosophical basis, a review of quantitative software reliability methods (QSRMs) is performed[2], which is expected to contribute to the development of consensus methods for modeling and quantifying the failures of safety-critical software in NPPs.

From the review of QSRMs, the BBN modeling is identified as a promising method for accounting for the quality of software lifecycle activities, while the statistical testing is identified as a promising method for accounting for the quality of the final product, *i.e.* software. Because it is generally accepted that the consideration of both the quality of software lifecycle activities and the quality of final product is necessary to give more confidence on the estimation of the reliability of software, both the BBN modeling and the statistical testing are considered in this paper.

2 BBN modeling

2.1 Assessment approaches

Bayesian networks[3] are directed acyclic graphs in which the nodes represent propositions or variables, the arrows (or arcs) signify the existence of direct causal dependencies between the linked propositions, and the strengths of these dependencies are quantified by conditional probabilities. Recently, Bayesian networks have been applied to the reliability estimation of safety-critical software. Many BBN models for software reliability quantification such as Helminen[4], Gran[5], and Fenton *et al.*[6] have been developed, and each of the BBN models has different focus and emphasis, and therefore has its own

Received date: May 31, 2012
(Revised date: June 4, 2012)

advantages and disadvantages. To take advantage from different focus and emphasis of each BBN model, it seems to be helpful to consider the BBN modeling from the assessment approaches.

Neil *et al*.[7] mentioned three ways of carrying out systems assessment and their purposes, which are:

- In-process assessment
- Pre-deployment assessment
- In-field (retrospective) assessment

The selection of the assessment approach is closely related to the purpose of the assessment. In-process assessment is performed to identify and prevent problems earlier, pre-deployment assessment is performed to evaluate products and processes after the system has been produced, but before deployment, and in-field assessment is performed to assess a system that is already being operated.

If the purpose of the assessment is interpreted in terms of the regulatory process, it can be summarized as shown in Table 1. Considering that the risk-informed analysis process for digital systems, which is the safety analysis process for digital systems with the information on PSA model and results, has not yet been satisfactorily developed, it is preferable to develop a software reliability assessment method with the consideration of the risk-informed analysis process in mind.

Table 1 Application area of software reliability assessment in regulatory process

Assessment approach	Application in regulatory process
In-process	Regulatory review
Pre-deployment	Issuance of amendments
In-field	Risk-informed regulation

2.2 Consideration on evidence

When evidences are found during or after the software lifecycle activities, the evidences can be reflected to change the probability distribution of the BBN model based on Bayes's theorem. This process is called Bayesian update. One desirable feature that should be considered in the BBN model is the easiness in evidence collection. In this sense, it is also desirable to model observable quantities in the BBN model, so that evidences on the observable quantities can be reflected to the BBN model by Bayesian update.

Helminen[4] identified main sources of reliability evidence in the case of safety critical system as follows:

- Design features
- Development process
- Testing
- Operational experience

If we compare the three assessment approaches given by Neil *et al*.[7] and the four main sources of reliability evidence given by Helminen[4], the available sources of reliability evidence depending on the assessment approach can be summarized as shown in Table 2.

Table 2 Sources of evidence depending on assessment approach

Assessment approach	Sources of evidence
In-process	Development process
	Design features
Pre-deployment	Development process
	Design features
	Testing
Retrospective (in-field)	Development process
	Design features
	Testing
	Operational Experience

The four main sources of reliability evidence in the case of safety critical system can be divided into the following two categories:

- Qualitative evidence
- Quantitative evidence

The design features and the development process of the system are considered to be the sources of qualitative evidence, while the testing and the operational experience are expected to provide directly measurable statistical evidence and thus are considered to be the sources of quantitative evidence[4]. A similar distinction can also be found in Gran[5], where the BBN model can be divided into the quality part which reflects the qualitative evidence and the testing part which reflects one of the quantitative evidence, the testing result. Neil *et al*.[7] also mentioned five sources of evidence, which are development process evidence, product evidence, resource evidence, evidence about the operating environment, and analogy. The five sources of evidence are considered to be more related to the qualitative evidence.

The use of quantitative evidence from testing and operational experience is considered to be relatively easy, while the use of qualitative evidence from design features and development process is considered to be relatively difficult, because it requires extensive use of expert judgment. One of the observable quantitative evidence from the development process which also have significant impact on the software reliability is the number of defects, which can be estimated by the number of anomaly reports.

The qualitative evidence from the design features and the development process follows certain quality assurance and quality control principles, which are based on applicable standards[4]. Even though it is true that there is very little empirical evidence to confirm the link between the process quality and the product quality, as mentioned by Fenton *et al*.[8, 9], it is generally believed that the more strict standards the design features and development process fulfill the more reliable the system is believed to be.

2.3 Cause-consequence relation

After reviewing various BBN models for software reliability quantification, it is concluded that the development of a BBN model based on a specific guideline for evaluating safety-critical software products is most promising. An example of such specific guideline is the branch technical position (BTP) 7-14 (guidance on software reviews for digital computer-based instrumentation and control systems) in NUREG-0800 (standard review plan).

The acceptance criteria for design outputs in BTP 7-14 are divided into two sets: functional characteristics and process characteristics. The functional characteristics are composed of seven individual characteristics, which are accuracy, functionality, reliability, robustness, safety, security, and timing. The process characteristics are also composed of seven individual characteristics, which are completeness, consistency, correctness, style, traceability, unambiguity, and verifiability.

Another identified issue related to the BBN modeling is the cause-consequence relation of a set of characteristics and an individual characteristic. In one viewpoint, a set of characteristics is viewed as the result of its individual characteristics, and therefore each of the individual characteristics is viewed as the cause and the set of characteristics is viewed as the consequence. This viewpoint is illustrated in Fig. 1, with the example of functional characteristics. In Fig. 1, the seven characteristics nodes such as accuracy, functionality, timing and so on are considered as independent nodes and affect the functional characteristics node. In this case, the seven characteristics nodes are considered as the causes and the functional characteristics node is considered as the consequence. The advantage of this viewpoint is its intuitiveness, because the concept behind the structure of the BBN model is easier to be understood and accepted by most practitioners in the field. But, it should be noted that several limitations associated with this viewpoint such as independence among the seven individual characteristics nodes also exist. In other words, several disadvantages were also found in actual application of this viewpoint to software reliability quantification.

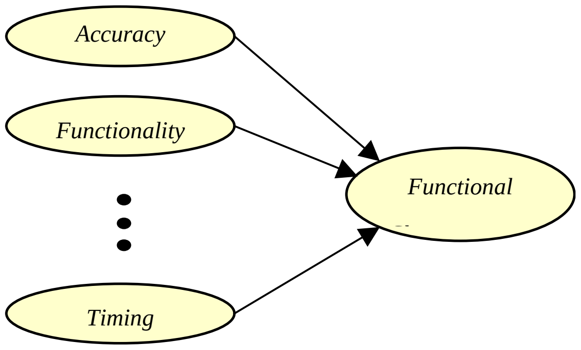


Fig. 1 Viewpoint of each characteristic as cause.

In another viewpoint, each characteristic can be viewed as the result of an overall function or process, and therefore a set of characteristics is viewed as the cause and each of its individual characteristics is viewed as the consequence. This viewpoint is illustrated in Fig. 2, with the example of functional characteristics. In Fig. 2, the functional characteristics node is considered as an independent node and affects the seven characteristics nodes such as accuracy, functionality, timing and so on. In this case, the functional characteristics node is considered as the cause and the seven characteristics nodes are considered as the consequences. The advantages of this viewpoint such as the existence of interdependence among the seven individual characteristics nodes were found in actual application of this viewpoint to software reliability quantification. But, it should be admitted that this viewpoint is less intuitive and more difficult to be understood and accepted by the practitioners in the field.

As mentioned above, each viewpoint has its advantages and disadvantages, and therefore proper selection of the viewpoint is necessary in the development of the BBN model for safety-critical software reliability quantification.

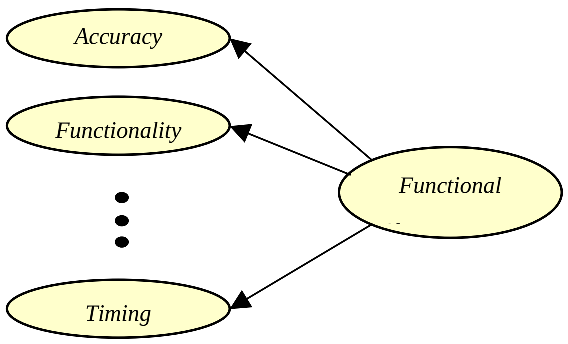


Fig. 2 Viewpoint of each characteristic as consequence.

3 Statistical testing

The idea of statistical testing to demonstrate the reliability of software seems to begin by Currit *et al*.[10] and Musa *et al*.[11]. As explained by Chillarege[12], the central idea of statistical testing is to use software testing as a means to assess the reliability of software, contrary to the popular use of software testing as a debugging method. Therefore, instead of preparing the test cases based on specifications, requirements, and testers' expectation on what an implementation is most likely to do wrong[13], test cases should be prepared based on the operational profile of the software.

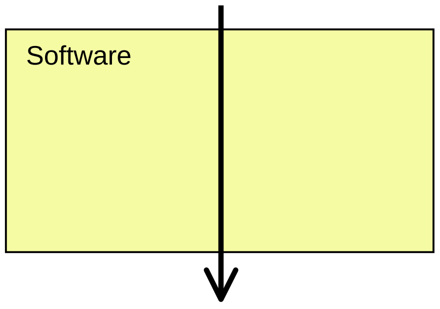
The recognition of the importance of the operational profile leads to the discussion on how to determine the operation profile, which will be the basis for preparing test cases for the statistical testing. If the operation profile is determined as an average of the integration of all possible contexts that the software will be subjected to, the statistical testing will provide an average reliability of the software over all possible contexts. If an operational profile is determined to each of all possible contexts, the statistical testing will provide a context-specific reliability of the software. Because software failures are, in general, context-specific, the average reliability of the software over all possible contexts or initiating events does not have a lot of meaning when the software is subject to a specific context.

In NPPs, a specific initiating event is likely to cause one or more plant parameters to exceed certain conditions, and thus cause safety-critical digital I&C systems such as reactor protection systems (RPSs) or engineered safety features actuation systems (ESFASs) to generate actuation signals. Because different initiating events are expected to cause different sets of plant parameters to exceed certain conditions, the software in the safety-critical digital I&C systems is required to provide correct output in different conditions depending on different initiating events. In other words, a specific initiating event forms a specific context to the software in the safety-critical digital I&C systems.

Considering that PSA/PRA develops event trees and fault trees to each of the initiating events, strictly speaking, software failures following an initiating event should be considered in a context-specific manner, and therefore a context-specific software failure probability should be provided to the context-specific software failure.

But, in reality, it is already widely known that the demonstration of a reasonable average software failure probability requires a statistical testing with a significant number of test cases without failures. For example, Chu *et al*.[2] mentioned that more than 10^5 tests with no failures must be conducted to demonstrate a mean software failure probability of 10^{-5}, which is a normally expected software failure probability for safety-critical digital I&C systems such as an RPS or ESFAS.

Figure 3 conceptually shows this type of testing for averaged software reliability. It should be noted that this type of software testing for averaged software reliability corresponds to the black-box-based statistical software testing for the software reliability demonstration. In this software testing, the software is considered as a black box and therefore it is simply assumed that there exists only one single path from the input to the output, instead of considering the detailed arrangement of the internal software modules and the structure of the paths among the modules. Test cases are prepared based on the operational profile of the software. The basic logic behind this type of software testing is that the single path in the software is considered to represent the average of various paths in the software and the test cases are considered to represent the average context. In this sense, successful execution of 10^5 test cases representing the averaged context on the single path representing the averaged software paths is believed to demonstrate a certain level of software reliability, say 10^{-5}.

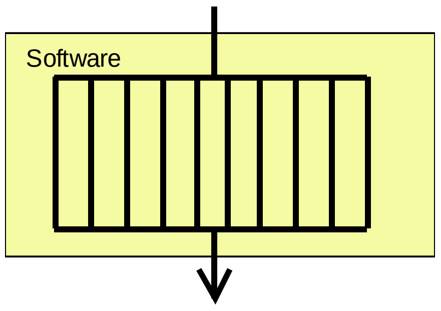


10^5 tests for averaged context

Fig. 3 Testing for averaged software reliability.

Figure 4 conceptually shows the software testing for context-specific software reliability. It should be noted that this type of software testing for context-specific software reliability corresponds to the white-box-based statistical software testing for the software reliability demonstration. In this software testing, the detailed arrangement of the internal software modules and the structure of the paths among the modules are all considered. In the example shown in Fig. 4, it is assumed that ten different paths exist depending on the ten different contexts that the software will be subjected to.

One of the difficulties in this type of the white-box-based statistical software testing is that the software reliability in each path has to be demonstrated. In the example shown in Fig. 4, if it is assumed that the occurrence probabilities of all ten contexts are same as 0.1 and it is required to demonstrate the software failure probability is less than a certain value, say 10^{-5}, the calculation shows that the software failure probability of a path corresponding to a specific context should be demonstrated to the same value, 10^{-5}. If successful execution of 10^5 test cases is required to demonstrate such software failure probability, the total number of successful execution of test cases for demonstrating the software failure probability of 10^{-5} becomes 10^6, because there exist ten contexts and the successful execution of 10^5 test cases is required to each context. In summary, considering that each of the software failure probability specific to an initiating event has to be demonstrated, there is a concern whether a statistical testing with such a large number of test cases is possible in reality or not.



10^5tests/context $\times$ 10 contexts = 10^6 tests

Fig. 4 Testing for context-specific software reliability.

In fact, even if it is assumed that an averaged software failure probability for all possible initiating events can be used to a specific initiating event in PSA/PRA, the demonstration of such a low averaged software failure probability is already not easy.
Between the need of the consideration of context-specific software failure probabilities and the inability to perform a huge number of tests in the real world, it is necessary to find an appropriate way of compromising the two sides.

The above example also raise another issue related to the statistical testing on whether the knowledge on the internal structure of the software is helpful in reducing the number of necessary test cases for demonstrating a predefined software reliability level or not. In one viewpoint, ironically, the knowledge on the internal structure of the software seems to increase the number of necessary test cases. More discussions are necessary to find a clear answer to this issue.

4 Conclusions

For the purpose of developing a consensus method for the reliability assessment of safety-critical digital I&C systems NPPs, several high level issues in reliability assessment of the safety-critical software are discussed. Considering that both the quality of the development process and the quality of the software product are important for safety-critical software, BBN modeling is considered as a method for accounting for the quality of the development process and the statistical testing is considered as a method for accounting for the quality of the software product. The high level issues discussed related to the BBN modeling are:

- Relation between the assessment approach and the sources of evidence
- Relation between qualitative evidence and quantitative evidence
- How to consider qualitative evidence.
- Cause-consequence relation

The high level issues discussed related to the statistical testing are:

- Need of the consideration of context-specific software failure probabilities
- Inability to perform a huge number of tests in the real world
- Usefulness of the knowledge on the internal structure of the software in reducing the number of required software tests

It seems that a long way is ahead to reach a consensus method for software reliability quantification. In this sense, the contributions from a lot of experts with various backgrounds are necessary to develop a consensus method for the reliability quantification of safety-critical software. The high level issues and related discussions introduced in this paper are expected to provide a common basis for future discussions among the experts in the field of the reliability assessment of safety-critical software.

Nomenclature

BBN	Bayesian Belief Network
ESFAS	Engineered Safety Features Actuation Systems
I&C	Instrumentation and Control
NPP	Nuclear Power Plant
PSA	Probability Safety Assessment
PRA	Probability Risk Assessment
QSRM	Quantitative Software Reliability Method
RPS	Reactor Protection Systems

Acknowledgement

This work was partly supported by Nuclear Research & Development Program of the National Research Foundation of Korea (NRF) grant funded by the Korean government (MEST) (grant code: 2012-011506).

References

[1] CHU, T.L., MARTINEZ-GURIDI, G., YUE, M., SAMANTA, P., VINOD, G., and LEHNER, J.: Workshop on philosophical basis for incorporating software failures in a probabilistic risk assessment, BNL-90571-2009-IR, Upton, NY: Brookhaven National Laboratory, 2009.

[2] CHU, T.L., YUE, M., MARTINEZ-GURIDI, G., and LEHNER, J.: Review of quantitative software reliability methods, BNL-94047-2010, Upton, NY: Brookhaven National Laboratory, 2010.

[3] PEARL, J.: Probabilistic reasoning in intelligent systems: networks of plausible inference, San Mateo, CA: Morgan Kaufmann Publishers, 1988.

[4] HELMINEN, A.: Reliability estimation of safety-critical software-based systems using Bayesian networks, STUK-YTO-TR 178, Helsinki: STUK, 2001.

[5] GRAN, B. A.: SCIENCE AND SUBSTANCE: ASSESSMENT OF PROGRAMMABLE SYSTEMS USING BAYESIAN BELIEF NETS, SAFETY SCIENCE, 2002, 40: 797-812.

[6] FENTON, N., NEIL, M., MARSH, W., HEARTY, P., MARQUEZ, D., KRAUSE, P., and MISHRA, R.: PREDICTING SOFTWARE DEFECTS IN VARYING DEVELOPMENT LIFECYCLES USING BAYESIAN NETS, INFORMATION AND SOFTWARE TECHNOLOGY, 2007, 49:32-43.

[7] NEIL, M., LITTLEWOOD, B., and FENTON, N.: Applying Bayesian belief networks to system dependability assessment In: Proceedings of safety critical systems club symposium, 1996, 71-94.

[8] FENTON, N. E., PFLEEGER, S. L., and GLASS, R. L.: SCIENCE AND SUBSTANCE: A CHALLENGE TO SOFTWARE ENGINEERS, IEEE SOFTWARE, 1994, 11(4): 86-95.

[9] FENTON, N. E., and NEIL, M.: A CRITIQUE OF SOFTWARE DEFECT PREDICTION MODELS, IEEE TRANS SOFTWARE ENG., 1999, 25(5): 675-689.

[10] CURRIT, P. A., DYER, M., and MILLS, H. D.: CERTIFYING THE RELIABILITY OF SOFTWARE,

IEEE TRANS SOFTWARE ENG., 1986, SE-12 (1): 3-11.

[11] MUSA, J. D., IANNINO, A., and OKUMOTO, K.: Software reliability: measurement prediction application, New York: McGraw-Hill, 1987.

[12] CHILLAREGE, R.: Software testing best practices, IBM Research Technical Report RC 21457, IBM Research, 1999.

[13] BANKS, D., DASHIELL, W., GALLAGHER, L., HAGWOOD, C., KACKER, R., and ROSENTHAL, L.: Software Testing by Statistical Methods - Preliminary Success Estimates for Approaches based on Binomial Models, Coverage Designs, Mutation Testing, and Usage Models, Gaithersburg, MD: National Institute of Standards and Technology, 1998.