APPLICATION OF A BAYESIAN BELIEF NETWORK MODEL TO RELIABILITY ASSESSMENT OF NUCLEAR SAFETY-RELATED SOFTWARE

Seung Jun Lee
School of Mechanical and Nuclear Engineering
Ulsan National Institute of Science and Technology
50, UNIST-gil, Ulsan, Republic of Korea
sjlee420@unist.ac.kr

Sang Hun Lee, and Hyun Gook Kang
Department of Mechanical, Aerospace, and Nuclear Engineering
Rensselaer Polytechnic Institute
110 8th St, Troy, NY, USA
kangh6@rpi.edu; lees35@rpi.edu

Tsong-Lun Chu, Athi Varuttamaseni, and Meng Yue
Brookhaven National Laboratory
Brookhaven Avenue, Upton, NY, USA
chu@bnl.gov; avarutta@bnl.gov; yuemeng@bnl.gov

Heung Seop Eom, and Jaehyun Cho
Korea Atomic Energy Research Institute
111, Daedeok-daero, Daejeon, Republic of Korea
ehs@kaeri.re.kr; chojh@kaeri.re.kr

Ming Li
U.S. Nuclear Regulatory Commission
Washington, DC, USA
Ming.Li@nrc.gov

ABSTRACT

As the instrumentation and control (I&C) systems in nuclear power plants (NPPs) have been replaced with digital-based systems, the need to incorporate software failures into NPP probabilistic risk assessments has arisen. In order to assess the probability of software failure on demand, a Bayesian belief network (BBN) model was developed which estimates the number of defects and the resulting probability of software failure on demand in nuclear safety-related software. To assess the feasibility of the BBN framework, the BBN model was applied to the prototype Integrated Digital Protection System-Reactor Protection System (IDiPS-RPS) to estimate the number of remaining faults and the software failure probability of a target software. The developmental- and V&V-activities carried out during the IDiPS-RPS development process were evaluated based on the well-defined checklist derived by the V&V team and were estimated based on expert elicitation. In addition, the attribute evaluations and the number of FPs of the target software is provided as the inputs for the BBN model. The application results showed the feasibility of using BBNs for quantifying software failure probabilities and several insights were gained from the applications of the BBN model. The proposed BBN framework can be applied to estimate the software failure probability for other safety-related NPP software and provide an insight on modeling the software development process that involves iterations between different development phases.

Key Words: Nuclear Power Plant; Probabilistic Risk Assessment; Software Reliability; Bayesian Belief Network

1 INTRODUCTION

The analog systems in nuclear power plants (NPPs) are approaching obsolescence; thus, to attain the functional advantages of digital systems, existing NPPs have begun to replace current analog instrumentation and control (I&C) systems with digital-based ones, with new plant designs fully incorporating digital systems. This shift in technology necessitates the need to develop and integrate digital I&C risk models into NPP probabilistic risk assessment (PRA) models. Previous research has explored the possibility of addressing failure in digital I&C systems within the framework of current NPP PRAs, including reliability modeling of software failure in a digital I&C system [1, 2].While there is no consensus on the definition of software failure [3, 4], software failure in safety graded systems in digitalized NPPs can be defined as the triggering of a software fault that results in, or contributes to, the host digital system failing to accomplish its intended function, or initiating an undesired action. Therefore, the reliability of the software must be quantified to guarantee the safety of a digitalized NPP, since software failure can significantly affect the digital safety systems [5].

In order to estimate the probability of software failure on demand and incorporate it into an NPP PRA model, a Bayesian belief network (BBN) model was developed in the authors' previous study that estimates the number of defects in software programs considering the software development life cycle (SDLC) characteristics. Especially, the BBN model demonstrates the framework for (1) identifying software development characteristics; (2) establishing and quantifying the causal relationships between these characteristics, estimating the number of defects remaining; (3) probabilistically aggregating multiple expert inputs in order to quantify the BBN nodes and further estimate the software failure probability.

To assess the feasibility of the BBN framework, the BBN model was applied to an example system, namely the Integrated Digital Protection System–Reactor Protection System (IDiPS-RPS) developed by the Korea Nuclear Instrumentation and Control System (KNICS) project. When applied to a target software program, the quality of software development and verification and validation (V&V) activities is evaluated against the attributes, and software-specific data on the number of faults detected and the software size are estimated and used to Bayesian-update the BBN model to make it specific to the program being analyzed.

2 TARGET SYSTEM

2.1 Configuration of IDiPS-RPS

The IDiPS-RPS is a digitalized RPS developed through KNICS project for newly constructed NPPs as well as for upgrading existing analog-based RPSs. It has the same function as an analog-based RPS to automatically generate a reactor-trip signal and engineered safety-features actuation signals whenever process variables reach their corresponding predefined trip set-points. As shown in Fig. 1, the IDiPS-RPS consists of four redundant channels located in rooms that are electrically and physically isolated, with each channel composed of four main processors: the bistable processor (BP), the coincidence processor (CP), the automatic test and interface processor (ATIP), and the cabinet operator module (COM) [6, 7].

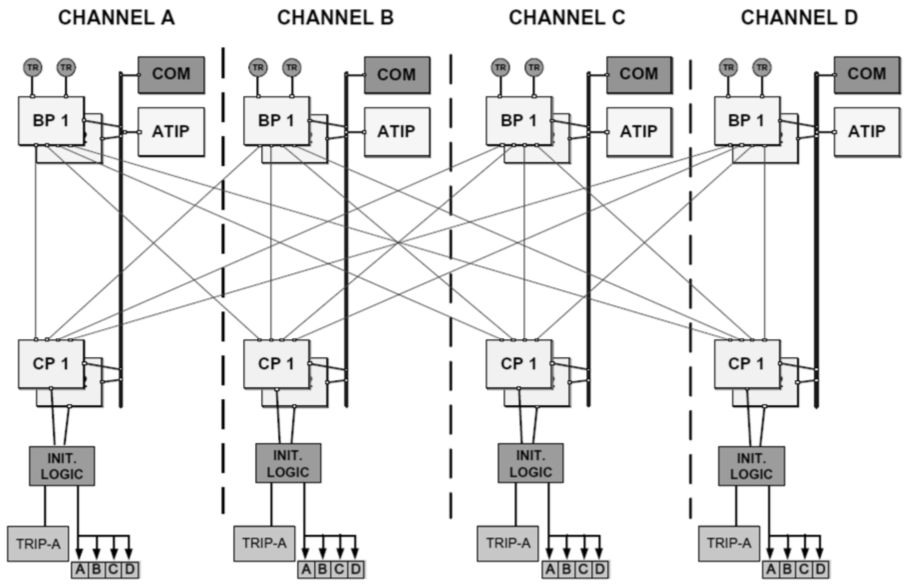

Figure 1. Overall configuration of IDiPS-RPS

2.2 Software Development and V&V Activities of IDiPS-RPS

IDiPS-RPS software implements the trip functions of the BP, the voting logic of the CP, and the test functions of the ATIP. Thus, any malfunction of the software in the BP or CP may result in irreversible consequences. As most of the software in an RPS is classified into a safety–critical or safety-related class, the software used in the IDiPS-RPS was developed under a rigorous procedure with independent V&V; Fig. 2 shows the SDLC phases and the V&V activities performed during the development of the IDiPS-RPS software. The main V&V activities in the Requirement and Design phases are the licensing suitability evaluation, detailed inspection, and a traceability analysis.

Detailed inspection of the software requirement specification (SRS) and the software design specification (SDS) was conducted by the Fagan inspection method [8], which focuses on the functional behavior of the software system. It evaluates development documents for their correctness, completeness, and consistency. Each viewpoint is divided into four sub-viewpoints: functional definition, input- and output-definitions, behavior specification, and interfaces. Analyses for evaluating licensing suitability together with the sub-viewpoints in the detailed inspection were undertaken based on the previously derived checklists. They were refined carefully with the aid of many software V&V experts based on various standards, guidelines, and some related reports, as well as their V&V experience.

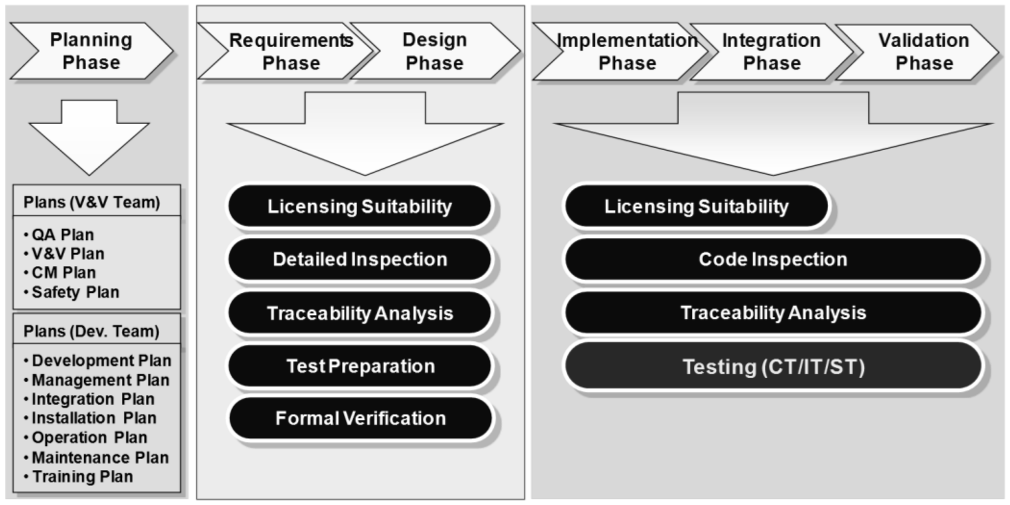

Figure 2. Software V&V activities for BP software

3 APPLICATION OF THE BBN MODEL TO IDIPS-RPS

For the specific software application, the inputs for the BBN model requires attribute evaluations and the number of function points (FPs) of the target software. In this study, experts who are familiar with IDiPS-RPS developments evaluated the attributes (e.g., three ordinal levels "low", "medium" and "high") carried out during the development process of each software. The attribute scores were then used to obtain a probability distribution (i.e., a probability indicating the likelihood of a node being in each state) for the quality and V&V nodes in the model. It is notable that the development experience of the expert was related to the prototype of the IDiPS-RPS rather than its final version that is planned to be adopted in digitalized NPPs after thorough revision.

The diverse opinions provided by multiple experts were treated in an integrated manner to estimate NPTs for each specific software development process by using the distribution of the experts' opinions. The number of FPs of each software was estimated based on the lines of code (LOC) evaluated by experts. The BBN models were then executed using WinBUGS [9] to obtain the number of defects introduced and remaining at each phase, and the detection probability of the defects introduced in the current and passed from previous phase for a target software program.

The evaluation results for the attribute qualities of the prototype IDiPS-RPS are shown in Table I. The attributes in Installation-and-Checkout phase are assumed to have *Medium* quality since there is no data for the Installation-and-Checkout phase as the IDiPS-RPS has yet been installed.

Table I. Attribute evaluation results of IDiPS-RPS

Phase	High Attributes	Medium Attributes	Low Attributes
Requirement Development	0	9	3
Requirement V&V	0	7	8
Design Development	0	12	3
Design V&V	0	6	10
Implementation Development	0	11	5
Implementation V&V	0	4	14
Test Development	0	7	3
Test V&V	0	0	11
Installation/Checkout Development	NA	NA	NA
Installation/Checkout V&V	NA	NA	NA

For the specific software application, the number of FPs of the target software as well as the software attribute evaluations must be provided as the inputs for the BBN model. The BP software of IDiPS-RPS was developed using a function block diagram and converted to C source code by the compiler. The counting rules for the source code are based on logical statements rather than on physical lines of code; a more accurate conversion is obtained when logical statements are counted rather than physical lines [10].

The total LOC of the C source program for the BP software is 18,652, and the LOC related to trip functions is 13,047. For a more accurate conversion, the logical statements of the BP C code were counted and the LOC of logical statements for trip functions was identified as 7,186. Since the source statements per function point of C language is 60, 128, and 170 for low, mean, and high, respectively, the estimated FPs of the BP code is in the range of 42 to 120 (Low: 120 FPs, Mean: 56 FPs, and High: 42 FPs).

Tables II to V show the evaluation results of the IDiPS-RPS BP software. In this evaluation, it was assumed that the BP software has the mean source statements per FP. Table II shows the number of defects introduced in each phase of developing the IDiPS-RPS. Table III shows the estimated number of defects remaining at the end of each phase. Tables IV and V show the detection probabilities for defects introduced in the current and previous phases, respectively.

Table II. Number of defects introduced in each phase for IDiPS-RPS

	Mean	σ	5th	50th	95th
Defects introduced in Requirement Phase	21.78	39.85	0	0	112
Defects introduced in Design Phase	47.52	58.94	0	56	168
Defects introduced in Implementation Phase	55.28	63.76	0	56	168
Defects introduced in Test Phase	22.55	39.76	0	0	112
Defects introduced in Installation/Checkout Phase	14.37	33.23	0	0	56

Table III. Number of defects remaining at the end of each phase for IDiPS-RPS

	Mean	σ	5th	50th	95th
Defects remaining at the end of Requirement Phase	8.67	19.50	0.00	0.00	47.00
Defects remaining at the end of Design Phase	27.40	34.45	0.00	17.00	96.00
Defects remaining at the end of Implementation Phase	41.31	41.59	0.00	31.00	123.00
Defects remaining at the end of Test Phase	23.58	25.09	0.00	16.00	73.00
Defects remaining at the end of Installation/Checkout Phase	9.89	13.19	0.00	5.00	36.00

Table IV. Defect-detection probability for defects introduced in the current phase of IDiPS-RPS

Phase	Mean	σ	5th	50th	95th
Requirement	0.60	0.25	0.17	0.63	0.95
Design	0.57	0.22	0.18	0.58	0.91
Implementation	0.60	0.25	0.17	0.63	0.95
Test	0.64	0.16	0.35	0.65	0.88
Installation/Checkout	0.80	0.14	0.54	0.83	0.97

Table V. Defect detection probability for defects introduced in previous phases of IDiPS-RPS

Phase	Mean	σ	5th	50th	95th
Requirement	0.22	0.13	0.04	0.20	0.48
Design	0.29	0.16	0.07	0.27	0.59
Implementation	0.63	0.18	0.31	0.64	0.90
Test	0.70	0.19	0.33	0.73	0.96
Installation/Checkout	Mean	σ	5th	50th	95th

Tables VI show the estimated number of defects remaining in each phase, respectively, for three different levels of source statements per FP. Since the ratio of source statements per function point determines the number of FPs, the final remaining defects of low source statements per FP is about four times greater than that of high source statements per FP. If the exact number of FPs can be identified by the source code analysis, this uncertainty might be eliminated.

Table VI. Number of defects remaining at the end of each phase

Phase	Source statement per function point					
	Low (FP=120)		Medium (FP=56)		High (FP=42)	
	Mean	σ	Mean	σ	Mean	σ
Requirement	20.89	44.91	8.584	19.59	6.574	14.82
Design	67.2	82.36	27.27	34.37	20.76	26.15
Implementation	103	101.4	41.3	41.45	31.25	31.41
Test	59.3	58.96	23.63	25.04	17.77	18.88
Installation/Checkout	30.33	37.33	9.842	12.96	7.359	9.892

The number of faults remaining at the end of the Installation-and-Checkout phase given in Table III was used with the generic fault size distribution (FSD) estimated from the operating experience of safety-related protection system software to obtain a distribution for the probability of software failure on-demand for the IDiPS-RPS, as shown in Table VII. It is notable that due to the proprietary nature of the example software and its development process, the number of defects remaining and failure probabilities reported in this study are based on limited information for demonstration purpose only.

Table VII. Probability of IDiPS-RPS software failure on-demand

Mean	σ	5th	50th	95th
9.51E-04	1.10E-02	0.00E+00	1.89E-05	2.39E-03

4 CONCLUSIONS

In order to assess the probability of software failure on demand, a BBN model was developed in authors' previous study which estimates the number of defects and the resulting probability of software failure on demand in nuclear safety-related software. The model captures nuclear safety-related SDLC activity quality indicators and product information, and establishes the quantitative causal relationships between these indicators and the number of remaining defects. In this study, to assess the feasibility of the BBN framework, the BBN model was applied to the prototype IDiPS-RPS and the number of remaining faults and failure-probability of the target software was estimated. The SDLC phases and software developmental- and V&V activities (attributes) of each software were examined based on specifications and guidance of the target software.

In this study, experts who are familiar with IDiPS-RPS developments evaluated the attributes that are carried out during the software development process and the number of FPs of the software was estimated based on the LOC evaluated by the experts. The BBN models were then executed to obtain the number of defects introduced and remaining at each phase, and the detection probability of the defects introduced in the current and passed from previous phase for the target software. The number of final remaining defects in the software is further converted into a software failure probability based on FSD method.

Several insights were gained from the applications of the BBN model to a target nuclear safety graded software, and the application results showed the feasibility of using BBNs for quantifying software failure probabilities. The number of defects in each SDLC phase of a target safety software can be also used to gain quality insights on the software development process of each software. The proposed framework may be applied to estimate the failure probability and the number of residual defects for other NPP safety-related software, and can provide an insight on modeling the actual process of software development that involves iterations between different development phases.

5 ACKNOWLEDGMENTS

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: 2016M2B2A9909815, 2015M2A8A4021648).

6 DISCLAIMER

7 REFERENCES

1. Chu, T. L., et al., *Traditional probabilistic risk assessment methods for digital systems.*, US Nuclear Regulatory Commission, Washington, DC (2008).
2. Chu, T. L., et al., *Modeling a Digital Feedwater Control System Using Traditional Probabilistic Risk Assessment Methods.*, US Nuclear Regulatory Commission. Washington, DC (2009).
3. IEEE, *Systems and software engineering Vocabulary - First edition*, IEEE Std 61012, ISO/IEC/IEEE 24765 (2010).
4. Lyu, M.R., *Handbook of Software Reliability Engineering*, McGraw-Hill, 1996.
5. Kang, H. G., and T. Y. Sung. "An analysis of safety-critical digital systems for risk-informed design." *Reliability Engineering & System Safety*, **78.3**, pp. 307-314 (2002).
6. Eom, H. S., et al., "V&V-based remaining fault estimation model for safety–critical software of a nuclear power plant," *Annals of Nuclear Energy*, **51**, pp. 38-49 (2013).
7. U.S. NRC, *Guidance on Software Reviews for Digital Computer-Based Instrumentation and Control Systems*, NUREG-800, Standard Review Plan, Branch Technical Position 7-14, Revision 5 (2007).
8. Fagan, Michael E., "Design and code inspections to reduce errors in program development," *IBM Systems Journal*, **38.2/3,** pp. 258 (1999).
9. Spiegelhalter, David, et al. "WinBUGS user manual." (2003).
10. Jones, Capers., *Applied software measurement: global analysis of productivity and quality.*, McGraw-Hill Education Group (2008).