Transactions of the Korean Nuclear Society Spring Meeting
Gyeongju, Korea, May 29-30, 2008

General Method of Using Bayesian Nets for a Software Reliability Assessment in Varying SW Development Lifecycle

Heung-Seop Eom [a*], [a] Seung-Cheol Chang
[a]Korea Atomic Energy Research Institute, ISA Div., P.O.Box 105, Yuseong Daejeon, ehs@kaeri.re.kr
[*]Corresponding author: ehs@kaeri.re.kr

1. Introduction

Bayesian Net (BN) has been used in many researches to predict software defects, because it allows all the evidence to be taken into account [1,2]. However one of the serious difficulties in the earlier works was that the user had to build a different BN for each software development lifecycle. This limits the practical use of BN in the field. One way to solve this problem is the use of general BN templates which are not restricted to a particular software lifecycle. This paper describes a method for this purpose on the strength of Object-Oriented BN (OOBN) and Dynamic BN (DBN) technique.

2. Building the BN Model in Varying Software Development Lifecycle

The structure of a software lifecycle (SLC) basically consists of requirement phase, design phase, coding phase, and testing phase. Most SLCs which are available for the development of safety-critical software are a form of any number and combination of these 4 phases. Therefore the general BN model of a software development lifecycle can be constructed by ① creation of general phase-BN models and ② linking separate phase-BN models into a model for an entire lifecycle.

2.1 Overview of BN Modeling

A Bayesian Net [3] is a graph together with an associated set of probability tables. The BN graph consists of nodes and arcs. The nodes represent uncertain variables and the arcs represent the causal relationships between the variables. The BN has been known as one of promising technique which can predict the reliability of safety-critical software.

The creation of a large and repetitive BN is straightforward but very time consuming and laborious work. Object-Oriented BN (OOBN) [4] simplifies this work by creating predefined subnets which is called a 'Class' in Object-Oriented methodology terms. The BN of a SLC can be created by linking these phase subnets. There are general classes which are common in each development phase, so we can convert these classes into general BN subnets. The SLC is a set of phases and it defines the overall project. This can be expressed by using a dynamic BN (DBN) [5]. Figure 1 shows the BN modeling schema by using OOBN and DBN.

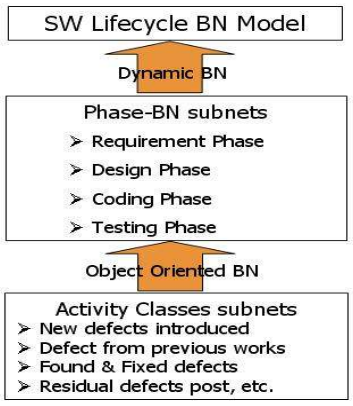

Fig. 1. Schema of BN modeling by using OOBN and DBN

2.2 Phase BN Model

A phase BN is a BN subnet which models a single software development phase. The phase BN consists of five classes.

▪ New defects introduced: This class models the quantity of the introduced defects during the works of the current phase.

▪ Residual defects from previous works: This class indicates the number of defects in the product from the previous phase.

▪ Defects found & fixed: This class models the number of defects which were found and fixed during the current phase.

▪ Residual defects post: This class models the number of defects that exist in the final product of the current phase.

▪ Phase frame: This class is a frame which deals with changes of defect's quantity in the current phase.

Figure 2 is an example of the class (BN subnet) 'Defects found & fixed.'

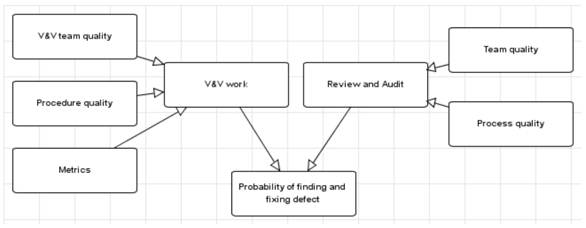

Fig. 2. BN subnet for the class "Defects found & fixed"

All the activities of the software development phases are concerned with the fault avoidance, fault detection & fix, and fault tolerance. This concept is reflected in the phase BN, and Figure 3 shows a generalized phase BN subnet.

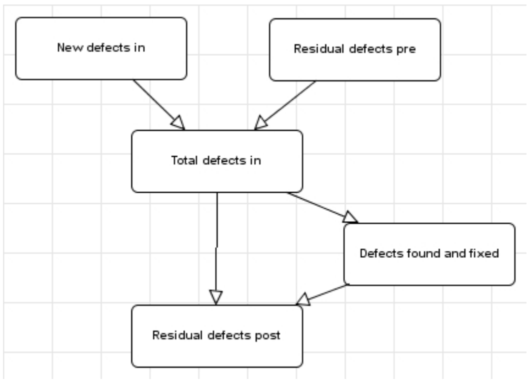

Fig. 3. Generalized phase-BN subnet

The classes in a phase BN can be applied (instantiated) to each development phase of a SLC as table 1.

Table 1: An example of instantiation of activity classes to each development phase

BN nodes / Dev. Phase	New defects introduced	Residual defects from previous works
Requirement Phase	Defects introduced in requirement specification	Defects exist is system design specification
Design Phase	Defects introduced in design specification	Defects exist in requirement specification
Coding Phase	Defects introduced in coding works	Defects exist in design specifications
Testing Phase	Defects introduced in testing works	Defects exist in code

BN nodes / Dev. Phase	Defects found & fixed	Residual defects post
Requirement Phase	V&V/ Review & Audit activities in requirement phase	Defects exist in requirement specification
Design Phase	V&V/ Review & Audit activities in design phase	Defects exist in design specification
Coding Phase	V&V/ Review & Audit activities in coding phase	Defects exist in codes
Testing Phase	V&V/ Review & Audit activities in testing phase	Defects exist in executable binary code

2.3 Life-Cycle BN Model

Two SLCs are modeled by linking a series of phase BNs and their classes. Figure 4 is the BN model of a spiral software lifecycle. The example in Figure 5 is the BN model of a waterfall lifecycle. Each phase BN in a SLC model is connected by a time-indexed variable. With the help of DBN we can link one of the parents of a time-indexed variable to a variable from the previous time frame.

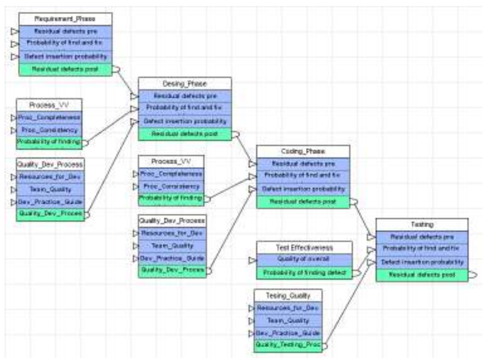

Fig. 4. BN graph for Waterfall lifecycle model

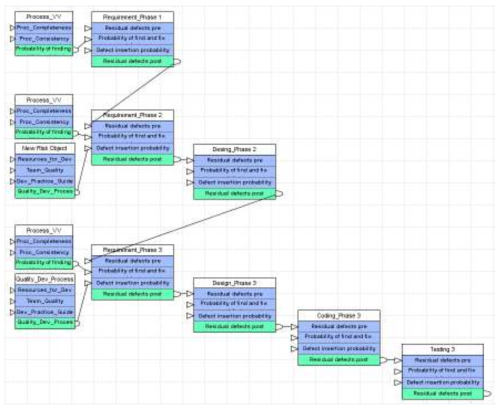

Fig. 5. BN graph for Spiral lifecycle model

3. Summary and Future Works

Object-Oriented BN simplifies laborious and repetitive works in constructing a large BN, and Dynamic BN enables a BN modeling of various SLCB by combining all the development phases. The proposed method will be applied to the modeling of reliability assessment of the safety-critical software which will be embedded in a reactor protection system developed by the KNICS project.

Acknowledgement

This research was supported by “The Mid and Long Term Nuclear R&D Program” of Ministry of Science and Technology, Korea.

REFERENCES

[1] H. S. Eom, BBN based Quantitative Assessment of Software Design Specification, KNS 2007 Spring, 2007.
[2] N. E. Fenton, Software Measurement: Uncertainty and Causal Modeling, IEEE Software 10(4), 116-122, 2002
[3] F. V. Jenson, Introduction to Bayesian Networks, UCL Press, 1996
[4] D, Koller, A. Pfeffer, Object-Oriented Bayesian Networks, Proceedings of the 13th Conference on Uncertainty in AI, Providence, Rhode Island, USA, 1997
[5] O. Bangs, Top-down construction and repetitive structures representation in Bayesian networks, In Proceedings of the 13th International Florida AI Research Symposium Conference, Florida, USA, 2000.