"Dimitrie Cantemir" Christian University
Knowledge Horizons - Economics
Volume 11, No. 1, pp. 47 – 55
P-ISSN: 2069-0932, E-ISSN: 2066-1061
© 2019 Pro Universitaria
www.orizonturi.ucdc.ro

METHOD OF EVALUATION AND CERTIFICATION OF ECONOMICAL SOFTWARE RELIABILITY

Marian Pompiliu CRISTESCU

"Lucian Blaga" University of Sibiu, Romania, Email: marian.cristescu@ulbsibiu.ro

Abstract *This paper present the efficient development of propagation algorithms followed by the availability to easily employ certain commercial software systems, as well as the increase of the collaborative applications number have turned the Bayesian Belief Networks (BBNs) into attractive models for various fields researchers, as it has been used since the beginning of the 1990s. After two decades of studies, Bayesian Belief Networks were successful in developing models for fields like: artificial intelligence, medical diagnosis and computer networks analysis.*

Key words:
economical software systems, software reliability, software components, Bayesian Belief Networks
JEL Codes:
O31
O32
O33

1. INTRODUCTION

Bayesian Belief Networks are a general frame for the networks used in representing and analyzing models involved in the study of uncertainty. The *BBN* concept derives from fields like probabilities, artificial intelligence and decision analysis. In (Vouk, 2002), *BBNs* are defined as a main and efficient frame that may be used to understand lack of belief (uncertainty) issues. *BBNs* exploit the conditional and independent relationships necessary to create natural and compact models in various domains that would offer useful templates, probabilistic interfaces and learning algorithms. One of the main reasons for having developed *BBNs* is that they offer the possibility to use models and hypotheses related to the lack of belief. Typically, in *BBNs* modeling, we assign a Bayesian belief value to each uncertain event, such as: the belief value associated to the sentence "the statement is unclear" is of 0.55 while the belief value assigned to the sentence "the student is late" is of 0.2.

All these probabilities, associated to uncertain events, derive from people's subjective thinking, which is influenced by the empirical, historical and statistical data collections. From the engineering projects' point of view, these probabilities may be determined by experts in the field, like project managers, developers, programmers and persons involved in the testing activity. When modeling *BBN*, the probabilities of representing the lack of belief are associated to the priority belief values for each node in a *BBN*.

2. RELIABILITY MODELING WITH BAYESIAN BELIEF NETWORKS

This type of modeling has become more and more popular when representing lack of belief in engineering, artificial intelligence and statistics. Applications that use *BBNs* are present in areas like: analysis and command systems of nuclear reactors, consumers' safe evaluation and help systems, medical diagnosis systems, templates recognition, and computer networks analysis and data fusion (Pham, 2000).

When *BBNs* are used as modeling tools, the following aspects must be taken into consideration:

- calculations regarding the combined probability distributions are not possible in the event of extended problems;
- the conditional independence implies reducing multiple combinations;
- a conditional probability table offered by experts is much more easy to use than a join probability;
- *BBNs* causal structure is much easy to understand than the mathematical structure;
- rapid algorithms are available for compiling and executing *BBNs*;
- the clarity is spread through the entire network by exploiting Bayes' rule;
- the predictions cannot be made if the records are incomplete;
- Bayesian analysis may be used both for "forward" and "reverse" reasoning's.

3. OFF-THE-SHELF COMMERCIAL PROMOTING METHODS FOR SOFTWARE SYSTEMS

A major problem in the reliability study is the manner in which certain commercial components (**C**ommercial **O**ff-**T**he-**S**helf Systems) may be used within program systems destined to safety-critical activities (Cristescu, 2001). Users agree to use commercial off-the-shelf systems in safety non-critical systems, such as text processors; they are extremely cautious with the system's reliability level when the commercial off-the-shelf components are largely used in military program systems, nuclear industry and other safety critical systems (Kontio et al., 2005).

Very few users agree to use these components in such critical fields in the absence of intense activities that would ensure an acceptable level for the reliability of the **C**ommercial **O**ff-**T**he-**S**helf Component Systems. Even if the development plan is fast, and the cuts of the budget allotted to a system programs based on commercial off-the-shelf components is highly attractive, the use of such components inside system programs meant for critical domains is still not accepted due to the uncertainty level and to the reliability of the components.

Each commercial off-the-shelf component has a strict functionality, and the objectives of using COTS components inside a system programs aim to reduce the time and risk associated to a new product by cutting off or eliminating the impact of the new design manner and development effort.

COTS-based system programs offer users several versions of the same product at a lower price.

Producers and users belong to different organizations. Users do not use solely the concept of choosing random COTS components, but develop a new system by integrating COTS components with specific interfaces. The functionality of COTS components , of the project's necessities and of the resulted outputs must be completely understood before those commercial off-the-shelf components are integrated to the target system.

The certifications of the awareness (Kontio et al., 2005) of a software product that comprises commercial off-the-shelf components does not imply a "yes" or "no" type question, but a "at what level" type question, and the *BBNs* are the ideal tool for modeling this process. Thus, a Bayesian Belief Network can be developed, based on priority knowledge regarding the commercial off-the-shelf components and historical information provided by those in charge with selling software products. In addition, the *BBNs* may be updated by using information collected during the certification process, and the outcome may be the reliability of the entire system programs.

During the certification process, it is recommended the use of object oriented *BBNs* (*OOBBN*), as three main aspects of the lack of belief that can be associated to the system programs comprising commercial off-the-shelf components have been identified. They are: the lack of belief when choosing components, the lack of belief in the components' quality and the lack of belief in the components' impact on the system.

These aspects are interdependent, and the users encounter them sequentially over the process of choosing software products. It is not easy to find an answer to questions like: "is this component the desired one?", "is the quality of the software high enough?" and "what is the impact of the component over the system?", as way to many problems derive from these aspects related to the lack of belief. *OOBBN* offers the structure to model any aspect of the lack of belief, separately for each *BBN*; thus, each *BBN* subnetwork is connected to the network with the highest level through target nodes.

Figure 1 presents a high level BBN.

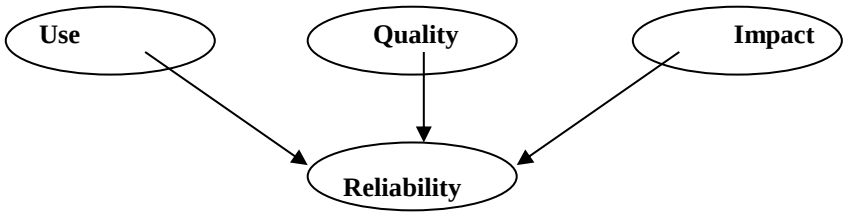

Figure 1. High level BBN

The use, quality and impact are the main factors that contribute to establishing the reliability of commercial off-the-shelf components based system programs. These nodes act as target nodes within BBN subnetworks. Each BBN subnetwork

models the main contribution factors of the specified component.

4. THE CHARACTERISTICS OF THE CERTIFICATION PROCESS

4.1. USE OF SOFTWARE SYSTEMS

Use may be understood in an extended sense, as there are many factors influencing it.

Figure 2 presents the BBN subnetwork and its most important influencing factors.

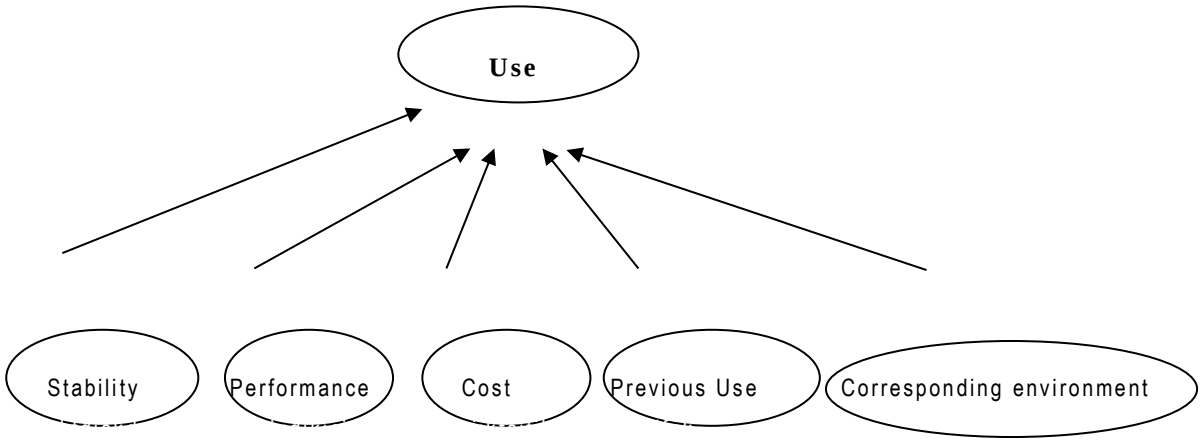

Figure 2. The first BBN subnetwork: Use

Stability: the stability is highly critical for real time systems, such as: controllers, operating systems and monitoring systems, as the deployment of these systems is extensive and they are usually used with the same functionality over long periods.

Performance: when users purchase an off-the-shelf software component, they do not purchase only its function but also its performance, especially when other competitive companies are on the market.

Life cycle cost: when over a year the system's maintenance cost is higher than the price of a new system having the same functionality (or even a better one), there is no reason to keep maintaining the old system. Nevertheless, users pay far too much attention to the costs of a new system and less care to the money they spend each year on maintaining the old system. The life cycle cost of a system is as important as the cost of the off-the-shelf components based system.

Previous use level: off-the-shelf software components must have a recorded history related to their performance level, especially when they have been on the market for a long period of time. The previous use corresponding level must be recorded as knowledge associated to the previous use of the node. In order to measure the previous use level, the mean time to failure (MTTF) is usually applied, as it is the most popular among reliability engineering and the easiest method to obtain such information from those in charge with selling software products.

Corresponding environment: the corresponding environment is a remarkable particularity for commercial off-the-shelf components based system programs, as the best performances of an off-the-

shelf software component may only be reached, for instance, when it is used on a unique OS. It is the users' responsibility to ensure a corresponding user environment for off-the-shelf software components. We must point out that an off-the-shelf software component may offer certain functions that the users need not, so if the requested function runs correctly in the user environment and the other functions do not, the component is nevertheless accepted.

4.2. SOFTWARE QUALITY

What matters here are the system programs quality and not the COTS quality? The covering technique (Vouk, 2002), is at present the most used technique in integrating system programs comprising off-the-shelf software components. According to (Vouk, 2002), a system programs created with this technique is a software product that "surrounds" a component and limits all its functions. Incompatibilities among system's requirements and component's functionalities are inevitable. The "layer" ensures the interface between the system and off-the-shelf software components (Ivan et al., 2003); in this manner, no relative modifications to a component's functionality must take place, and the systems do not have to suffer changes in order to eliminate the incompatibility. Another advantage of using the layer is the easiness in bringing modifications in case of needing to update an off-the-shelf software component or changing the system's requirements; the layer may be changed so that it would keep the functionality of the system programs.

Figure 3 presents the BBN subnetwork due to the quality.

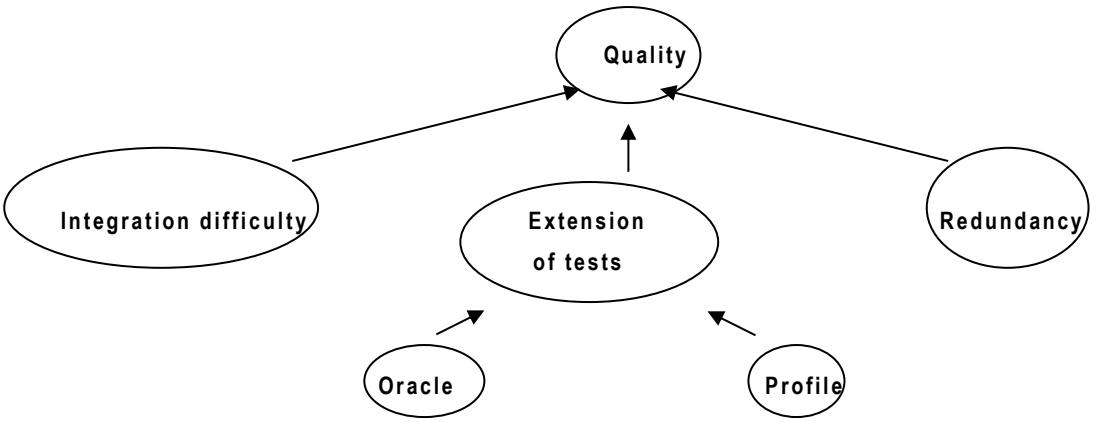

Figure 3. The third BBN sub network: Quality

Integration difficulty: an integration problem may reduce the time saved following a fast response, using off-the-shelf software components to solve an operational necessity, and, sometimes, the incomplete and ambiguous information related to the system's requirements may turn the off-the-shelf software components incompatible with the analyzed system programs. The layer may fix this situation as it conceals the interface of a COTS component through the interface of different parts of

the system; nevertheless, the layer implies certain expenses as it necessitates maintenance over the system's lifespan and modifications when the functionalities of the system are improved.

The integration difficulty measures the effort to assembly COTS components and to make the overall system programs function in accordance with the specified functionalities, using the covering technique (Vouk, 2002) for this purpose.

Extending the tests: typically, companies marketing COTS components do not provide users the main code to their products. Therefore, black-box testing is the only solution to test system programs with COTS components. These components must be tested in their new environment. The good functioning of COTS components cannot guarantee the functioning of the entire system. To supervise the black-box testing, users must carefully select the adequate oracle and an inclusive testing profile. These two factors may establish the extension level of tests.

System redundancy: quality systems should have a certain redundancy project, recommended especially to system programs used in safety-critical environments (Weinberg, 1993). The quality of system programs comprising COTS components may be designed with redundant elements, using duplex components and available multi functional components.

4.3. IMPACT, INDICATOR USED IN EVALUATION AND CERTIFICATION

After successfully browsing several tests, the system programs comprising COTS components are ready to be launched onto the market. At that moment, the evaluation of the system's reliability is only half-way made. Ideally, the users would want the system programs comprising COTS components to execute the certified functions as long as specified, and the system to be updated when a new versions appears on the market. Reaching such an objective is a difficult task as the prediction of the functioning period of a system and updating new products require high understanding of current techniques and of development tendencies of new products. According to (Vouk, 2002), the BBN sub network associated to the impact may be split in three nodes: *lifespan, integrated logistics support* and *updating effects (*figure 4*).*

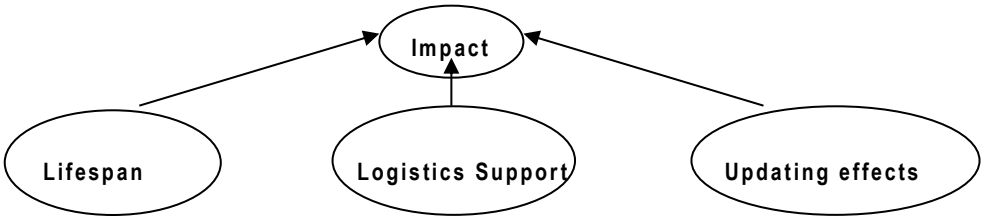

Figure 3. The forth BBN sub network: Impact

Lifespan: the continuous availability of off-the-shelf software components should be guaranteed when they are put on the market. Users use exclusively the functionality of COTS product; they have no idea about what is inside the product. When a problem occurs, even if it is very simple, users cannot solve it without the source code. Prior to launching a system programs comprising COTS components onto the market, one must take into account the possibility that the person marketing the product might want to give up marketing it. When it is impossible for the user to obtain the source code for the COTS components - and this is a very common situation - he should negotiate with the product seller the right to the product's support sources (Voas, 2008).

Integrated logistics support. Testing is a dynamic evaluation method for the reliability of the system programs. There are several formal methods used to evaluate the reliability of the target system, such as: types of errors, analysis of effects, analysis of error tree and analysis using stochastic Petri Nets. The outcomes of such analysis are represented in this node (Vouk, 2002).

Side effects of updating: the update of COTS components may bring unwanted additional functions or may eliminate the needed ones; even the needed functions may disturb the elements controlled by certain functions of the system. To preserve the system's reliability level, such unpleasant situations must be avoided. To ensure a gradation of the side effects generated by updates, it is recommended to collect experts' opinions, and to use the evidence of real cases may to upgrade the effects (Vouk, 2002).

5. CASE STUDY

The following example describes the manner in which a Bayesian Belief Network may be populated, how the discovery and use of records may lead to the improvement of probabilities, and how to use information.

First step: Populating nodes probability tables (***NPT*** = **N**ode **P**robability **T**ables)

Once the components and the relationships have been established, the next step in building a BBN is to populate nodes probability tables. Figure 5 presents the node probability table represented by the BBN subnetwork associated to the use.

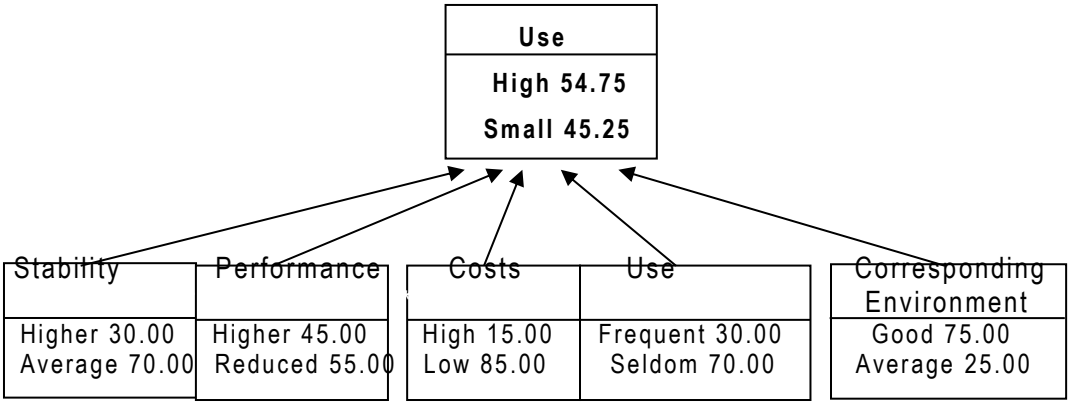

Figure 5. Example of table for BNN sub network associated to the use

The nodes from this example are binary nodes: each node has two statuses, and hypothetical values are used for the nodes statuses and tendency transposition. The values are associated based on the quality analysis which usually represents the outcome of software engineering and experts' opinions deductions.

There are two statuses associated to the node *stability:* **high** and **average**. The value that discriminates these two statuses is 0.65, which means that if we consider that the stability of a system programs comprising COTS components is higher or equal to 0.65, then the system is in the higher status of the node *stability*; otherwise the system is in the average status. The initial probabilities for the *stability* nodes are presented in figure 5. There is a 70% probability that the stability be under 0.65 and a 30% probability to the contrary.

The same method may be used to interpret the other nodes from figure 5**.** The value used to discriminate the two statuses is not displayed.

Second step: Collecting records

It is assumed that during process certification two samples of records were obtained: it is known for sure that the stability is high and that the previous use of this COTS component is seldom. The knowledge available about other factors remains unchanged.

Third step: Updating BBNs

The nodes are updated based on the record samples existing at that particular moment. The probabilities for the node stability change from <higher, average = 30.70> into <higher, average = 100.0>; the probabilities for the node *previous use level* change from <often, seldom = 30.70> into <often, seldom = 0.100>

Using marginal probabilities and the linkage rule, the records are propagated to the target node *use.* The probabilities of the target nodes statuses change from <high, reduced = 54.75, 45.25> into <high, reduced = 66.67, 33.33>.

The updating probabilities of the node *use* statuses are afterwards propagated to the target node of the high level BBN.

The updating and propagation processes of the record may normally be continued until reaching the specified level of acceptability on the final target node ***reliability***

6. CONCLUSIONS

Building a Bayesian network to model a system needs a distinct notion over the system's functioning manner. Thus, it is necessary to know all the system's relevant variables and to have exact knowledge on the interacting variables.

When software manufacturers shift from the method implying writing each code line to the method implying creating collaborative systems using existing components, they have to deal with new problems. The priority problem that concerns software industry companies is the one referring to the certification of system programs based on commercial off-the-shelf components.

The reliability certification process for collaborative system programs implies the evaluation of its ability to execute the requested function in a specified environment and to be used for the entire planned span life.

REFERENCES

- Cristescu M.,"*The tools of software reliability estimation*", Academia de Studii Economice Bucureşti, The 5-nd internaţional symposium on economic informatics, Bucharest, May 13-15, 2001;
- Ivan I., Saha P., "*Quality characteristics of The Internet Applications*", in DIGITAL ECONOMY - The Proceedings Of The Sixth International Conference On Economic Informatics, Bucureşti, may, 2003;
- Kontio J., Chen S., Limperos K., "*A COTS Selection Method and Experiences of ItsUse*", Proceedings of the 20th Annual Software Engineering Workshop, NASA,Greenbelt, Maryland, 2005;
- Pham H., "*Software Reliability*", New York, Berlin, et al., 2000;
- Voas J., "*An Approach to Certifying Off-theShelf Software Components*", In IEEE Transactions on Computer, June 2008, pp. 42-51;
- Vouk M.A., "*Using Reliability Models During Testing With Non-Operational Profiles*", 2002, in http://www.renoir.csc.ncsu.edu/Faculty/Vouk/Papers/non-op-testing.ps
- Weinberg G.M., "*Quality Software Management - System Thinking*", Dorset House Publishing, New York, U.S.A. 1993;.