Available online at www.sciencedirect.com
SCIENCE DIRECT®
The Journal of Systems and Software 73 (2004) 193–203
The Journal of Systems and Software
www.elsevier.com/locate/jss

BBN-based software project risk management

Chin-Feng Fan [a,*], Yuan-Chang Yu [b]

[a] Computer Science and Engineering Department, Yuan-Ze University, 135 Far East Road, Chung-Li 320, Taiwan
[b] Nuclear Instrumentation Division, Institute of Nuclear Energy Research, Long-Tang, Taiwan

Received 2 November 2002; received in revised form 27 December 2003; accepted 27 December 2003
Available online 28 February 2004

Abstract

This paper presents a scheme to incorporate BBNs (Bayesian belief networks) in software project risk management. A theoretical model is defined to provide insights into risk management. Based on these insights, we have developed a BBN-based procedure using a feedback loop to predict potential risks, identify sources of risks, and advise dynamic resource adjustment. This approach facilitates the visibility and repeatability of the decision-making process of risk management. Both analytical and simulated cases are reported.
© 2004 Elsevier Inc. All rights reserved.

Keywords: Uncertainty; Software project risk management; BBN (Bayesian Belief Network); Risk profile

1. Introduction

There exist many uncertainties in software development processes and products; for instance, the uncertainties in estimating project size, schedule, quality, and in determining resource allocation. Current software engineering techniques cannot eliminate such uncertainties. Thus, risk management (Boehm, 1991; Charette, 1989; Karolak, 1996) is critical. Kitchenham and Linkman (1997) proposed that uncertainty estimates were "best managed across an organization's total portfolio". This implies that resources should be adjusted or reallocated among several projects so as to enhance the probabilities of project success. However, it may not be possible to manage several projects at the same time and share resources among them. Subjective ways to manage project risks are common in current practice. Such approaches are human-intensive and opaque. This paper shows that Bayesian Belief Networks (BBNs) can be utilized to provide more objective and visible support for decision making in risk management.

We have designed a procedure to incorporate BBNs in a continuous monitoring loop to support the decision-making process of risk management. The rationale for this approach is as follows:

1. Risk management should be performed continuously in a feedback loop so that problematic situations can be dynamically detected and adjusted.
2. BBNs' influence diagrams can visually model cause consequence relations, and thus, help to identify sources of risks.
3. BBNs can model uncertainties and provides probabilistic estimates. Whenever new evidence is available in the proposed monitoring loop, the new data can be plugged in the related BBNs again to recalculate and update previous estimates.

Our approach may seem to be similar to that of system dynamics model of software development process (Abdel-Hamid and Madnick, 1989). However, instead of simulation, we combine BBNs in an algorithmic way to identify, predict and estimate risks in a probabilistic fashion. It is more appropriate to model and calculate uncertain information using BBNs. Moreover, instead of deterministic results typically generated by simulation, BBNs may yield a set of probabilistic results at each run.

In the following, we first define our theoretical model, which provides insights into resource allocation. We then

* Corresponding author. Tel.: +88-63-463-8800x3; fax: +88-63-463-8850.
E-mail address: csfanc@saturn.yzu.edu.tw (C.-F. Fan).

0164-1212/$ - see front matter © 2004 Elsevier Inc. All rights reserved.
doi:10.1016/j.jss.2003.12.032

present a scheme to incorporate BBNs in risk management, followed by verification of its effectiveness.

2. Bayesian Belief Network (BBN)

BBNs (Heckerman and Wellman, 1995; Jensen, 1996) have attracted a lot of recent attention in the area of decision support under uncertainties (Bouissou et al., 1997; Fenton and Neil, 1999; Ziv and Richardson, 1997). BBNs' underlying theory (Bayesian probability) has been around for a long time; while the implementation algorithms (Jensen, 1996) and software tools [1] are available in these few years. A Bayesian Belief Network is an acyclic graph with an associated set of probability tables. Nodes in a BBN represent random variables, whose states are usually expressed in discrete numbers or ranges. Arcs represent the causal relationships between the variables. A Conditional Probability Table (CPT) is associated with each node to denote such causal influence. The node representing a variable A with parent nodes $B_1, B_2, \ldots, B_n$ is assigned $P(A|B_1, B_2, \ldots, B_n)$ as its CPT. CPTs are filled using a mixture of empirical data and expert judgment. When root nodes are unknown, current tools usually assign evenly distributed probabilities to these roots. Once new evidence is obtained, evidence can be plugged in the graph. Then, re-calculation and updating of node values are performed, propagating from parent nodes to child nodes and vice versa. A BBN graph can be expanded into an influence diagram by adding decision nodes and utility (cost, or profit) nodes, represented by rectangles and diamonds, respectively. Fig. 1 is a simple example, where *product quality* is influenced by *manager capability* and *developer capability*; also, *training* is a decision node with the associated *training cost*, a utility node. The CPT for node *product quality* is shown in Table 1.

BBNs are criticized for the subjectivity in constructing its influence diagrams and CPTs. In general, a BBN models the constructor's belief. Based on this belief, it provides mathematical calculation and prediction. BBNs can be used to support visible and repeatable decision-making. Ziv and Richardson (1997) has used BBN in software testing and maintenance. Fenton and Neil (1999) suggested using BBNs for complex software metrics.

3. A theoretical model of project risk management

We first define a theoretical model for risk management so that we may have in-depth understanding of it. A software project involves a set of activities

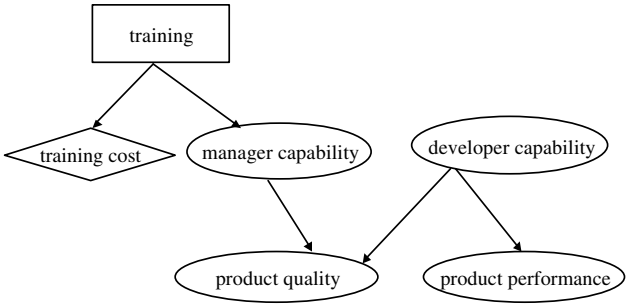

Fig. 1. BBN example.

Table 1
Sample CPT for product quality

Manager capability	High		Low	
Developer capability	High	Low	High	Low
Pr('product quality' = 'High')	0.9	0.85	0.35	0.15
Pr('product quality' = 'Low')	0.1	0.15	0.65	0.85

$a_1, a_2, \ldots, a_n$, such as development activities, review, testing, QA, etc. Project risk management should monitor this set of activities to assess problematic situations and formulate solutions. Since most projects are allocated with limited resources, a large part of risk management is to ensure project activities are performed in a cost-effective way in terms of reducing project risks.

In general, when more resources are spent on a development activity, the activity costs (C) will be higher and the project risk (R) may be lower. Estimation of risk can be defined as the expected value of the occurrence probability (Prob) of an undesired outcome multiplied by its damage cost.

$$R = \text{Prob} * \text{Damage cost}$$

In the definition, we exclude activity expenses from damage costs. The reason is that we are dealing with dynamic risk management, which adjusts activities dynamically; and thus, the varied activity costs are our major concerns and will be treated separately. There exists a certain point after which the more resources spent on an activity, the less productive the activity becomes. But the activity costs (C) will always accumulate and increase as long as the activity is still going on. Therefore, the combined costs L combining risk and associated activity costs ($L = R + C$) should be a concave shape as shown in Fig. 2. The concave shape of L has an optimal or a lowest point, at which risk and activity expenses reach a balance.

If we extend this analysis to the whole project, then we have the following equations:

Total project risk

$$R = R(a_1, a_2, \ldots, a_n)$$

where $a_1, a_2, \ldots, a_n$ represent different activity effort.

[1] Hugin, distributed by Hugin Expert, www.hugin.dk.

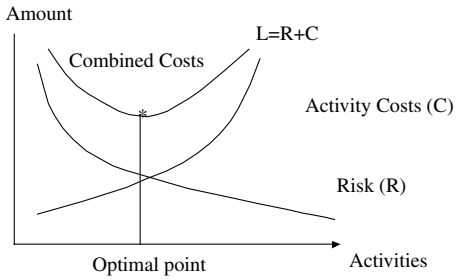

Fig. 2. Costs and risk relation.

Total project activity costs

$$C = C(p_1a_1, p_2a_2, \ldots, p_na_n) = \sum_{i=1}^{n} p_ia_i$$

where p_i refers to the unit price of activity a_i.

Total combined cost

$$L = p_1a_1 + p_2a_2 + \cdots + p_na_n + (R(a_1, a_2, \ldots, a_n) - R_0)$$

where R_0 is the smallest risk that cannot be eliminated.

The conditions for L to be optimal are as follows:

$$\frac{\partial L}{\partial a_1} = \frac{\partial C}{\partial a_1} + \frac{\partial R}{\partial a_1} = 0$$
$$\frac{\partial L}{\partial a_2} = \frac{\partial C}{\partial a_2} + \frac{\partial R}{\partial a_2} = 0$$
$$\ldots$$
$$\frac{\partial L}{\partial a_n} = \frac{\partial C}{\partial a_n} + \frac{\partial R}{\partial a_n} = 0$$

Thus, for an individual activity, the optimal point of L is that the activity's marginal cost is in proportion to its marginal contribution (risk reduction).

$$\frac{\partial C}{\partial a_1} = -\frac{\partial R}{\partial a_1} \tag{1}$$

When L is optimal, the relationship of all activities is as follows:

$$\sum_{i=1}^{n} \frac{\partial C}{\partial a_i} = -\sum_{i=1}^{n} \frac{\partial R}{\partial a_i} \tag{2}$$

Eq. (1) divided by Eq. (2) gets Eq. (3).

$$\frac{\dfrac{\partial C}{\partial a_i} \Big/ \displaystyle\sum_{i=1}^{n} \frac{\partial C}{\partial a_i}}{\dfrac{\partial R}{\partial a_i} \Big/ \displaystyle\sum_{i=1}^{n} \frac{\partial R}{\partial a_i}} = 1 \tag{3}$$

Eq. (3) indicates that the best situation is that the ranking of marginal costs of all activities equals the ranking of their marginal risk contributions (reductions). This relation will be used in the following resource adjusting scheme.

4. BBN-based software project risk management

We propose using BBNs to support decision making of software project risk management. The system context of our method is shown in Fig. 3. In this context, our procedure utilizes BBNs to analyze risks and generates information to the manager; while the manager may input evidence or decisions to the BBNs for further estimation and prediction. For continuous risk management, a feedback loop is needed. Thus, BBNs are employed in this continuous loop. BBNs are updated in each iteration with new project data, and then generate new estimates. A risk profile is also needed to record chronological inputs and state probabilities. In addition, there is a knowledge base of risk treatment.

Basic factors affecting software project risks can be organized first as a BBN template. A simple sample is shown in Fig. 4. The square boxes in the figure are the nodes that can be further expanded. We categorize factors into two groups:

(1) organization-related factors,
(2) project-related factors.

Organization factors involve factors such as organization's risk culture and practice, management experience and capability, as well as process maturity. Project factors include development quality at each stage of this project. A quality node in the diagram is influenced by such factors such as the quality of the preceding stage's artifacts as well as this stage's schedule, budget, developer capability, and techniques. The quality of each development stage has a similar influence diagram. Design quality, for example, can be expanded into the graph shown in Fig. 5. The CPTs of these BBNs are not shown here. They can be assigned by a group of experts.

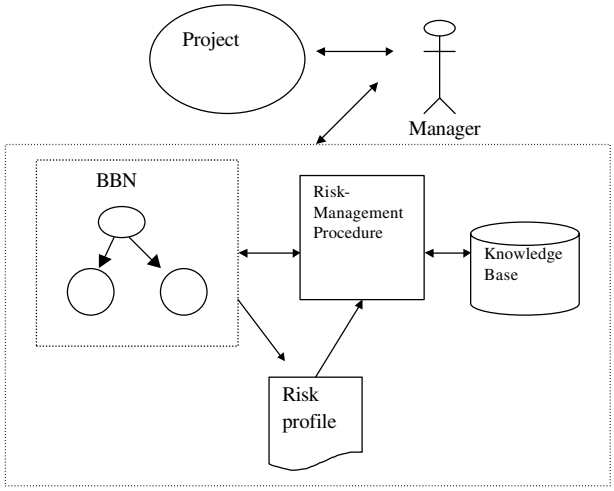

Fig. 3. Context of our approach.

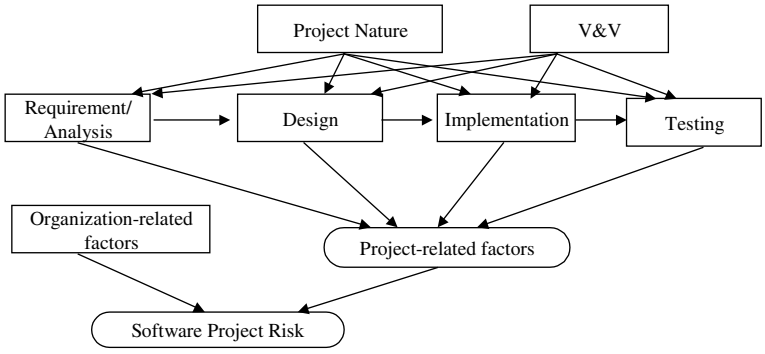

Fig. 4. Basic BBN template for software project risk.

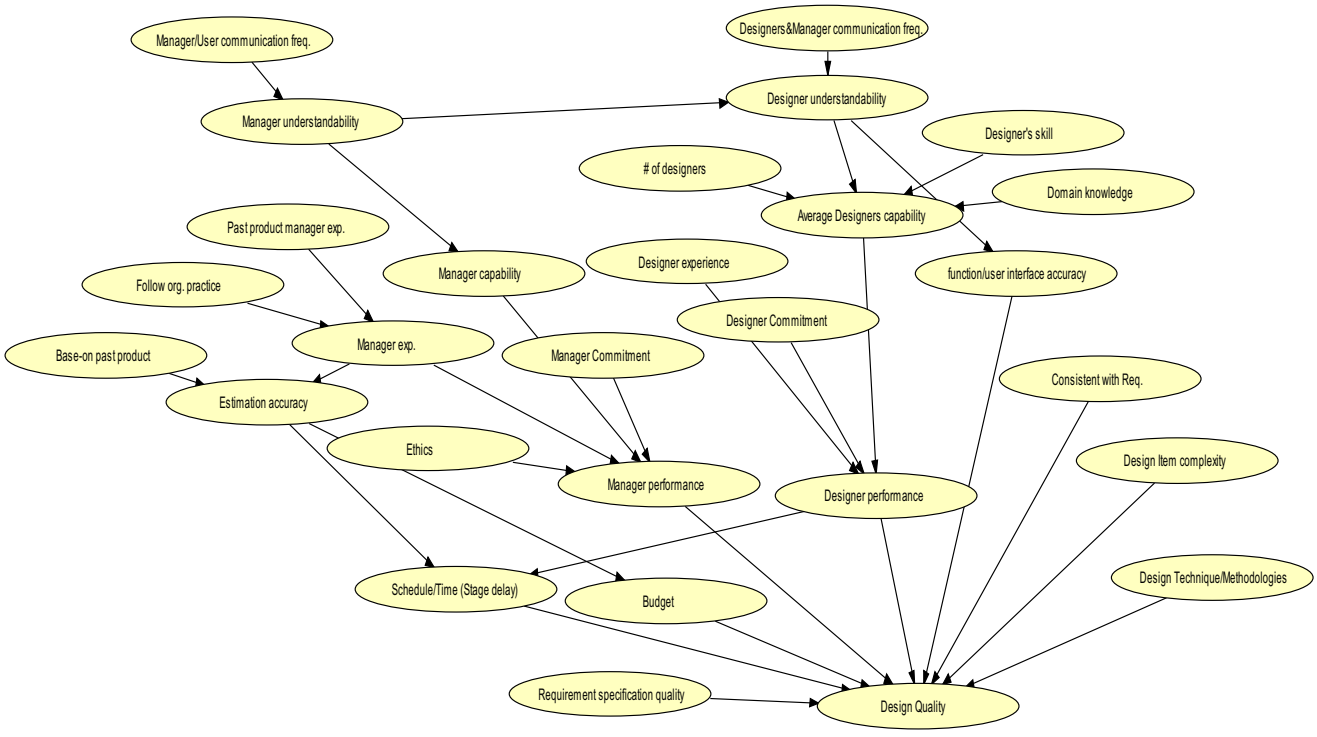

Fig. 5. Factors influencing design quality.

Our proposed BBN-based risk management procedure is given in Fig. 6. Similar to the tasks recommended by IEEE Standard 1540 (2001), our steps are grouped into the following tasks:

(1) Initialization,
(2) Maintaining Project Risk Profile,
(3) Performing Risk Analysis and monitoring,
(4) Performing Risk Treatment.

The procedure is explained below.

Step 1 (*BBN initialization*): Expand the basic BBN template to further comprise factors specific to the examined case. Assumptions and expected states are saved. Nodes to be monitored are also set.

Step 2 (*Risk profile*): Once the project starts, a continuously monitoring loop starts. Whenever new project data are obtained, they are plugged into the BBNs to update estimates. A chronological record of such inputs and estimates are kept as a risk profile. In other words, a risk profile can be viewed as historical snapshots of these BBNs' images.

Step 3 (*Risk monitoring and analysis*): The following sub-steps will be performed independently.

Step 3.1 (*Correct pre-assumptions*): Assumptions need to be right in the first place so that further analysis can be correct. *Tracing Module* will be invoked when the average of previous N units of the profile records for a particular node conflicts with its expected states or assumptions. The *Tracing Module*, shown in Fig. 7,

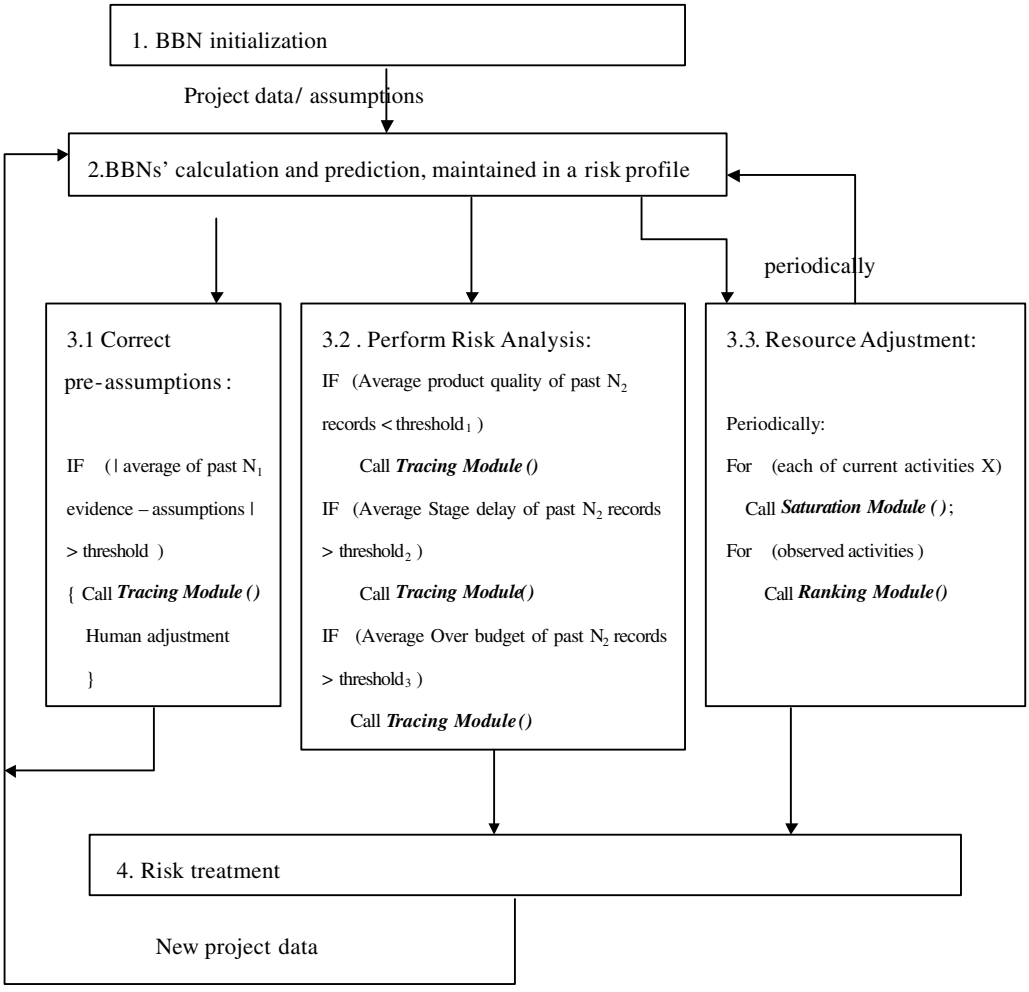

Fig. 6. Risk management procedure.

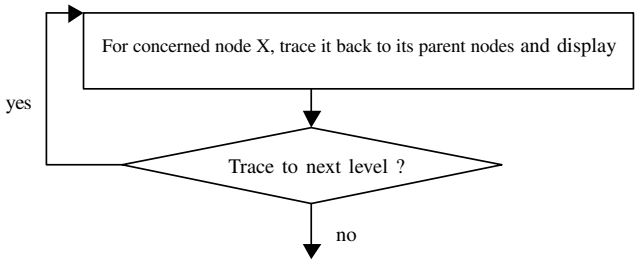

Fig. 7. Tracing module.

traces the observed node to its ancestors interactively with the user to identify potential erroneous assumptions. Then, human correction is needed.

Step 3.2 (*Risk analysis*): In the generic BBN template, we have risk-related nodes, such as *stage delay*, *product quality*, and *budget*. The procedure analyzes the average of the predicted risks of the previous N time units, to identify whether there exist such risks as behind schedule, over budget, and poor quality. If so, the procedure invokes *Tracing Module* to locate their causes.

Step 3.3 (*Resource adjustment*): As mentioned above, resources utilized in a software activity may have a saturation point. From this point on, the more resources this activity uses, the less productive the activity becomes. In such a saturation state, resources should be adjusted. Hence, periodically, say daily or weekly, our procedure invokes *Saturation Module* (Fig. 8) to check whether the saturation points of observed activities have been reached. To avoid possible transient situation, *Saturation Module* continues to monitor for M more time units before reaching a final conclusion.

To better utilize available resources, *Ranking Module* (Fig. 9) is invoked periodically. Assume that activities expenses are always known. However, there are two different cases for activity effects:

(1) If these activities' effects have quantitative data, the effect per thousand dollars of each activity can be calculated and then the cost-effectiveness can be ranked straightforwardly.
(2) If there are no direct quantitative data, BBN estimates should be used to estimate their effectiveness.

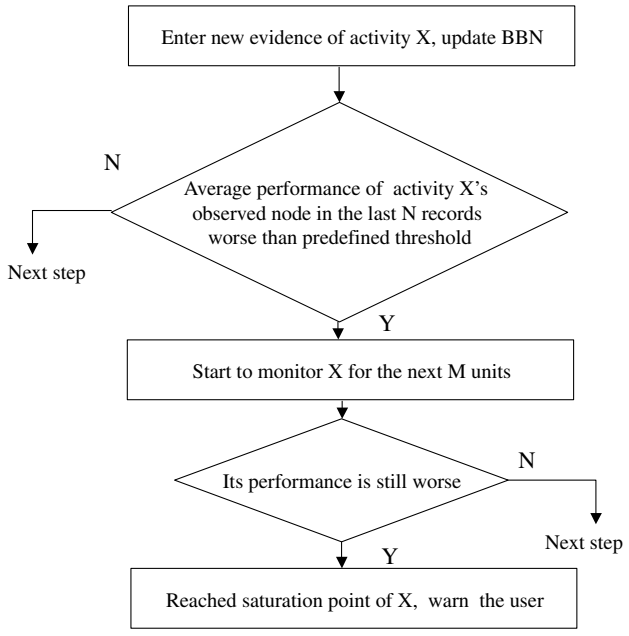

Fig. 8. Saturation module.

The ranking of these activities' effectiveness and the ranking of their costs are sorted in separate sets. According to the theoretic model in Section 3, the best situation is that these two sets of ranking order are identical; otherwise, resource adjustment is needed to achieve better cost-effectiveness.

If resource adjustment is needed, Step 4 will be carried out; otherwise, the control returns to Step 2.

Step 4 (*Risk treatment*): The knowledge base of risk treatment can be consulted. BBN calculation can also be used to predict the effectiveness of risk treatment decisions.

5. Analytical cases

The above procedure was verified statically and dynamically. This section presents the static case studies. The next section addresses a dynamic test.

Two past software projects, Confirm project (Oz, 1994) and SizeWell B nuclear protection system (Nuclear Engng. Int., 1993), were studied. They represent two different situations: the former was a failed project due to technological difficulties and management problems; while the latter was a project with huge V&V effort, which well passed its saturation point. We used cases similar to these two to analyze what would happen if the proposed risk management scheme were used.

5.1. CONFIRM-like case

This case can be described briefly as follows. In 1988, Hilton, Marriott, and Budget Rent-A-Car subcontracted a large project to AMR Information Services, Inc. to develop a new system, CONFIRM, combining airline, rental car, and hotel information (Oz, 1994). AMR was famous for its SABRE, a very successful airline reservation system. After three-and-a-half years and a total of $125 million expense, the project was canceled. The major troubles of this project involved technological difficulties, personnel inexperience, as well as ethical and managerial problems.

If the proposed BBN-based risk management was used in a similar case, the potential progress might be as described below. The BBN diagrams would be the same as the generic ones shown in Figs. 4 and 5, except that we added a *user-cost estimation accuracy* node as a child of *estimation accuracy* node. Theoretically, the following scenario might happen in the first two years:

1. Initial assumptions of all nodes were all positive or good, due to AMRs previous success.

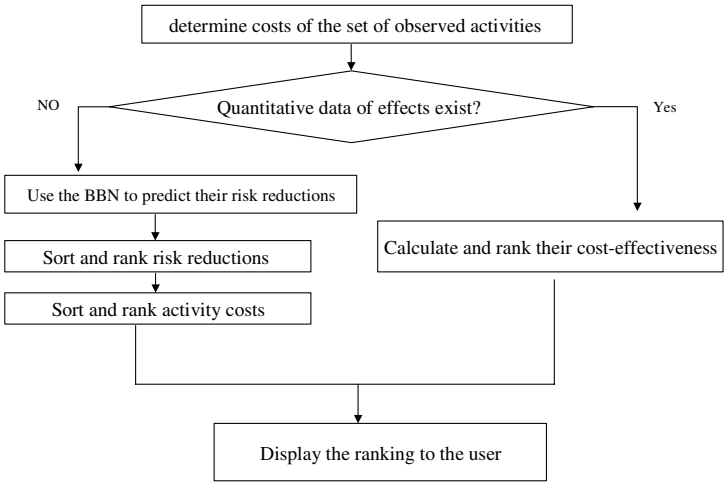

Fig. 9. Ranking module.

2. After the first year, the evidence of the project showed that the development budget was raised from original \$55.7 million to \$72.6 million, the estimated cost per reservation was raised from \$1.05 to \$1.3, the development plan was revised, and development team had to be recruited outside the firm. According to Step 3.1 of the proposed procedure in Section 4, the following BBN nodes were then adjusted as below:

   use-cost estimate accuracy = poor,
   over budget = yes,
   manager/user communication = poor,
   specification change = frequent.

3. Our procedure then assisted the user to correct erroneous initial assumptions. Since evidence '*User-cost estimation accuracy*' conflicted with assumptions, *Trace Module* was invoked (Step 3.1 of the procedure). Its result calculated by Hugin tool is shown in Fig. 10. The user might identify potential sources of the erroneous assumption, and then changed the nodes '*past product manager experience*' and '*follow organization practice*' to be low.
4. In the second year, the project development team missed the design deadline and users demanded many new individual features. Thus, the user would plug in the following evidence in the BBNs:

   specification change = frequent,
   stage delay = yes.

5. If *Stage delay* was worse than the predefined threshold. *Trace Module* was then invoked again (Step 3.2) and traced to the potential sources of this problem to be such nodes as *Designer experience*, *Designer understandability*, *Project size*, or *technical complexity*.
6. At this point, it was possible that the manager using our algorithm could realize that the project had personnel and technical problems, and an independent audit could be suggested by the knowledge base of risk treatment to find out further facts (Step 4).

This simplified scenario shows that it is possible to disclose potential project risks by continuously monitoring project situations and diagnosing problems using BBNs.

5.2. Sizewell B-like case

This case is described as below. In early 90's Britain's Sizewell-B nuclear reactor (Nuclear Engng. Int., 1993) invested huge V&V effort to assure the reliability of its digital primary protection system (PPS). This protection system was developed by Westing House, using structured design. The program was written in PL/M86, about 1000 procedures 100 lines per procedure. The project spent 200 man-years development effort and 50 man-year V&V effort. The following is a list of the V&V and IV&V techniques used in this project:

Westing House: Static analysis and dynamic test.
NNC Ltd.: Confirmatory analysis and independent design assessment.
TA Consultancy Services: MALPAS analysis.
NE Technology Division: Source to code comparator.
Rolls Royce and Associates (RR&A): Dynamic testing.

We analyzed a similar case assuming our procedure was used. V&V node in the generic BBN (Fig. 4) was expanded as that shown in Fig. 11. Except for the Westing House's V&V, the rest are IV&V (independent

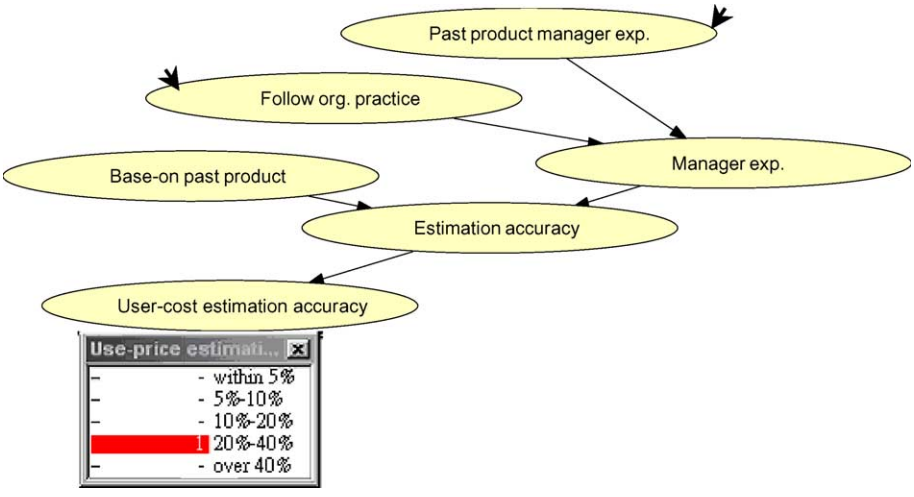

Fig. 10. Trace module applied.

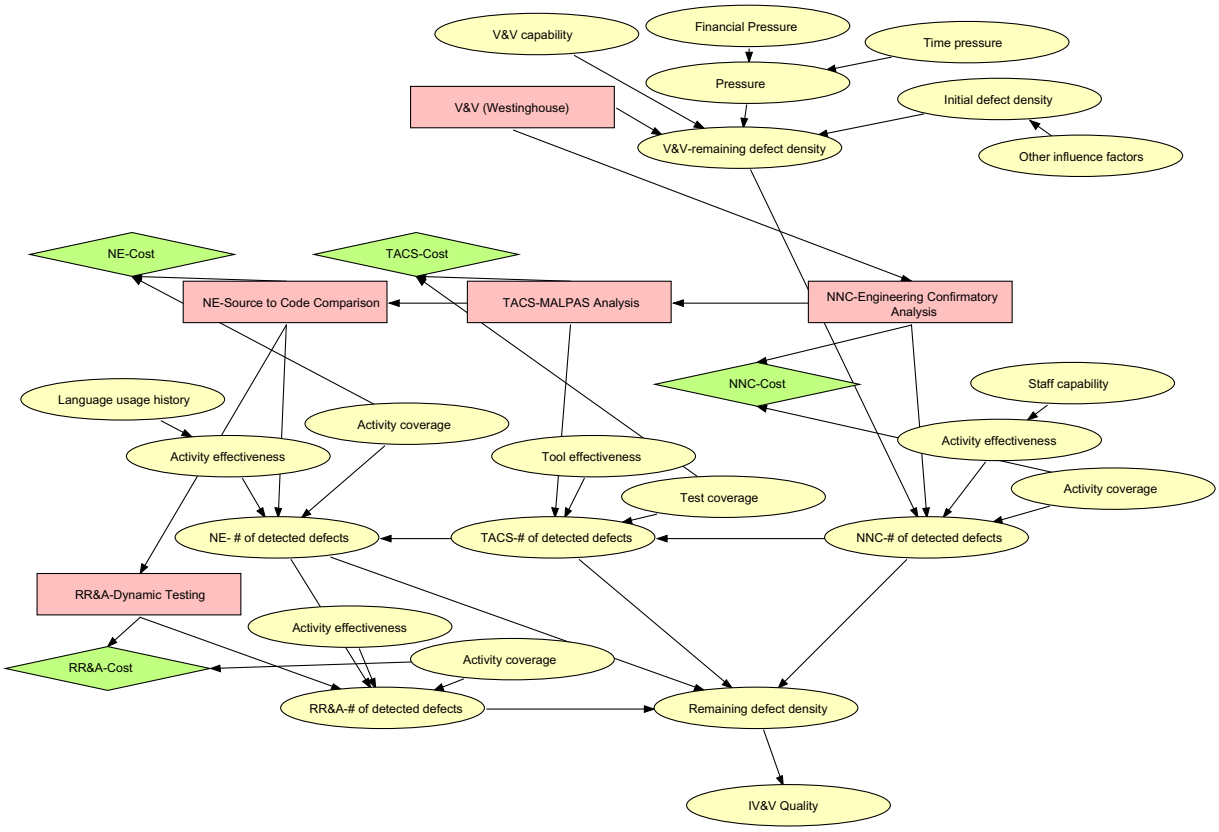

Fig. 11. Expanded BBN with Sizewell B's IV&V Activities.

V&V). As shown in the figure, the major differences between V&V and IV&V are that the latter has no schedule, financial, or management pressure. We might have the following scenario:

1. Due to Westing House's reputation, our initial assumptions were positive:

   Westing House's V&V capability = high,
   Initial defect density = low,
   schedule pressure and financial pressure = high.

2. Assume NNCs IV&V was performed. The following evidence could be gathered:

   Numbers of detected defects = low,
   No major safety-related errors were found in the past N units,
   According to Step 3.3 of the proposed procedure, *Saturation Module* was then invoked and confirmed that the IV&V activity reached its saturation point.

3. Similar analysis could be applied to the other three types of IV&V if they had started.
4. *Ranking module* could also be invoked. However, no distinctive differences could be found.

This case shows that it is possible to detect nonproductive activities in time using our algorithm. In this case, the manager could be warned early that V&V effort had been saturated and no further investment was justified. Thus, resources could be utilized for more productive activities to enhance project success.

6. Dynamic test

It is desirable to apply the proposed method to an ongoing project to dynamically verify the feasibility and effectiveness of the method. However, it is difficult to find such a real project for experiment. Thus, we implemented the proposed procedure and tested it using a simulated context as that depicted in Fig. 3. A simple simulator was constructed to simulate the progress of a hypothetic software project. The risk management loop fed data generated from the simulator to the BBNs, running on Hugin,[1] and then used BBNs' outputs for decision-making.

The BBN-based procedure is implemented in ANSI C, calling Hugin's API, and then results are plotted in Borland C++. For the simulated environment, there are some implementation issues. BBN states are probabilistic and discrete, yet many input data and algorithm thresholds are numerical. Thus, we use fuzzy triangular

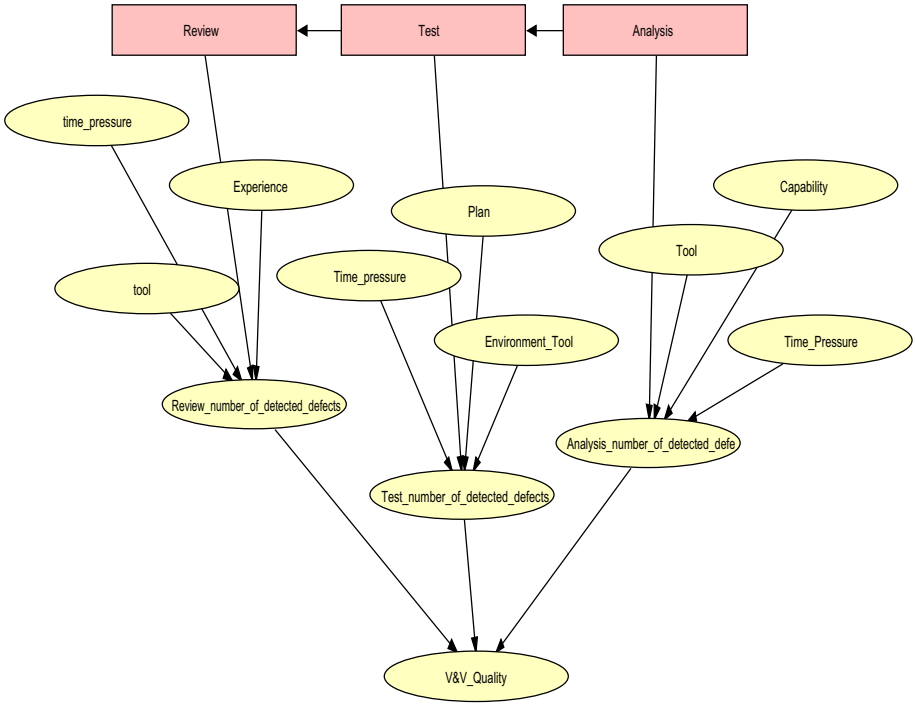

Fig. 12. V&V test case.

functions to convert numerical data to suitable forms for BBN inputs. On the other hand, in order to plot BBNs' outputs, we converted a BBN node's probabilities to a numerical value using weighted summation.

Formulae similar to Lin et al. (1997) are used in simulating software development process. Details of the simulator and test cases can be found in (Yu, 2002). Here we will only report a V&V test case. For testing V&V alone, the simulator can start from the following input:

Project size,
Number of initial defects,
Team's defect detection capability,
Maximum percentage of defects that can be detected,
Costs,
Thresholds,
Observed nodes.

In the test, we examined V&V activities, including analysis, testing, and review. Three capability levels, i.e., experts, average persons, and novices, are considered. The remaining defect number is calculated using the following formulae:

$$R_{t+1} = R_t\left(1 - \sum_{i=\text{Analysis}}^{\text{Review}} \sum_{j=\text{Expert}}^{\text{Novice}} D_{ij}\right) * \text{COV} \tag{4}$$

$$R_{t+1} = \text{Max}(R_{t+1}, r) \tag{5}$$

where R_t is the number of defects at time unit t, R_{t+1} is that at the time unit $t+1$. $\sum_{i=\text{Analysis}}^{\text{Review}} \sum_{j=\text{Expert}}^{\text{Novice}} D_{ij}$ represents the total defect detection ratio at the current time unit, and COV is a coefficient of variation, limited to ±5. However, defects may not be removed entirely; thus, r refers to the minimum number of residual defects.

We used the simulator to test the *Saturation Module* and *Ranking Module* together. The tested BBN diagram is shown in Fig. 12, which is expanded from the V&V node in the generic BBN (Fig. 4). In general, the effectiveness of analysis depends on analysts' capability; the effectiveness of review depends on reviewers' experience; while testing depends less on either.

Our test case assumed that the given analysts were novices, and testers and reviewers were of average capability. Since analysis relies heavily on analysts' capability; thus, the performance of analysis was soon saturated. The original performance is shown in Fig. 13, where review is better than testing and analysis. *Saturation Module* was invoked at week 7 and it identified that the analysis activity was not productive any more. The manager was then notified. The manager would terminate this saturated activity and reallocate the saved resources (i.e., budget, personnel, etc.) to the most productive V&V activity suggested by the *Ranking Module*. In this case, it was testing. The adjusted performance is shown in Fig. 14. It is obvious that adjusted testing performance improved after week 7. The curves of the remaining numbers of defects for both cases are shown in Fig. 15. It is noticeable that the case with dynamic resource allocation outperforms the one without it.

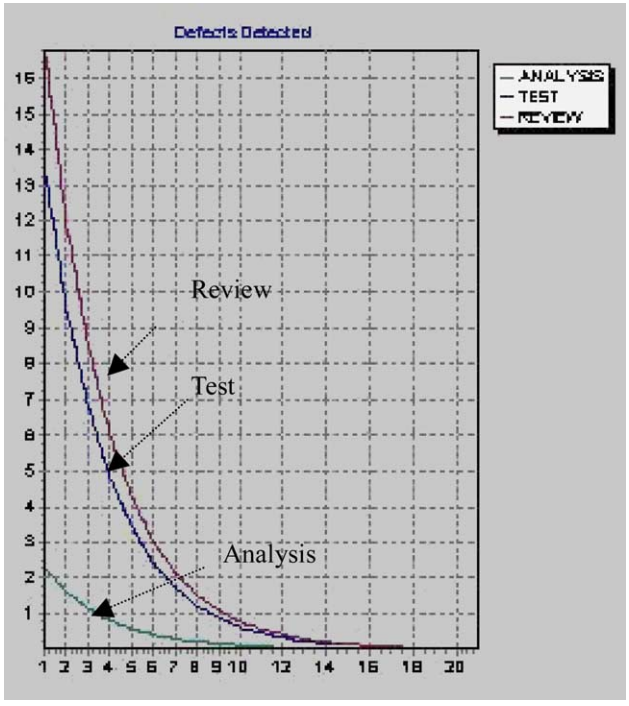

Fig. 13. Numbers of defects detected without risk management.

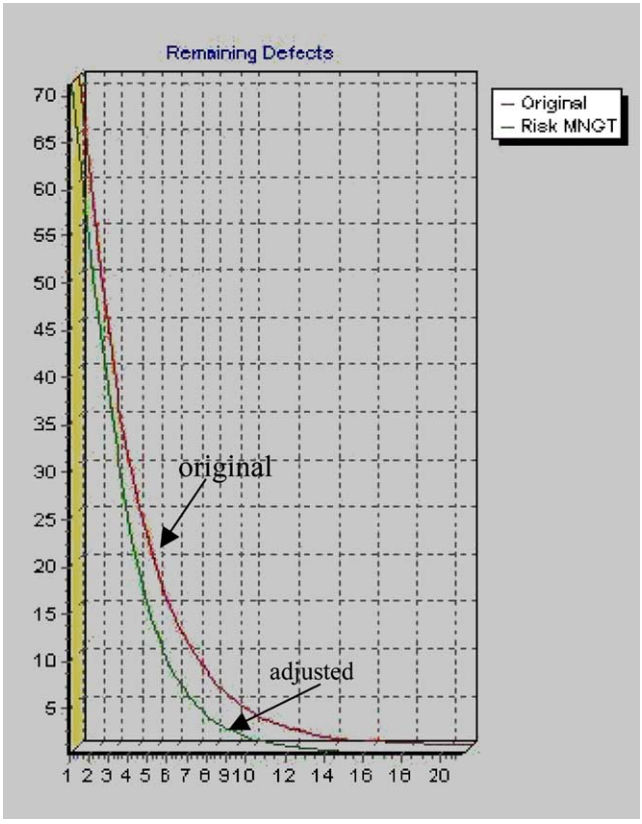

Fig. 15. Numbers of remaining defects with and without adjustment.

7. Conclusion

This paper proposed a scheme to incorporate BBNs in software project risk management. A mathematical model was first defined to provide insights into cost-effectiveness of resource utilization. We then developed a BBN-based procedure to continuously monitor project

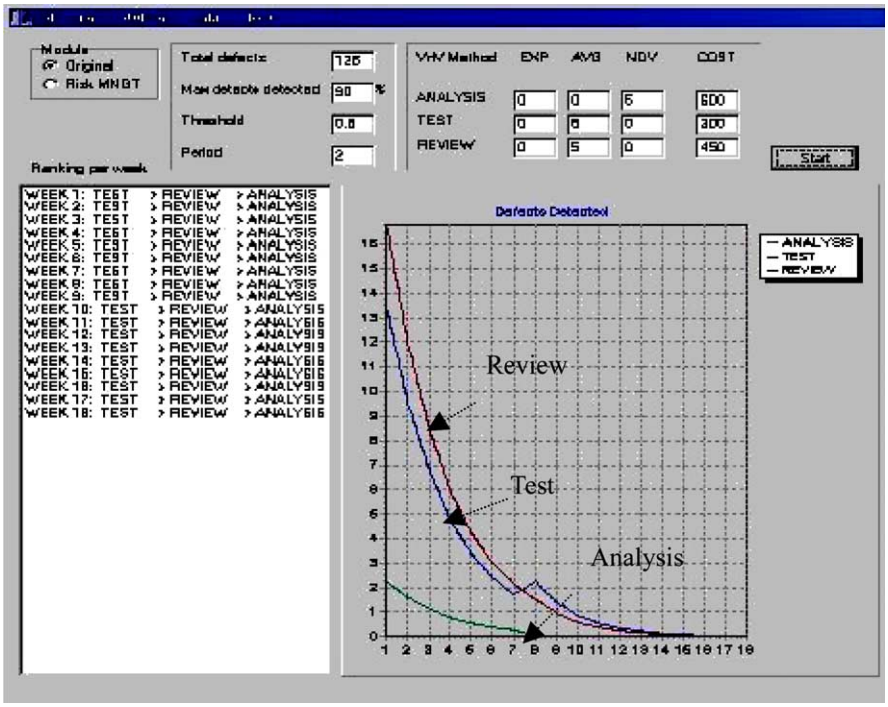

Fig. 14. Resource reallocated to testing at week 7.

progress, predict project risk, and assist managers to dynamically adjust resource allocation. The merits of BBNs are that they provide visible and repeatable decision-making support under uncertainties. Using both static analysis and dynamic test, we have shown that the direction of incorporating BBNs in risk management is promising. Further application of our approach to realistic on-going projects is desired.

Acknowledgements

The research has been partly supported by National Science Council and Nuclear Energy Council, Taiwan, under the grant number NSC90-2213-E-155-008 and NSC91-2623-7-155-001-NU.

References

Abdel-Hamid, I.K., Madnick, S.E., 1989. Lessons learned from modeling the dynamics of software development. Communications of the ACM 32 (12), 1426–1455.

Boehm, B.W., 1991. Software Risk Management: Principles and Practice. IEEE Software 8 (1), 32–41.

Bouissou, M., et al., 1997. Assessment of a safety-critical system including software: a Bayesian Belief Network for evidence. In: IEEE Proceedings of Annual Reliability and Maintainability Symposium, 1997, pp. 142–150.

Charette, R.N., 1989. Software Engineering Risk Analysis and Management. McGraw Hill, New York.

Fenton, N.E., Neil, M., 1999. Software metrics, failures and new directions. Journal of Systems and Software 47 (2–3), 149–157.

Heckerman, D., Wellman, M.P., 1995. Bayesian networks. Communication of ACM 38 (3), 23–30.

IEEE Std 1540, 2001. IEEE Standard for Software Life Cycle Processes-Risk Management.

Jensen, F.V., 1996. An Introduction to Bayesian Networks. Springer-Verlag, New York.

Karolak, D.W., 1996. Software Engineering Risk Management. IEEE Computer Society Press.

Kitchenham, B., Linkman, S., 1997. Estimates, uncertainty, and risk. IEEE Software 14 (3), 69–74.

Lin, C.Y. et al., 1997. Software engineering process simulation model (SEPS). Journal of Systems and Software 38 (3), 253–277.

Nuclear Engineering International, 1993. Sizewell B reactor protection reliability: Nuclear Electric presents its case, March 1993, pp. 28–33.

Oz, E., 1994. When professional standards are lax: the Confirm failure and its lessons. Communications of the ACM 37 (10), 29–43.

Yu, Y., 2002. BBN-based Project Risk Management. Master's thesis, Computer Science and Engineering Dept., Yuan-Ze University, Taiwan.

Ziv, H., Richardson, D.J., 1997. Constructing Bayesian-network models of software testing and maintenance uncertainties. In: Proceedings of International Conference on Software Maintenance, pp. 100–109.

Chin-Feng Fan has received the M.S. degree in computer science from Iowa State University, Ames, Iowa, and the Ph.D. Degree in computer science and engineering from Southern Methodist University, Dallas, Texas. She is currently an associate Professor of Computer Science and Engineering at Yuan-Ze University, Taiwan. Her research interests are mainly in the areas of software engineering and safety-critical computing.

Yuan-Chang Yu received the M.S. degree in computer science and engineering from Yuan-Ze University, Taiwan, in July 2002. His research interest is software engineering and network management.