ICS 2002

Workshop on Databases and Software Engineering

Title :
Software Project Risk Management Using
Bayesian Belief Nets (BBNs)

Abstract

Risk management has been considered as an effective technique to cope with inherent uncertainty associated with software development. Current software risk management mainly relies on subjective judgment of project managers. In such subjective approach, decision-making process is human-intensive and opaque. This paper presents a systematic approach to assist software risk management using BBN (Bayesian Belief Network). It converts the opaque decision-making process in risk management into a visible and repeatable process. Our method uses BBNs' causal dependency and prediction function to assist managers to perform risk analysis and resource adjustment. We have implemented the proposed algorithm in a simulation environment. Our results showed that this BBN-based approach could effectively enhance probabilities of project success.

Keywords : uncertainty, software project risk management, BBN (Bayesian Belief Network), risk profile.

Authors: Chin-Feng Fan and Yuan-Chang Yu
Affiliation: Computer Engineering & Science Dept., Yuan-Ze University, Taiwan
Address: 135 Far East Road, Chung-Li, Taiwan 320
E-mail: csfanc@saturn.yzu.edu.tw
Fax : (03)4638850
Contact: Chin-Feng Fan

Software Project Risk Management Using Bayesian Belief Nets (BBNs)*

Chin-Feng Fan, Yuan-Chang Yu
Computer Engineering & Science Dept., Yuan-Ze University, Taiwan

Abstract

Risk management has been considered as an effective technique to cope with inherent uncertainty associated with software development. Current software risk management mainly relies on subjective judgment of project managers. In such subjective approach, decision-making process is human-intensive and opaque. This paper presents a systematic approach to assist software risk management using BBN (Bayesian Belief Network). It converts the opaque decision-making process in risk management into a visible and repeatable process. Our method uses BBNs' causal dependency and prediction function to assist managers to perform risk analysis and resource adjustment. We have implemented the proposed algorithm in a simulation environment. Our results showed that this BBN-based approach could effectively enhance probabilities of project success.

Keywords : uncertainty, software project risk management, BBN (Bayesian Belief Network), risk profile.

* 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.

1. Introduction

There exist many uncertainties in software development processes and products; for instance, the uncertainties in estimating software size and quality, or in determining resource allocation and when to stop testing, etc. Current software engineering techniques cannot eliminate such uncertainties. Thus, risk management is critical. Dr. Kitchenham proposed that estimate uncertainty is "best managed across an organization's total portfolio"[10]. This implies that if resources can be shared or reallocated among several projects, then probabilities of project success can be enhanced. However, it may not be possible to manage several projects at the same time and share resources among them. Current risk management mainly relies on subjective judgment of project managers. Such subjective process is human-intensive and opaque. It will be desirable to assist risk management with more objective measures. This paper presents a systematic approach using BBNs (Bayesian Belief Networks) to assist risk management. A BBN provides improved clarity in defining and tracing causal dependencies. Moreover, it can perform calculation and prediction under uncertainties. Using BBN-based profile, our risk management algorithm can detect potential

risks and trace them to their root causes; our method can also warn the user of activities' saturation and assist the user to adjust resource allocation. In the following, we will first briefly introduce BBNs, then, present our proposed algorithm, followed by experimented case studies.

2. Bayesian Belief Networks (BBNs)

BBNs have attracted much recent attention in the area of decision support under uncertainties. BBNs' underlying theory (Bayesian probability) has been around for a long time; while the implementation algorithms [7] and software tools (eg., Hugin) [5] are available in these few years. Bayesian Belief Network [4,7] 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 casual relationships between the variables. A Conditional Probability Table (CPT) is associated with each node to denote such casual influence. The node representing a variable A with parent nodes representing variables B_1, B_2, ..B_n is assigned a CPT: P(A| B_1, B_2, ..B_n). CPT's are filled with a mixture of empirical/benchmarking data and subjective judgments. When the probabilities of nodes without parents are unknown, current tools usually assign evenly distributed probabilities to them. Once new evidence is obtained, evidence can be plugged in the graph to update the states of related nodes. Then, the calculation is propagated from parent nodes to child nodes and vice versa. The 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. Hadar Ziv[14] has used BBN in software testing and maintenance. Fenton[3] proposed that BBN would be the promising approach for representing and calculating complex software metrics.

3. BBN-based software project risk management

Our risk management scheme uses BBNs' clear causal dependency to identify risk source, and their estimation power to predict resource effectiveness. Our method provides a simple and visible analytical solution for risk management problems; it can be combined with other methods such as simulation to assist human experts. The system context of our method is shown in Fig.1, where the algorithm 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. A risk profile and a knowledge base of risks are the associated data structures. The proposed algorithm is given in Fig. 2. General speaking, software risk management process, as indicated in IEEE standard 1540 [6], should include the following tasks:

(a) Planning Risk Management
(b) Managing the Project Risk Profile
(c) Performing Risk Analysis
(d) Performing Risk Treatment
(e) Performing Risk Monitoring

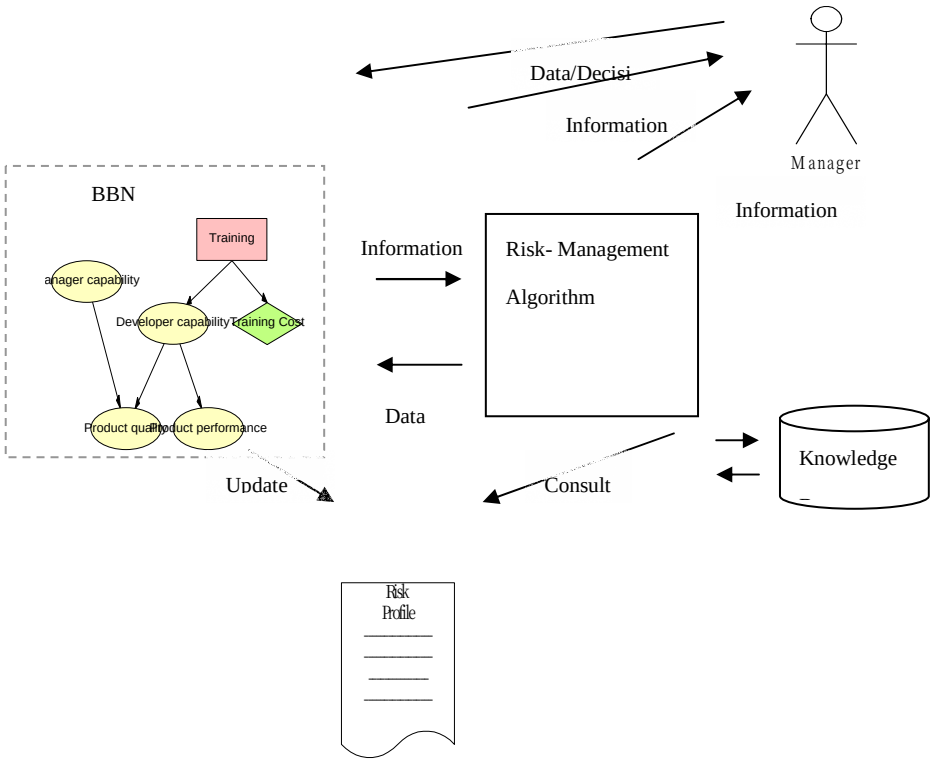

Fig.1. System context

Thus, we may categorize our steps similarly. The following is a simple explanation.

Steps 1 to 4 (Planning and initialization): Construct a BBN and initialize it.

A basic BBN including general factors [8,9] influencing project risks can be developed first as a template for our method. The first step will expand this basic BBN to further comprise factors specific to the examined project. The manager may use this expanded BBN to answer various "what-if" questions for resource planning. Assumptions may be used for initializing root nodes; then, the related expectation states may be saved in a separate file. Nodes to be monitored are also set. For example, if the assumption is that verifiers' capability is high, then the expected state is that the defects detection rate is high.

Step 5 (Risk Profile): Keep chronological records of the BBN's state probabilities and evidence inputs.

Once the project starts, a continuously monitoring loop will start. Whenever new evidence is obtained, evidence will be plugged into the BBN for analysis.

Steps 6-8 (risk monitoring and analysis)

To be able to perform risk analysis, various pre-assumptions should be correct in the first place. When evidence is gathered, it is plugged in the network to recalculate probabilities of its related ancestors or descendants. *Tracing Module* in our algorithm will be invoked when the average of previous N units of evidence conflict with original expectation. The *Tracing Module* shown in

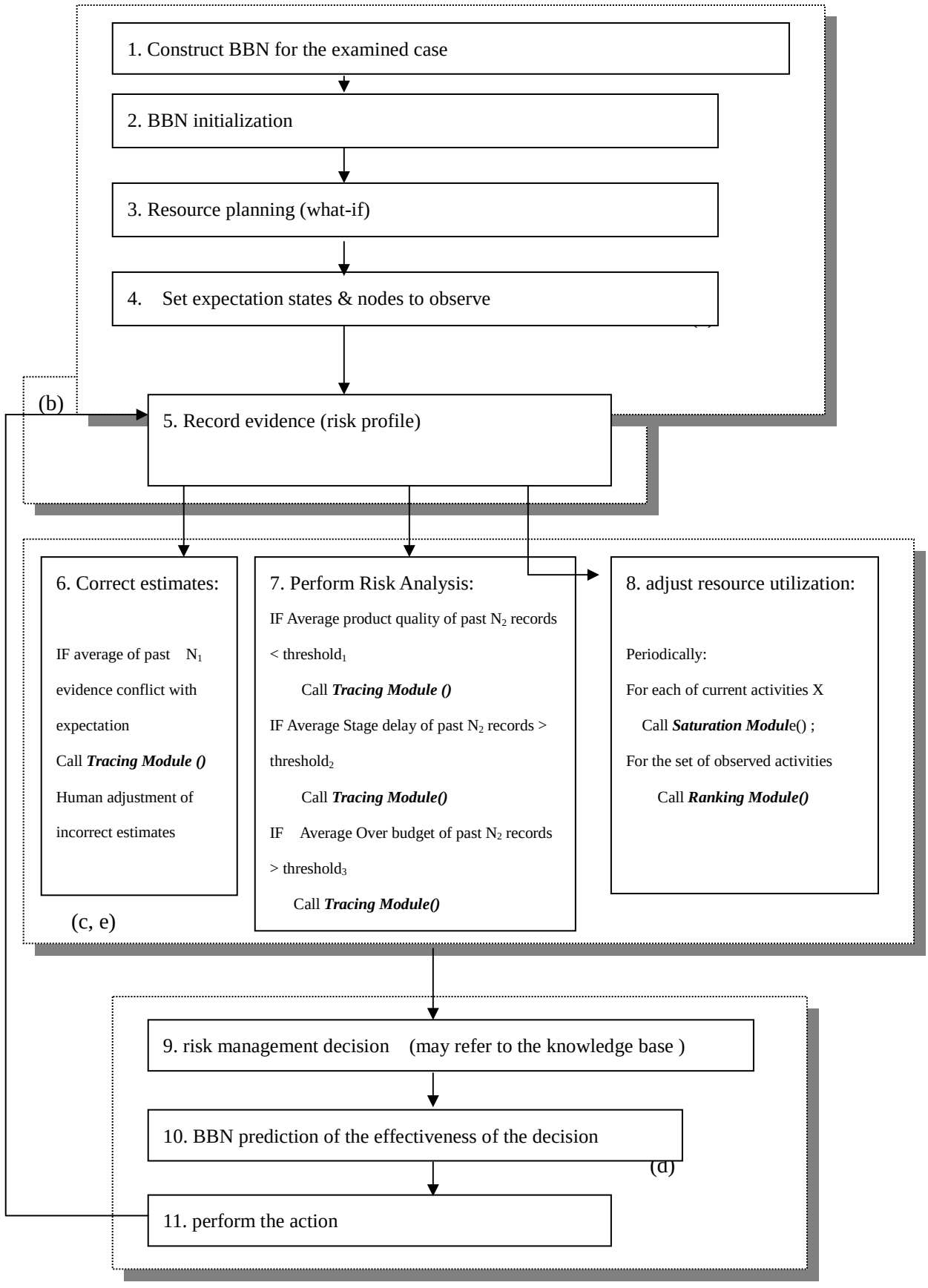

Fig.2. Risk management Procedure

Fig. 3 traces the current evidence back to its leading causes and interactively display them to the user, so that the user can identify the erroneous pre-assumptions and correct them.

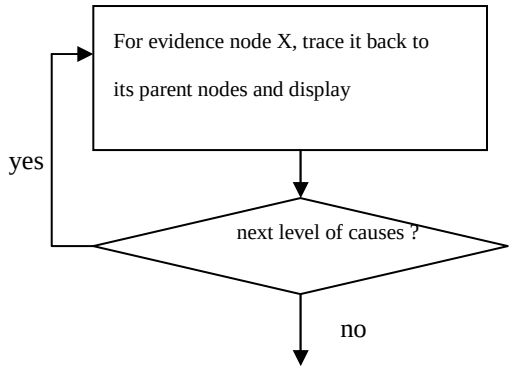

Fig 3. Tracing Module

Step 7 (Risk Analysis)

We analyze the average of the previous N risk profile data to identify whether there exist such risks as behind schedule, over budget, and poor quality. Assume that the basic BBN contains nodes *stage delay*, *over budget*, and *quality*. Then, these nodes are examined against the predefined thresholds. Once potential hazards are identified, our algorithm will invoke *Tracing Module* for the user to locate its major causes so as to assist the user to make risk treatment decision.

Step 8 (Resource adjustment)

Resources utilized in software activities may have their diminishing return points or saturation points; i.e., from a certain point on, the more resource spent, the less return it will yield. Thus, if project managers can realize such saturation points in time, they can reallocate these resources to alternative activities or save them for other more productive activities at later stages.

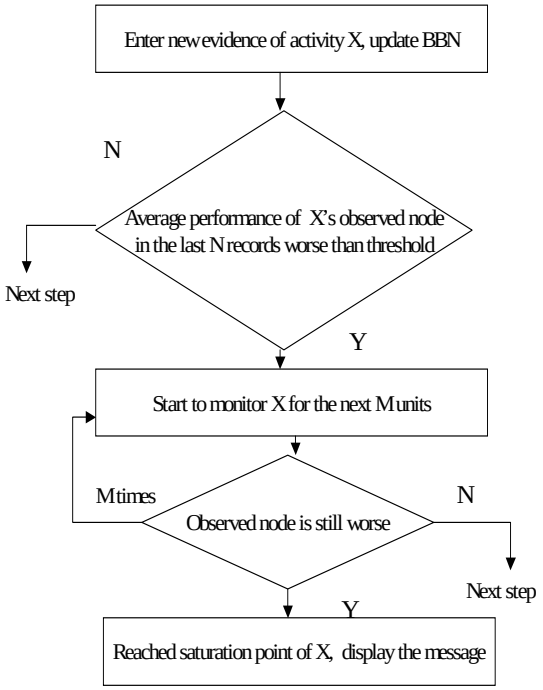

Fig. 4 Saturation Module

BBN risk profile can be used in revealing the falling cost-effectiveness of a certain activity, and identify its potential saturation points. Periodically, say daily or weekly, our algorithm invokes Saturation Module (Fig. 4) to check saturation points of the observed activities and ranking their cost-effectiveness (Fig. 5). In the Saturation Module, performance of observed activity X is examined; if its average value is less than the expected threshold, the saturation may be reached. To avoid possible transient situation, our algorithm continues to monitor the following M time units before reaching a final conclusion. Then the user will be informed.

Moreover, to better utilize resources,

Ranking Module (Fig. 5) is invoked periodically to compare and rank related activities. There are two different cases. If the set of activities have quantitative evidence in the risk profile; their effects per thousand dollars can be calculated straightforwardly; then, the ranking of these activities' cost-effectiveness are obtained and shown to the user. On the other hand, if there are no direct quantitative data for these activities, then, BBN should be used to estimate the effectiveness. Both the ranking of effectiveness and the ranking of their costs are sorted in separate sets. The best situation is that the ranking order of costs should be identical to that of their effectiveness; otherwise, resource adjustment may be needed to achieve better cost-effectiveness.

Steps 9 to 11 (risk treatment)

After the above analyses have been done, the manager may make decisions to treat possible risks. The knowledge base keeping rules of thumb for risk treatment can be consulted. BBN calculation can be used to estimate the effectiveness of decided risk treatment.

4. Implementation and Test Cases

We have implemented the above algorithm and tested it in a simulation environment. We have used a simple simulator to simulate the progress of the examined project and fed the resulting data to BBN for risk analysis.

4.1. Simulation Formulae

The proposed algorithm is implemented in ANSI C, using Hugin's APIs [5]; while the results are plotted by Borland C++. There exist some implementation issues. BBN variable states are discrete, yet input data to BBNs and various thresholds are numerical. Thus, conversion between numerical values and discrete probability states are needed. To

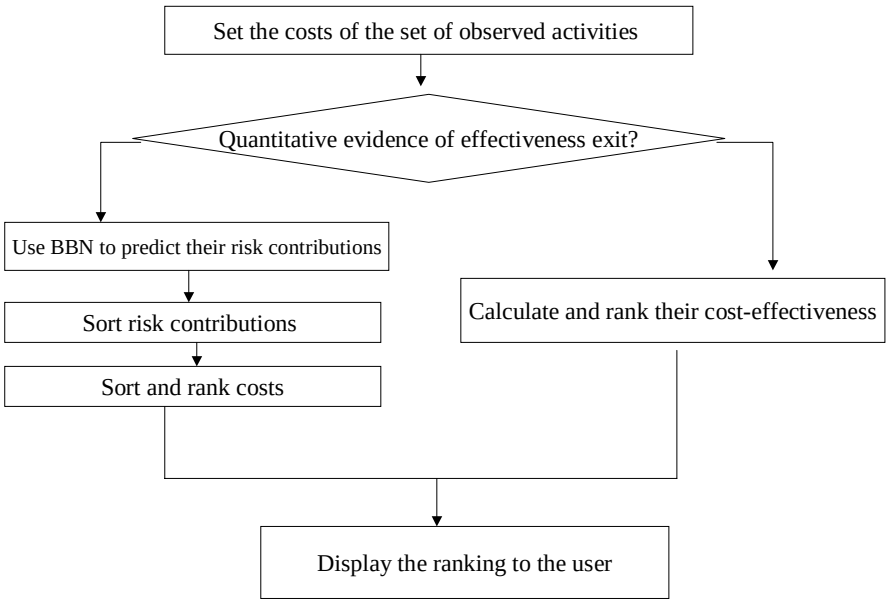

Fig. 5. Ranking Module

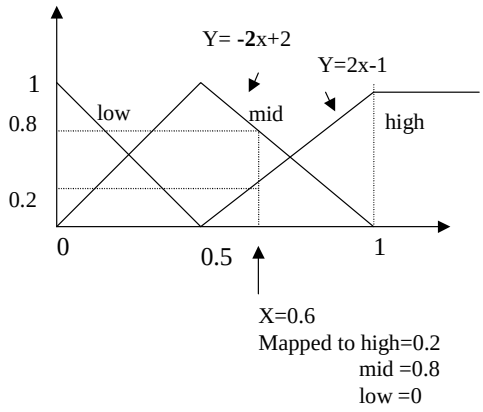

Fig. 6. Triangular function example

convert quantitative numbers to BBN variable state probabilities, we use fuzzy triangular functions. Fig. 6 is a simple three-state example, where the value 0.6 is mapped to probabilities of State high=0.2, mid=0.8, and low=0. In our test runs, we generally use three-state variables. On the other hand, to convert a BBN node's current state probabilities to a value, we use weighted summation. Currently we assign the continuous weights 1,2,3, etc. to the states low, mid, high, etc. For example, for a node with probability of State high=0.2, and mid=0.8, it will be quantified into (0.2X3 +0.8X2 + 0X 1)=2.2. Thus, for a three-state node, its state probabilities are converted into a number between 1 to 3.

We have constructed a process simulator to generate continuous project data as evidence for the BBN diagram. The inputs to the simulator include the following:

Project size (function points)

Schedule

Personnel (numbers of experts, average staff, and novices)

Productivity (function points/week)

Defect generation (numbers/week)

Defect detection rate for V&V activities (% of defects detected /week)

Observed nodes

Thresholds

The formulae we used in this simulator are given below [11]:

(1) ***Weekly team productivity =***

$$(\Sigma_{i=novice..expert} P_i * S_i) * C(S) * L(T) * COV$$

Where

P_i =weekly productivity of employees of type i

S_i= numbers of employees of type i

S= total numbers of employees

Communication overhead C(S)=1-t(S)

t(S)=1-{1.03exp(-0.02S)} [13]

Learning factor at time T =L(T) We used it when adding or changing staff after the project starts.

Coefficient of Variation COV=±10% It is generated by random numbers.

(2) ***numbers of defects produced weekly =***

$$(\Sigma_{i=novice..expert} S_i * D_i) * Pressure(delay) * COV$$

Where

D_i =Defect generation rate

Pressure factor *Pressure(delay)* is determined by current delay percentage.

(3) ***remaining defect numbers*** at time t

$$R_t =$$

$$\max (R_{t-1} * (1 - \text{average defect removal rate}), r)$$

Where r = total number of defects * maximum detection efficiency

4.2 Test case 1 : tracing causes and risk analysis

We used the BBN diagram in Fig.7 to test tracing and risk analysis modules. The test data are shown in Table 1. When our risk management scheme was used, at week 6, project delay caused Tracing Module to be invoked. The associated Hugin's monitor window is shown in Fig. 8; while the simulator called Hugin's APIs and got trace output shown in Fig. 9. It suggested that the potential causes for the delay might be staff numbers, capability, experience, and workload. Assume that the manager at this point decided to adjust and improve staff members' capability to be 10 experts, 10 average ones, and none novice. The performance with and with such adjustment is shown in Fig. 10. It is obvious that project *delay* and product *quality* were significantly improved.

Table 1 Simulation runs for case 1

Size (function points FP)	800
Schedule (weeks)	52
Personnel (expert, average, novice)	2, 10, 8
Productivity (FP/week)	1, 0.5, 0.2
Defect generation (# / week)	0.02, 0.05, 0.1
Observed nodes	Delay, quality, over budget
Learning factors *L(T)*	0.7 for 2 months when T <=10% 0.6 for 3 months when 10% < T <= 50% 0.4 for 4 months when T > 50% 1 otherwise
Pressure (delay)	0.7 when delay <0% 1 when delay=0 1.3 when 0%<delay <50% 1.5 when delay > 50%
Tracing module()	invoked at week 6 (Fig.8,9)
Staff adjusted (expert, average, novice)	10, 10, 0 at week 6 (Fig. 10) 10, 10,. 0 at week 35 (Fig.10)
Time adjusted	Expand to 100 weeks at week 6 (Fig. 11)

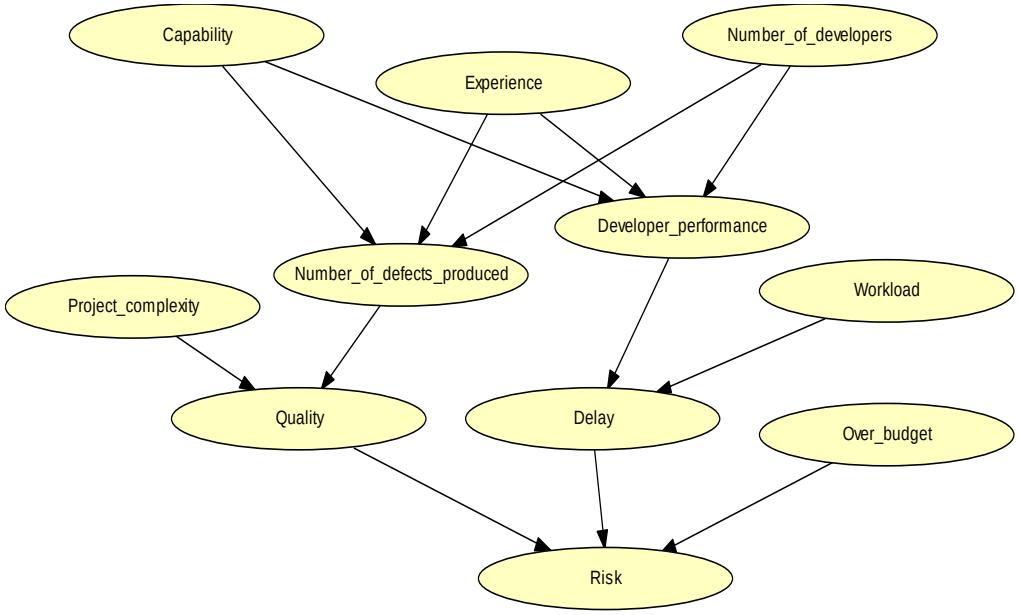

Fig. 7 Simplified BBN used in Case Study

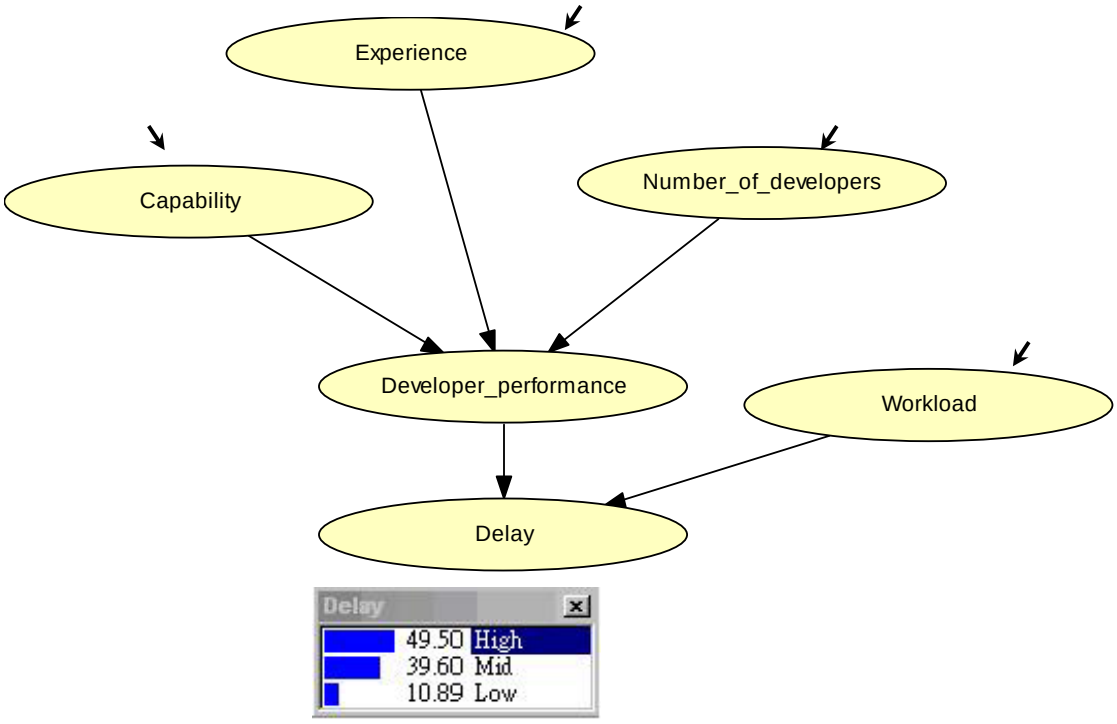

Fig. 8 Tracing back to causes

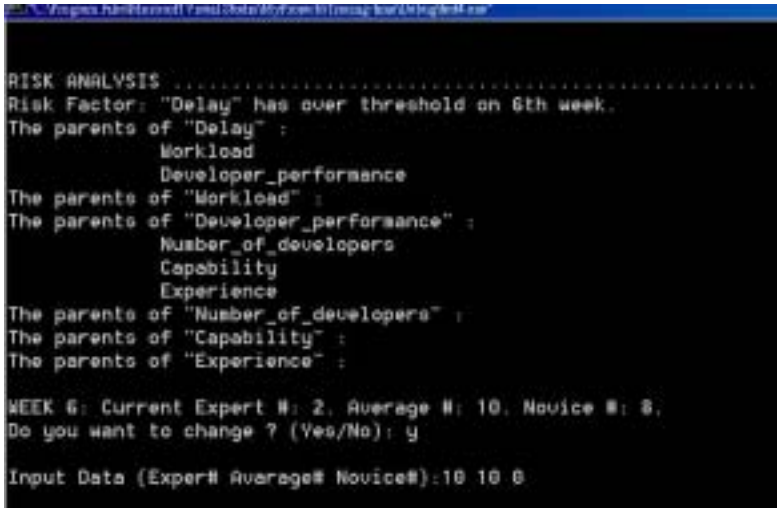

Fig. 9 Trace module output

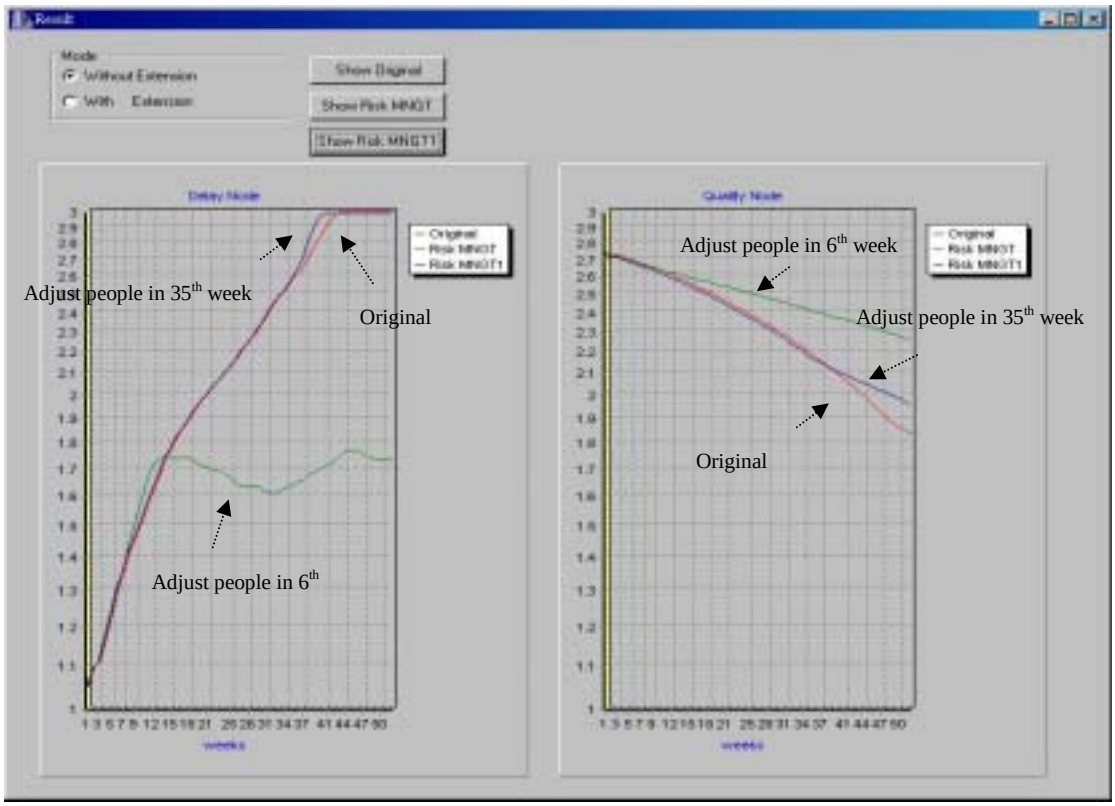

Fig.10 . Delay and Quality Nodes with and without risk management

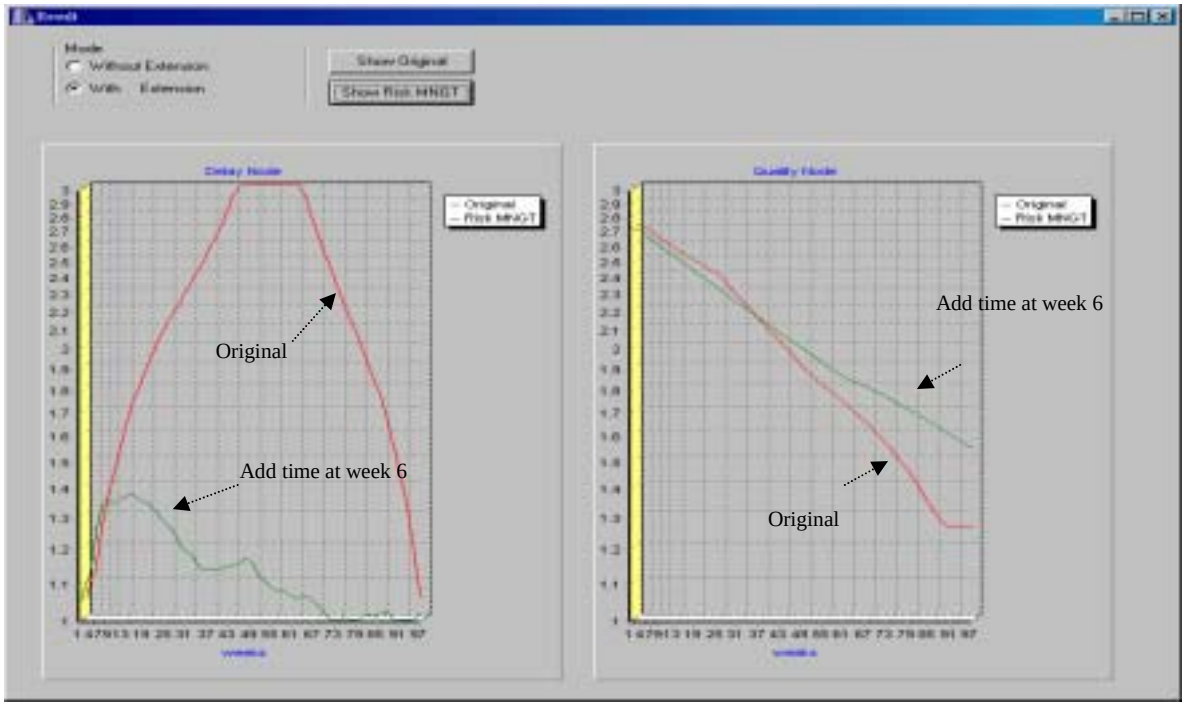

Fig. 11. Delay and Quality nodes with and without risk management (Adding time to 100 weeks)

Note that changing staff at week 6 incurred learning overhead, which is shown by the higher delay and lower quality in Fig. 10 immediately after the change. However, since it was still at the early development stage of the development, eventually the adjusted one outperformed the original one.

To compare the effect of changing staff at different stages, we also tested the personnel adjustment at a much later stage, say at 35th week in this case. The results are also shown in Fig. 10. As can be seen in the figure, there may have little improvement due to learning factors and the timing.

Another possibility for risk treatment at week 6 is to allocate extra time extension since *Trace module* indicated work load was not justified. Suppose that the completion time was adjusted to be 100 weeks. The resulting performance is shown in Fig. 11. Due to the fact that time pressure was lifted, the number of generated defects was reduced. Thus, delay and quality nodes were much better than the original ones.

In both cases, our algorithm could warn the manager of the potential risks at an early stage. Thus, proper measures could be taken to greatly improve the probability of project success.

5.2 Test case 2: Saturation point identification

We have tested *Saturation module* using a past problematic software project, the Sizewell B project [12]. This Britain's digital reactor protection system was developed by Westing House in early 90's with huge V&V effort. To ensure its safety, besides Westing House's V&V, the system was verified by the following

Independent Verification and Validation (IV&V): NNC Ltd.'s Independent Design Assessment, TA Consultancy Services' MALPAS (TCAS's IV&V), NE Technology's source to code comparison, as well as Rolls Royce and Associates' testing (RR&A's IV&V). The project spent 200 man-years development effort, and yet 50 man-year V&V effort, which did not find any significant defects. .

The BBN diagram of a Sizewell B-like case is shown in Fig.12. The data are given in Table 2. In the original execution, the BBN's product quality would have the numerical curve depicted in Fig. 13. If our algorithm was used, when software quality could not be improved, *Saturation Module* was activated and would inform the manager of this situation. Suppose that the manager decided to terminate the V&V activities (Fig. 14); then extra resources were saved. The comparison can be seen from Fig. 13 and 14, where product quality was the same, while with saturation identified, 14-week resources could be saved. With our risk management scheme, warning of potential risks for appropriate resource saving or adjustment is possible.

Table 2 Simulation run for test case 2

Size	1200
Number of defects	125
Maximum defection efficiency	95%
V&V schedule	20
Organization defect detection capability	25% ±5% per week
Without risk management	Fig. 13
With saturation point identified (at week 6)	Fig. 14

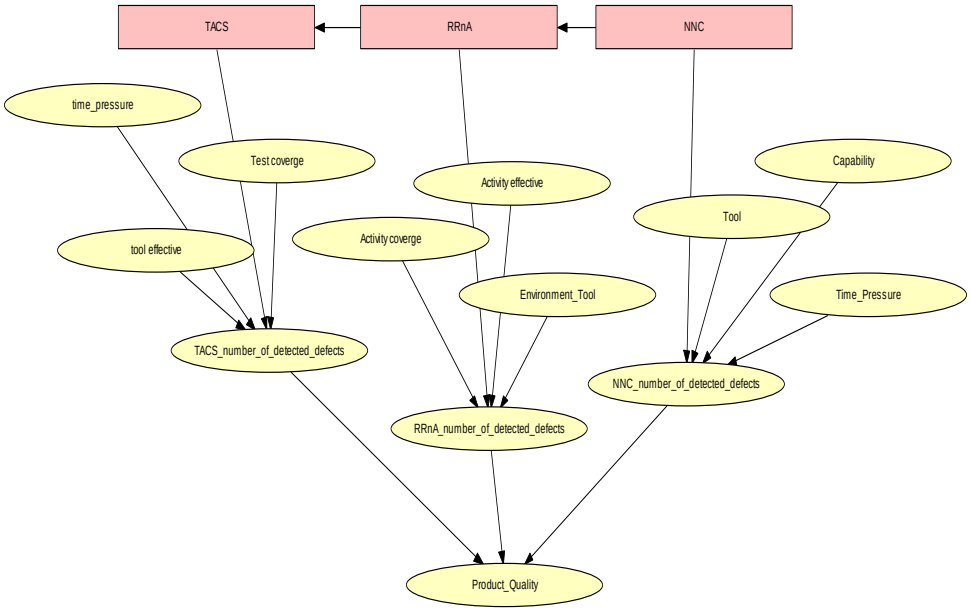

Fig. 12 IV&V Case (test Saturation module)

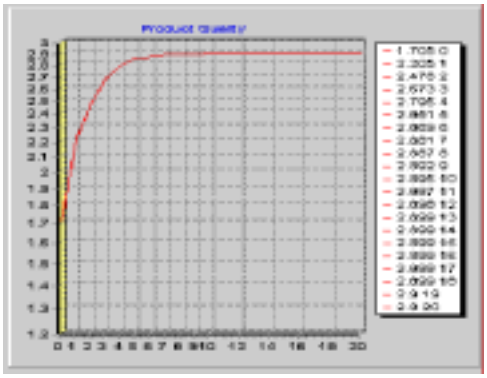

Fig. 13. Quality node at the original execution

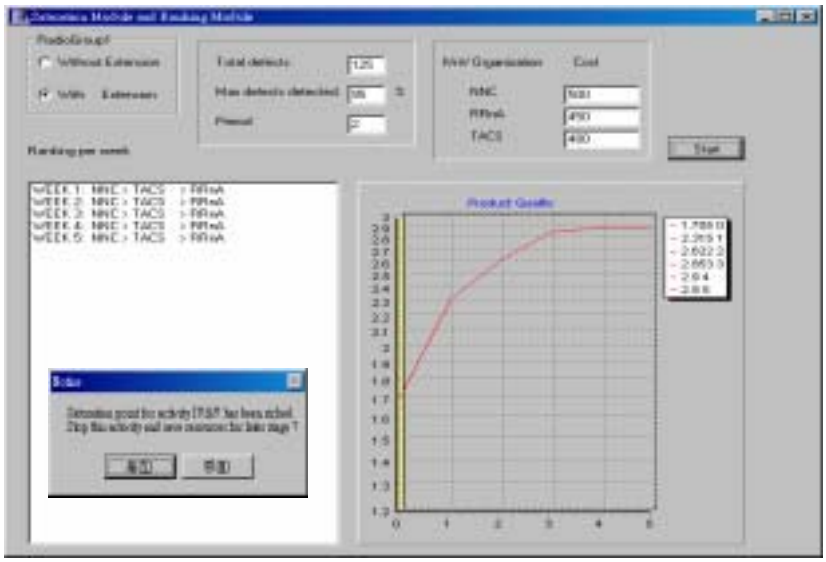

Fig .14 Quality node with Saturation identified and process terminate at week 6

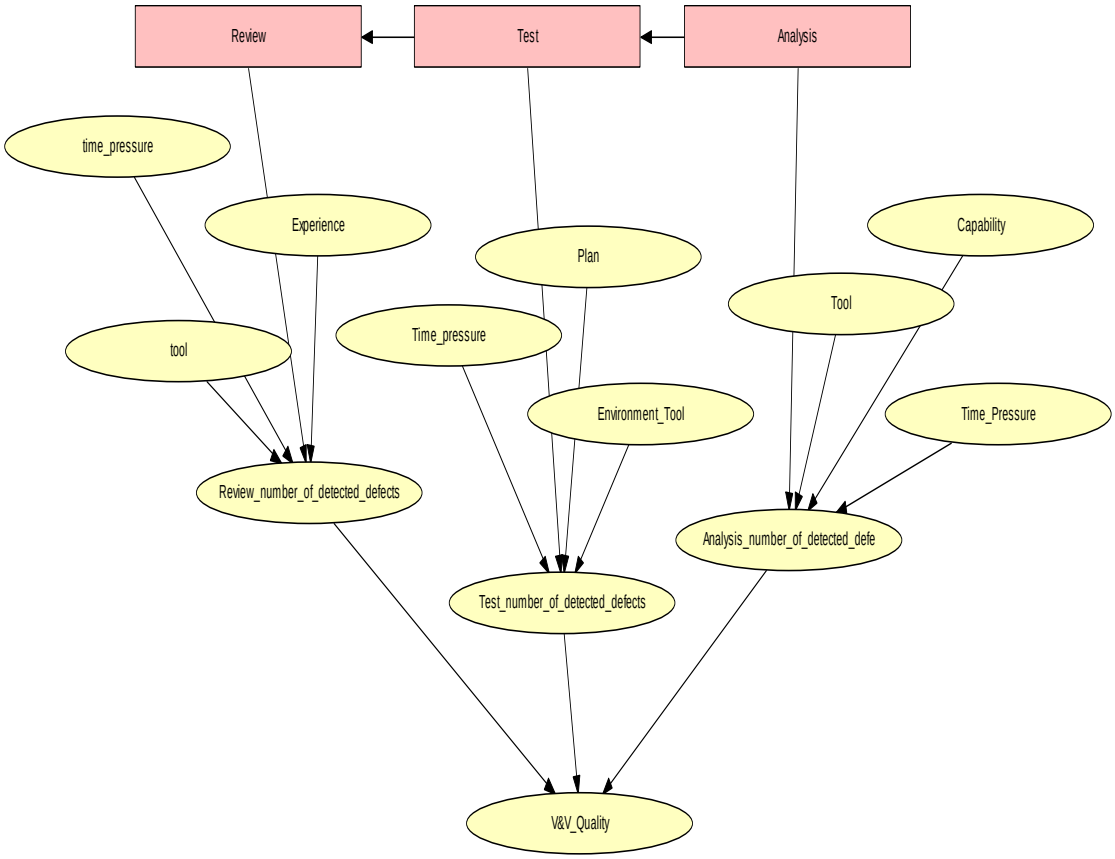

Fig. 15. Analysis, Review, and Testing (test saturation and ranking)

5.4. Test case 3: test Saturation and Ranking Modules together

The last test case we have tested the saturation module and ranking module working together. We assumed that three different activities: testing, review, and analysis were used. Periodically, the saturation and ranking algorithm would be invoked. The simplified BBN diagram is shown in Fig.15. In general, whether analysis is effective depends on analysts' capability; review depends on reviewers' experience; while testing depends less on capability and experience.

The third test case assumed that given analysts are novice or average; thus, very soon, the analysis' saturation point was reached. The input data is given in Table 3. Our algorithm would identify that the analysis activity was not productive any more and notify the manager. The manager might terminate this activity and reallocate the resources (person/time/budget) to most productive V&V activity suggested by the *Ranking Module*. The original performance is shown in Fig.16, where review outperformed testing and analysis. When our algorithm was used, *Saturation Module* was invoked at week 6 and found that novice analysis could not progress much. The *Ranking Module()* periodically listed the cost-effectiveness ranking among these three methods. For this particular set of data, testing is most cost-effective since

Table 3 Simulation run for test case 3

size	1200
defect	125
Maximum detection efficiency	90%
Saturation threshold	Average 0.5 in the past 3 weeks
Defect detection rate (expert, average, novice) per week	Analysis: 20%, 10%, 2% Review: 20%, 15%, 8% Testing: 15%, 12&, 10%
Team capability	Analysis: novice Review: average Testing: average
Cost (analysis, review, test)	600, 450, 300
Observed nodes	Number_of_detected_defects
With risk management	Fig. 16
With ranking and adjustment	Fig. 17 (terminated analysis at week 6, reallocated analysis personnel to testing) Fig. 18 (numbers of remaining defects)

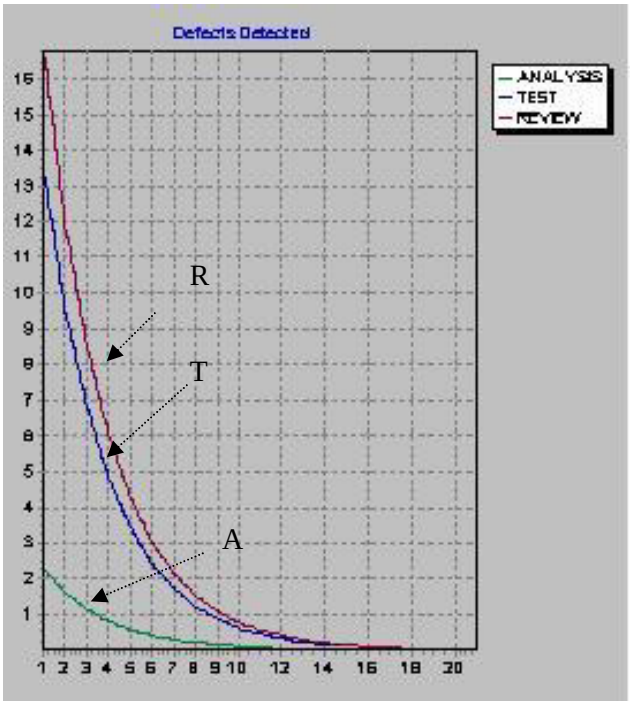

Fig.16. Performance of analysis, test, and review without risk management

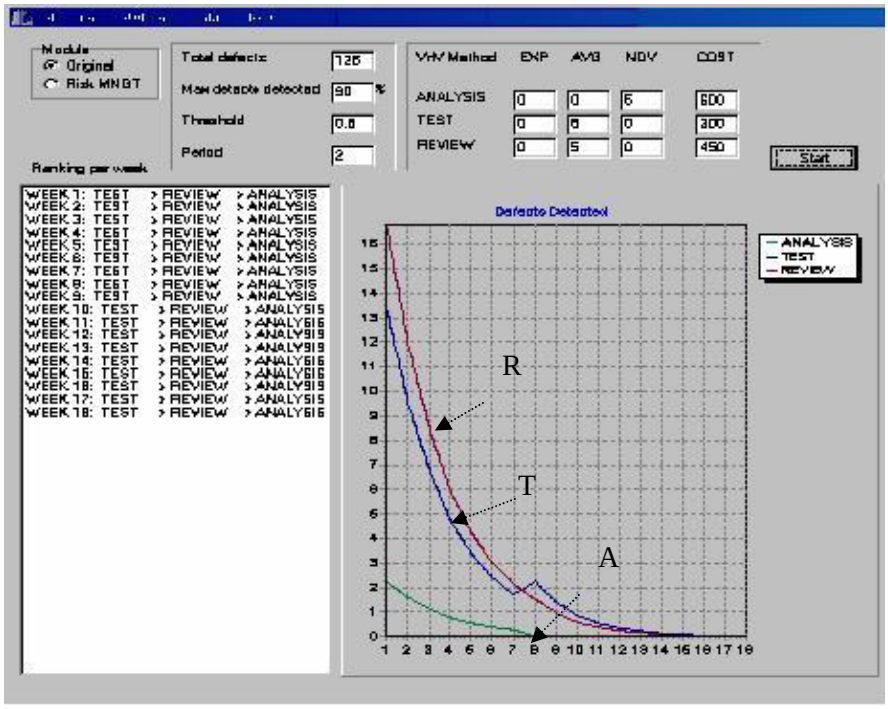

Fig.17. Resource reallocated to test at week 8

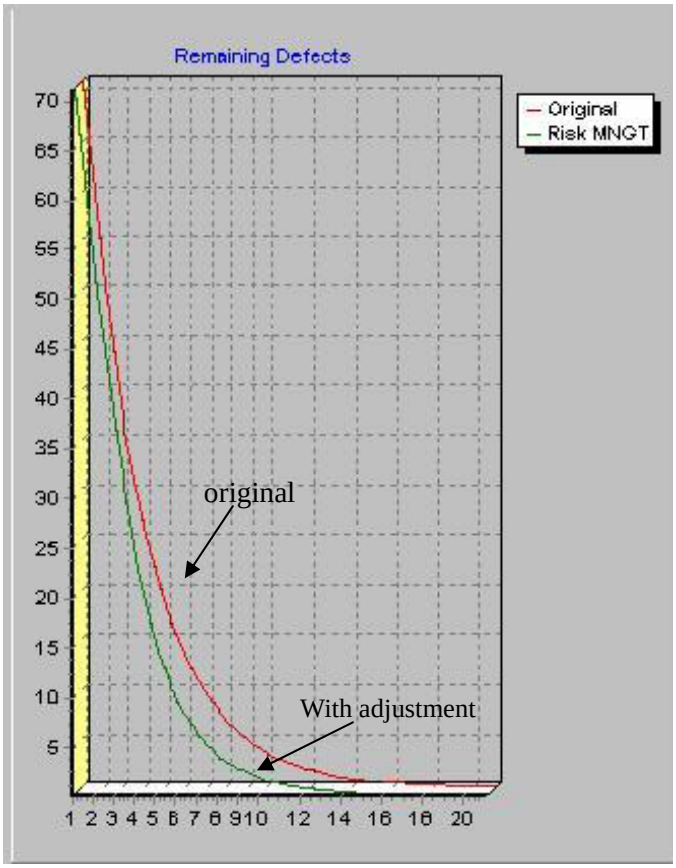

Fig.18 Remaining defects with and without resource adjustment

it was cheapest and depended less on personnel capability and experience than the other two methods. At week 8 the manager was advised that analysis' saturation had been reached. Suppose that the manager also took the suggestion from the Ranking Module. Thus, he terminated analysis activity and reallocated the analysis team to perform testing. This adjusted performance is shown in Fig.17. It is obvious that testing performance improved afterwards. To observe the overall effects of with or without risk management, we also plot the remaining numbers of defects for both cases in Fig. 18. It is noticeable that with dynamic resource allocation outperforms the one without it.

In all of the above cases, we have shown significant improvement can be achieved by using our BBN-based risk management algorithm. Our experiments demonstrated the feasibility and effectiveness of our method. Moreover, the rationales of manager's decision making is visible and repeatable when BBNs are used.

5. Conclusion

This paper proposed a BBN-based project risk management method. We utilized BBN's estimation power to predict potential risks or activity effects; also, we used BBN's causal dependencies to trace the potential causes of the risks. The BBN-based approach have the virtues in visibility and repeatability in the decision making process of software risk management. Using BBN, our method can continuously monitor and predict potential problems so as to assist the manager to perform risk treatment. We have implemented the proposed algorithm and tested it in a simulation environment. The experiment results show that the algorithm is effective. Further application of BBNs to other aspects of software project assessment seems to be promising.

References

[1] Marc Bouissou, et al., "Assessment of a Safety-Critical System Including Software: A Bayesian Belief Network for Evidence," 1999 IEEE Proceedings of Annual Reliability and Maintainability Symposium, pp. 142-150, 1999.

[2] Fred Brooks, "*No Silver Bullet: Essence and Accidents of Software Engineering*," *IEEE Computer*, April 1987, pp. 10-19.

[3] Norman E. Fenton and Martin Neil, "Software metrics, failures and new directions," *The Journal of Systems and Software*, 47, pp. 149-157, 1999.

[4] David Heckerman and Michael P. Wellman, "Bayesian Networks," Vol.38, No. 3 *Communication of ACM*, pp.23-30, March 1995.

[5] Hugin, A. S. , 1989, www.hugin.dk.

[6] IEEE Standard for Software Life Cycle Processes-Risk Management, IIEEE Std 1540-2001.

[7] Finn V. Jensen, *An Introduction to Bayesian Networks*, Springer1996.

[8] James J. Jiang and Gary Klein, "*Risks to different aspects of system success*," Information & management 36 (1999), Elsevier Science, pp. 263-272, 1999.

[9] Dale Walter Karolak, *Software Engineering Risk Management,* 1996 IEEE Computer Society Press.

[10] Barbara Kitchenham and Stephen Linkman, "Estimates, Uncertainty, and Risk," *IEEE Software*, May 1997, pp. 69-74.

[11] Chi Y. Lin, “Software-Engineering Process Simulation Model (SEPS), “ J. of System Software, vo. 38, pp. 253-277, 1997.

[12] “Sizewell B reactor protection reliability: Nuclear Electric presents its case,” *Nuclear Engineering International*, pp. 28-33, March 1993.

[13] Tausworthe, R. C., “Information Models of Software productivity: Limits on Productivity Growth,” *J. System Software*, 21, pp. 185-201, 1992

[14] Hadar Ziv, Debra J. Richardson, *Constructing Bayesian-network Models of Software Artifact Uncertainties*, PhD thesis University of California, Irvine, June 1997.