Construction of BBN Diagrams for Software Process Tailoring

Wan-Hui Tseng and Chin-Feng Fan
Computer Science and Engineering Dept., Yuan-Ze University.
csfanc@saturn.yzu.edu.tw

Abstract -*Tailoring industrial standards aims to reduce costs and/or improve quality for a particular organization or project. This paper proposes using Bayesian Belief Network (BBN) analysis to support tailoring decision-making under uncertainties. However, there are two major problems associated with the objectivity of BBNs; that is, the construction of the causal inference diagrams and the assignment of probabilities of their dependency relations. We have developed a method to solve the first problem. In general, the relations among different activities, resources, and products addressed in software standards can be expressed more directly in UML diagrams than in BBN's; such relations include association, aggregation, or inheritance relations. We have developed a scheme to construct BBNs from given UML modeling of software standards for process tailoring purpose. The proposed approach integrating UML and BBNs can also be used to assist decision-making in other software project management activities, such as planning and risk management.*

Keywords : software process tailoring, UML, BBN

1. Introduction

Different software organizations and projects have different quality requirements and resource constraints. Thus, tailoring of software standards is required to reduce costs and/or improve quality for a particular organization or project.

This paper proposes using Bayesian Belief Network (BBN) analysis to support tailoring decision-making under uncertainties. BBNs have been widely used to support decision-making under uncertainties. BBNs provide visible causal dependency diagrams, mathematical computation of probabilities, and support the visibility and repetition of the decision-making process. However, there are two major problems associated with the objectivity of BBNs; that is, the construction of the causal inference diagrams and the assignment of probabilities of their dependency relations. We have developed a method to solve the first problem. Software standards involve resources, processes, and products, which have various kinds of relations such as inheritance (is-a), aggregation (part-whole), input/output, temporal, and associations relations. They all can be modeled by UML diagrams more directly than by BBN's. Thus, we propose to first model the requirements of the concerned standards in UML, and then identified the explicit and implicit cause-consequence relationships in these UML diagrams to construct related BBNs. This way the factors considered in BBN diagrams have more objective bases than those if we directly model software standards using BBNs. Results of BBN analysis for what-if questions can then be used for tailoring decision-making.

The standards used in this research are IEEE/EIA 12207[3] and ISO/IEC 15504[4]. Section 2 provides a brief background introduction Section 3 shows the UML modeling of general standards such as IEEE/EIA 12207 and ISO/IEC 15504. Section 4 presents our approach to objective construction of BBNs from UML diagrams. Section 5 is the inferred BBNs and a sample tailoring case. Finally, a conclusion is given.

2. Related Background

Bayesian Belief Network [1] is an acyclic graph used for modeling and reasoning with uncertainties. Each node in a BBN represents a random variable, whose state is usually expressed in discrete numbers or ranges. Each edge in the graph represents the causal influence between connected nodes. A Conditional Probability Table (CPT) is associated with each node to denote such causal influence. CPT's are filled by experts or inferred from statistical data. Once new evidence is obtained, it can be plugged in the graph to update the states of the related nodes. The calculation is propagated 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 nodes. The former are shown by rectangles; the latter, representing cost or profit functions, are depicted by diamonds. Figure 1 is a sample BBN example.

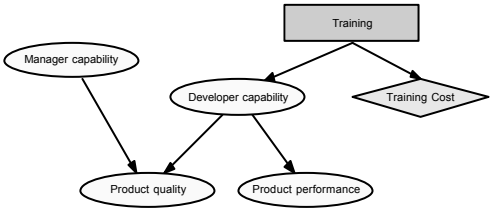


Figure 1. BBN Example

Current tailoring techniques are mostly subjective depending on domain experts. IEEE/EIA

12207 Annex A and B provide tailoring guidelines. The concerned factors include: project characteristics, development model, contract requirements, time and budget constraints, legal and safety factors. However, the standard does not provide technical schemes to use these factors in tailoring.

3. UML Modeling of Software Standards

UML is widely used for modeling at the object-oriented analysis and design stages. It provides both the data and behavior modeling of software systems. Software standards address the desired practice, resources, and products of software life cycles. The involved entities have such relationships as is-a, part-whole, input-output, temporal, and other associations. For example, the each individual process is a process (is-a); each process contains several activities (part-whole); design documents are inputs and source code is the output for the implementation stage (input-output); etc. UML diagrams can better model these entities and relations directly than BBNs. Besides, using UML, we can construct the modeling in a hierarchical way. That is the construction can start from the most abstract level to the details using different kinds of UML diagrams.

We first model the general software processes, products, and resources based on IEEE/EIA 12207. For tailoring purpose, we also consider the developer's process capability levels as indicated in ISO/IEC 15504.

Process, product, and resources are the three major groups of concerned entities. Moreover, software project management also deals with *pragmatic aspects* such as personnel issues and particular project's characteristics. Here we isolate personnel from resources. Therefore, we group the major concerned factors into the following six categories: (1) process, (2) product (3) resources (4) process capability levels, (5) personnel, and (6) project characteristics. Figure 2 shows details of these categories using UML package diagrams. Then these six categories are expanded into details using appropriate UML diagrams. They are mentioned below.

For activities, we use an activity diagram to show the temporal, input-output, and quality factors among different activities. Since whether IV&V (Independent Verification and Validation) should be used is an issue in our case study, we also add IV&V to our consideration. The resulting UML activity diagram is shown in Figure 3. Moreover, the structural relations of development and supporting processes can be represented using class diagrams as those shown in Figure 4.

For products, we consider different documents, their related quality and relations.

For resources, we consider organization support, budget, schedule, and tools. These are shown in Figure 5.

For capability levels, we refer to ISO/IEC 15504 and consider capability level, process attributes under each level, management practice of each process attribute, base practice of each management practice, and work practice of each management practice. This is shown in Figure 6.

For pragmatic concerns, we consider personnel types, experience, capability, team performance factor as well as project characteristics. These are shown in Figure 7.

As shown above, the desired software practice and related products and resources addressed in a general standard can be successfully modeled by UML diagrams in a hierarchical way.

4. Construction of BBNs Based on UML Diagrams

Once the UML modeling has been built, BBNs can be derived. The potential BBN nodes and their cause consequence relations need to be inferred or extracted from given UML diagrams. The causal relations we considered include the following:

- logical relations : logical relations between attributes and between classes/objects
- input/output relations: input affects processing, and processing affects output
- part-whole relations: parts affects the whole
- temporal relations: predecessors affect successors

We first define by the following UML notations:

1. Attribute $C.att = \{C.att_1, C.att_2, ..., C.att_n\}$: where *C.att* is the attribute set of class C.
2. Generalization relation R_{Gen}*=(C,CGen)*: where *C* is a superclass, *CGen* is the set of its subclasses
3. Aggregation relation R_{Agg}*=(C, CAgg)* : where *CAgg* is the set of parts of class *C*.
4. Association relation R_{Ass}*=(C,CAss)* : where classes *CAss* and *C* have association relation.
5. Dependency relation R_{Dep}*=(CDep, CDep1)* : where class *CDep* depends on class *CDep1*.
6. *C. implied* refers to one of C's implicit properties, not represented by *C*'s attributes. Similarly, *State.implied* refers to one of *State*'s implicit properties related factors.

For BBNs, we define the *acyclic graph= (N, E, r)*, where *N* is the sets of nodes, *E* is the set of edges, and *r* is the causal relations among nodes. If $e \in E \wedge r(e) = (a, b) \in N \times N$, then *a* is called the parent node, and *b* the child node of the causal relation.

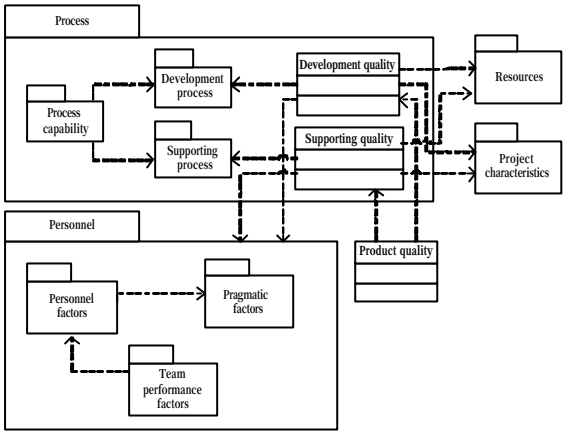


Figure 2. Major entity categories

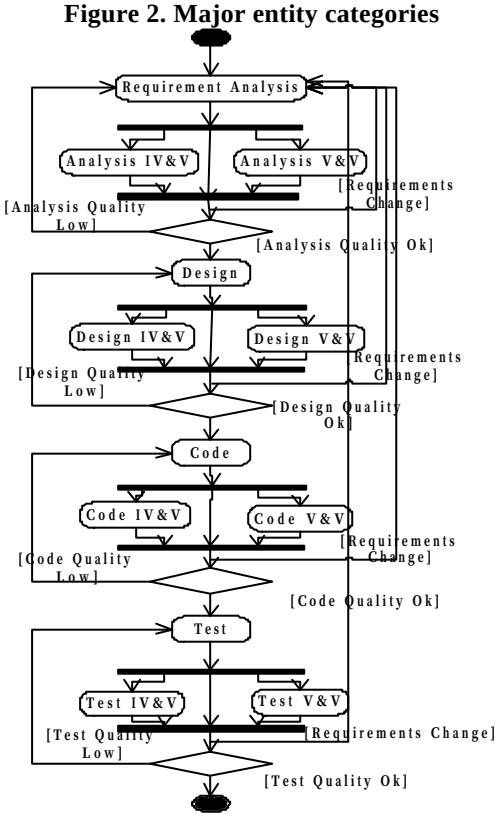
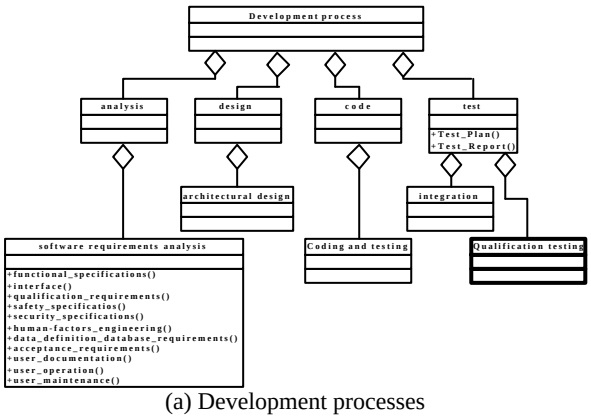
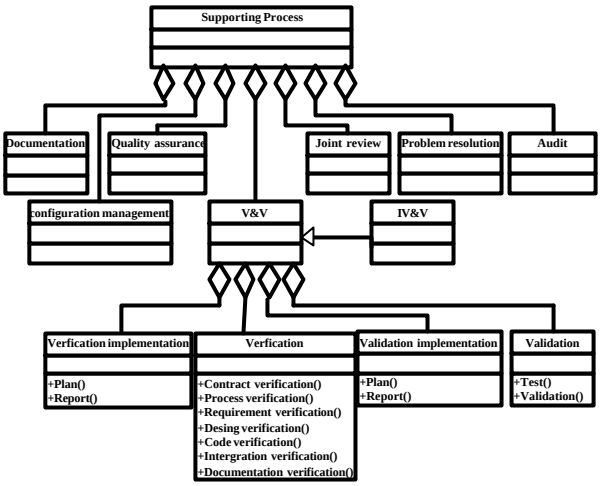


Figure 3. Activities and their relations



(a) Development processes



(b) Supporting processes

Figure 4. Composition relations of activities

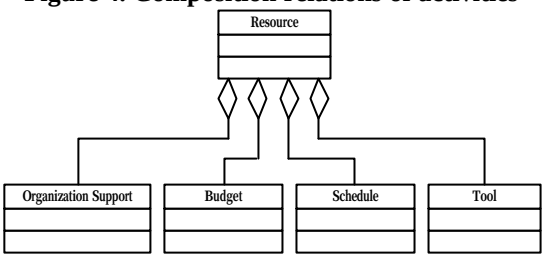


Figure 5. Resource factors

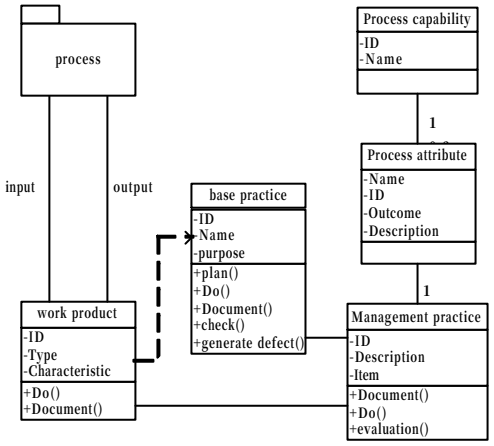


Figure 6. Capability related concerns

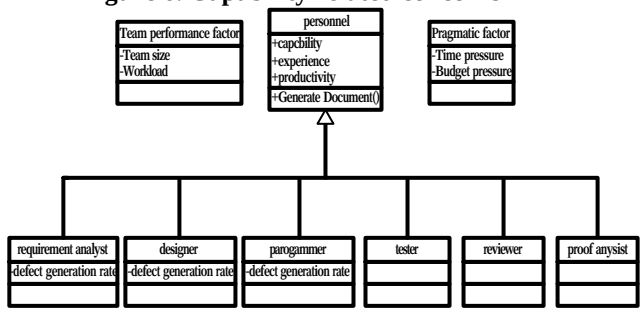


Figure 7. Personnel types and personnel quality

Our inference method can be categorized into structural relations and behavior relations. For structural relations, the following rules can be used:

Rule 1 derives potential causal relations between attributes of a class. However, the class may not explicitly model some of the desired attributes.

For example, the type of a project may affect the project complexity; while the complexity may not be modeled. Then it will be added as implied factors in the resulting BBN causal diagrams. Figure 8 shows the inferred BBN causal relations from UML class attributes and implied attributes. Similarly, in inheritance hierarchy, there may exist dependencies among inherited attributes and new attributes of child classes.

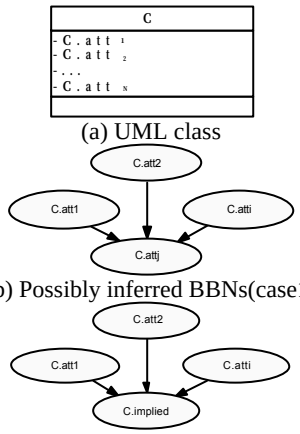
Rule 1: attribute

If $C.att = \{C.att_i \mid 1 \le i \le n\}$ in UML,

Then there potentially exits an edge *e* in BBN in the following cases:

Case 1: $e = (C.att_i, C.att_j)$ *for* $1 \le i, j \le n$

Case 2: $e = (C.att_i, C.implied)$



(a) UML class

(b) Possibly inferred BBNs(case1)

(c) Possibly inferred BBNs(case2)

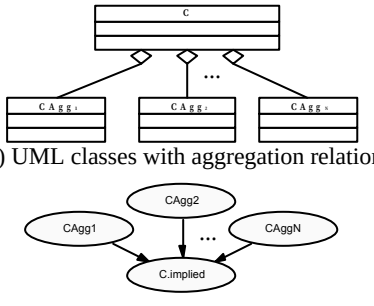
Figure 8. Causal Relations between attributes

Rule 2: Aggregation relations among classes

If R_{Agg}=*(C, CAgg)* where $CAgg = \{CAgg_i \mid 1 \le i \le n\}$ in UML,

Then, there potentially exits an edge *e* in BBN,

where $e = (CAgg_i, C.implied)$ for $1 \le i \le n$



(a) UML classes with aggregation relation

(b) Converted BBN

Figure 9. Part-of relations

Rule 2 deals with "part-of" relations between classes. The aggregation relations show in UML class diagrams can be converted to BBNs' causal relations: the parts depend on the whole. This is shown in Figure 9.

UML class diagrams also express various kinds of relations between two classes. The UML association relations and dependency relations are the most obvious ones that may be directly modeled in BBNs. The potential derivations from UML's association and dependency relations to BBNs' causal relations are shown in Figures 10 and 11, respectively.

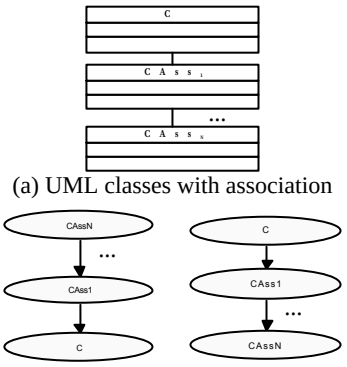
For behavior modeling, temporal and I/O relations can be derived from UML state transition diagrams or statcharts. Two states in the state transition diagrams may have temporal, input/output or data relations. The predecessors may affect the successors; the input may affect the output. For example, the quality of design state's output may affect the quality of coding process. The possible state transitions and the inferred BBN are shown in Figure 12. Note that the original state diagrams may have cycles; however, BBNs cannot have cycles. Similarly, BBNs may be generated from activity diagrams.

The above rules can assist us to construct BBNs systematically. If the given UML diagrams faithfully and completely model the examined standards, it is highly possible that the derived BBNs can also faithfully express the standards. Then tailoring of software standards can then be proceeded on the resulting BBNs.

Rule 3: Association among classes

If R_{Ass}=*(C,CAss)*, $CAss = \{CAss_i \mid 1 \le i \le n\}$ in UML, Then there potentially exits an edge *e* in BBN, where

$e = (CAss_i, C)$ or $e = (C, CAss_i)$ for $1 \le i \le n$



(a) UML classes with association

(b) Possible BBNs

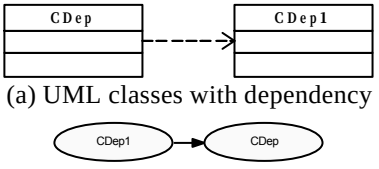
Figure 10. Association relations

Rule 4: Dependency relations among classes

If R_{Dep}=*(CDep, CDep1)* in NUL,

Then there exits an edge *e* in BBN,

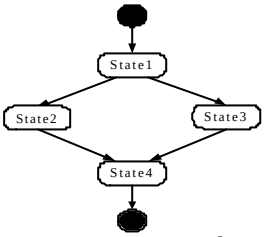
where $e = (CDep1, CDep)$



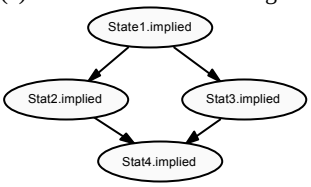
(a) UML classes with dependency

(b) Converted BBN

Figure 11. Dependency relation



(a) UML state transition diagram



(b) Potential BBN

Figure 12. Temporal and I/O relations

5. BBNs for Process Tailoring

In this section, we will present the constructed BBNs using our approach and a tailoring sample.

5.1. BBNs Derived from UML diagrams

Using the rules presented above, we may infer potential BBN causal relations from the UML diagrams in Section 3 and generate related BBNs. The generated BBNs represent process, product, and resource requirements as well as pragmatic concerns of the examined standards for tailoring. For example, the developer's capability factors, based on ISO/IEC 15504, depicted in Figure 6 may be converted to the BBN figure in Figure 13. Resource concerns shown in Figure 5 may be transformed to the BBN in Figure 14; the personnel attributes in Figure 7 may generate the BBN in Figure 15.

In general, the BBNs representing the quality of each development phase may be inferred from the UML diagrams in Sec. 3 and may look like the one shown in Figure 16.

After the BBNs have been constructed, the CPT (conditional probability tables) of each node needs to be assigned. In this research, we only provided a systematic way to construct BBN influence diagrams. The associated probabilities yet have to be assigned by experts. Multiple experts can be consulted and then Delphi approach can be used to get the average of majority opinions and trim the extreme ones.

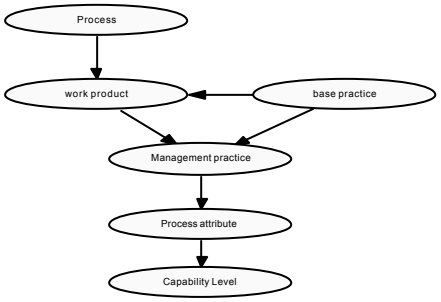


Figure 13. Capability Level BBN

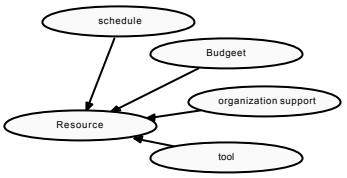


Figure 14. Resource BBN

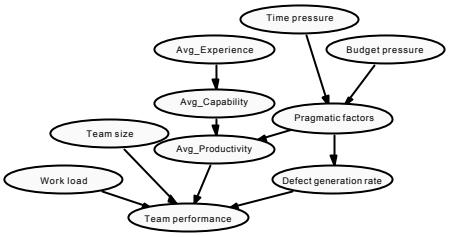


Figure 15. Personnel BBN

5.2. Process Tailoring using BBNs

Process tailoring can be performed on the derived BBNs. Different scenarios can be analyzed using different data. We have performed experiments with the following tailoring issues using Hugin, a popular tool [2]:

1. Whether to perform IV&V at design and coding stages
2. Whether to prepare test plans for module testing
3. Whether to add people during different stages
4. How to choose among testing, review, and analysis under different manpower, personnel experience, and capability levels.

Scenarios with different input quality, resource constraints, project types, V&V/IV&V team capabilities, and timing factors are considered. The BBN results of these sample runs are helpful for tailoring decisions. Details can be found in [5].

To demonstrate how the inferred BBNs can be used for process tailoring, we present one case here. We used the simplified portion of Figure 16 to consider whether IV&V should be performed. The considered BBN is shown in Figure 17. Appropriate CPT's are given using domain experts' knowledge. Assuming that the internal V&V quality is good. The Hugin results for the cases without performing IV&V, is shown in Figure 18. The one using IV&V is shown in 19. The latter case shows slight improvement in product quality. According to this result, IV&V may not be needed in the case that internal V&V has high quality.

The above constructed BBNs can thus be used to support the evaluation of different scenarios to assist process tailoring. However, BBNs can only provide a general indication. For detailed interaction or numerical information, a process simulator can be used for tailoring purpose. We have constructed a software process simulator based on the derived BBN factors and relations. Details can be found in [5].

6. Conclusion

This research combined the advantages of UML's modeling power and BBNs' treatment of uncertainties for software process tailoring. A systematic way was presented to use UML to model requirements of industrial standards, and then BBNs' causal diagrams can be derived from these UML diagrams for process tailoring consideration. Thus, it alleviates the frequently criticized problem of subjective construction of BBN's causal dependency diagrams. Our proposed approach integrating UML and BBNs can further be used to assist decision-making in other software project management activities, such as planning and risk management.

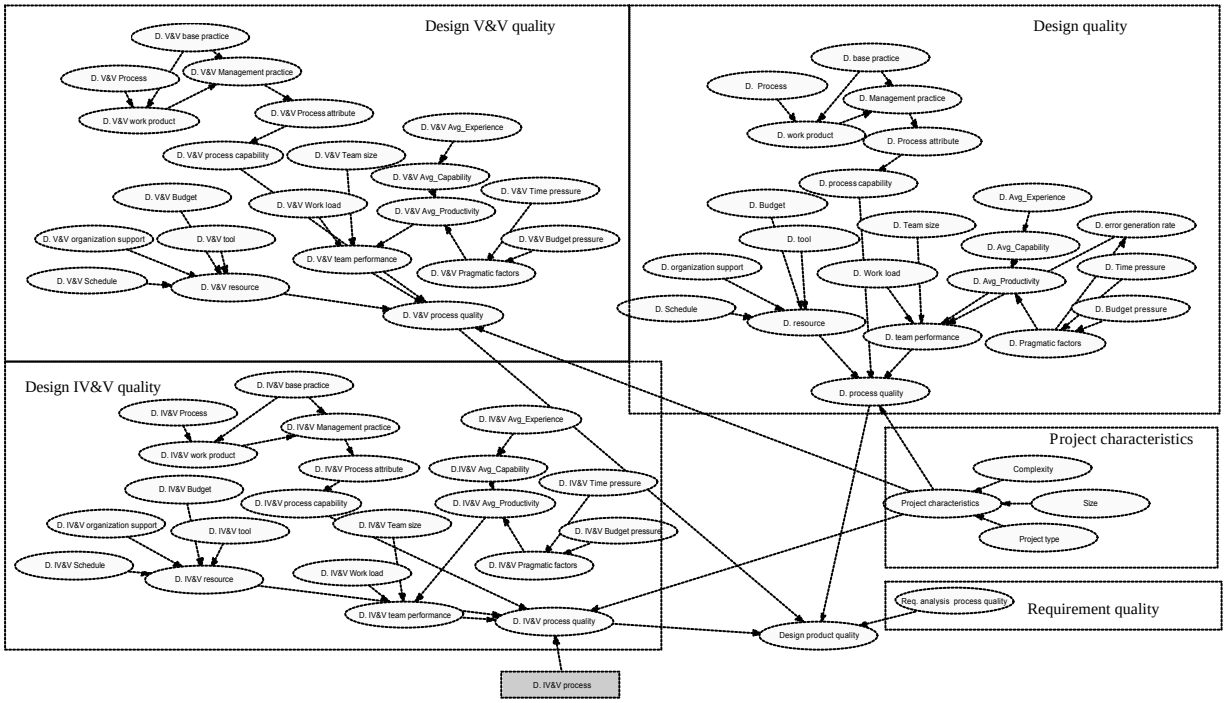


Figure 16. BBN diagrams with design phase factors expanded

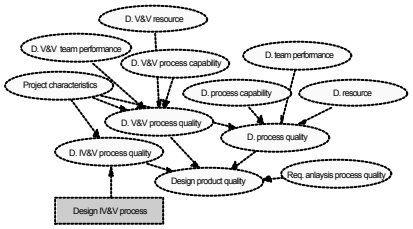


Figure 17. Whether to perform Design IV&V

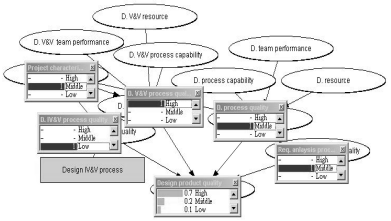


Figure 18. Not performing Design IV&V when Internal V&V quality high

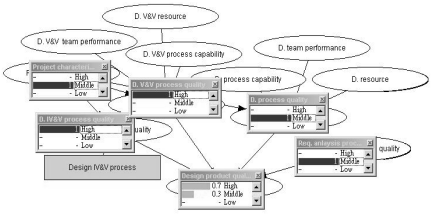


Figure 19. Perform IV&V when internal V&V quality high

Acknowledgements

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

References

[1] F. V. Jensen, *An Introduction to Bayesian Networks*, Springer, 1996.

[2] HUGIN EXPERT, *HUGIN API Reference Manual*, Version 5.2, June 2001.

[3] IEEE Std 12207, IEEE Standard for Information Technology- Software Life Cycle Processes, March 1998.

[4] International Organization for Standardization, ISO/IEC TR 15504: Software Engineering – Software Process Assessment, Part 1 to Part 9, ISO/IEC Technical Report, 1998

[5] Wan-Hui Tseng, *Development and Application of Software Process Tailoring Techniques*, M.S. Thesis, Computer Science and Engineering Dept., Yuan-Ze University. Taiwan. 2003