Using Bayesian Networks to Model and Analyze Software Product Line Feature Model

Musfiqur Rahman and Shamim Ripon

Department of Computer Science and Engineering,
East West University, Dhaka, Bangladesh
m.rahman.2014@ieee.org, dshr@ewubd.edu

Abstract. Proper management of requirements plays a significant role in the successful development of any software product family. Application of AI, Bayesian Network (BN) in particular, is gaining much interest in Software Engineering, mainly in predicting software defects and software reliability. Feature analysis and its associated decision making is a suitable target area where BN can make remarkable effect. In SPL, a feature tree portrays various types of features as well as captures the relationships among them. This paper applies BN in modeling and analyzing features in a feature tree. Various feature analysis rules are first modeled and then verified in BN. The verification confirms the definition of the rules and thus these rules can be used in various decision making stages in SPL.

Keywords: Software Product Line, Bayesian Networks, False Optional, Dead feature.

1 Introduction

Software Product Line (SPL) is a set of related softwares, also known as software family, where the member products of the family share some common features and each member is characterized by their varying features [4]. The main objectives behind SPL is reusability, time to market, increased quality [3,4]. The common and varying features of a SPL are arranged in model that helps the stakeholder to select their required product feature configuration. Common requirements among all family members are easy to handle as they simply can be integrated into the family architecture and are part of every family member. But problem arises from the variant requirements among family members as modeling variants adds an extra level of complexity to the domain analysis. Thus management of variants is considered to be one of the critical areas in SPL.

Constructing a feature model that correctly represents the intended domain is a critical task [2] and defects may be introduced while constructing the feature model. If defects are not identified and corrected at proper stage, it can diminish the expected benefit of SPL [10]. Among the various types of defects found in feature model, *dead* and *false optional* features are two most common types of defects that are of our interest in this paper. A dead feature is a feature

M.N. Murty et al. (Eds.): MIWAI 2014, LNAI 8875, pp. 220–231, 2014.

that is part of a feature tree but it never appears in any valid product [9,10]. Having a dead feature in FM indicates inaccurate representation of the domain requirements. A false optional feature on the other hand is a feature that is declared as an optional feature but it appears in all valid products. Both dead and false optional feature arise due to misuse of dependencies among features.

Due to increased complexity and uncertainly in software requirements, application of AI techniques are becoming evident for managing them [12]. Bayesian Network (BN) [7,8,13] has already been successfully applied to various areas in Software Engineering [6,11,14,15,19] as well as in Requirement Engineering [1]. Besides, a probabilistic feature model is presented in [5] by showing its use in modeling, mining and interactive configuration. Although various work have been carried out where BNs have been used to predict software defects, reliability, etc., very few addressed the requirement management issues in SPL, in particular feature model analysis. Our focus in this paper is to address this area where the inference mechanism and conditional probability in BNs can be used to analyze various defects in feature model.

The objective of this paper is to adopt AI technique to analyze the defects in SPL feature models. This paper focuses on two specific analysis operations: *false optional* and *dead features* of feature diagram. By extending our earlier definition [17], we define several analysis rules for false optional and dead features by using First Order Logic (FOL). We then represent these rules by using BNs. First, we define BNs graphs to represent the scenarios of the our analysis rules. After defining node probability tables of the variables of BNs graphs, the conditional probability of the variables related to analysis rules are calculated. The calculated results match our FOL analysis rules.

The rest of the paper is organized as follows. Section 2 gives a brief review of feature tree and BN and shows how to draw BN of a feature tree. Section 3 defines First Order Logic based analysis rules of false optional and dead features. Section 4 models the analysis rules in BN and then show the probabilistic calculation for the features. Finally, in Section 5 we conclude the paper by summarizing our contributions and outlining our future plans.

2 Feature Diagram and Bayesian Network

Feature modeling is a notation proposed in [9] as a part of domain analysis of system family. Feature modeling facilitates addressing the common and varying features of a product family in both formally and graphically. In feature model features are hierarchically organized as trees where root represents domain concepts and other nodes represents features. Features are classified as *Mandatory* and *Optional.* The relationship among the child sub-features of a parent feature are categorized into *Or* and *Alternative.* When there is a relationship between cross-tree (or cross hierarchy) features (or variation points) we denote it as a dependency.

BN is a directed graph where the nodes represent variables (events), and the arrows between the nodes represent relationship (dependencies). Each node in the graph has an associated set of Conditional Probability Distributions (CPD). For discrete variables, the CPD can be represented as a Node Probability Table (NPT) which list the probabilities that given node takes on each of its different values of each combination of values of its parents. When the nodes have two possible values, binary probabilities (0 or 1) are used in CPD. The construction of a BN starts with the identification of the relevant variables in the domain that has to be modeled. After that the dependencies between the variables has to be determined. Finally CPD is added for each variable.

A BN also corresponds to a directed acyclic graph (DAG). The directed edges represent the direction of dependencies between the nodes. A directed edge from node X_i to X_j indicates that X_i is an ancestor of X_j and the value of X_j depends on the value of X_i. Feature tree on the other hand is a rooted tree, where non-root nodes represent features, and features (parent) and sub-features (child) are connected via edges. Due to the strong resemblance between BNs and feature trees, BNs can be conveniently used to represent the information within feature tree. Added with this, BNs can also include probabilistic information to support decision making in uncertain situations. Figure 1 shows an example feature tree and its corresponding BN representation. Based on the dependency information of the the variable, a NPT can be constructed for each variable in the BN.

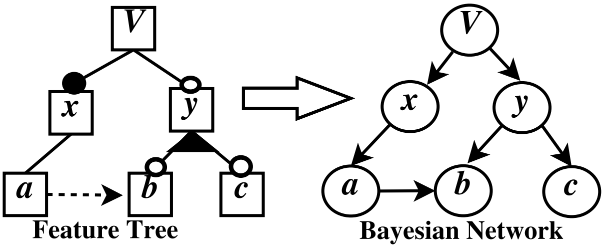


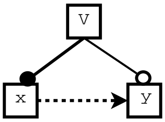
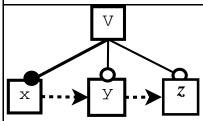
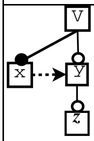
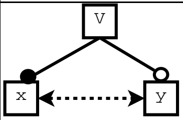
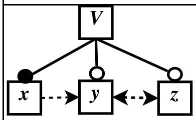
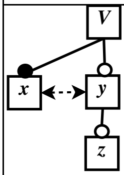
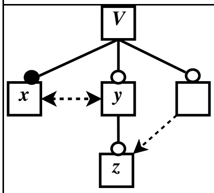
Fig. 1. BN representation of a feature tree

3 Feature Modeling Analysis Rules

Defects in feature model have adversary effects on the quality of the derived product model. Dead and false optional features are two most common types of defects found in any FM. A dead feature is a feature that is part of a feature tree but it never appears in any valid product [9,10]. Having a dead feature in FM indicates inaccurate representation of the domain requirements. A false optional feature on the other hand is a feature that is declared as an optional feature but it appears in all valid products. Both dead and false optional feature arise due to misuse of dependencies among features. Extending our earlier work [17] on logical representation of feature model and their analysis rules, we present in this section a set of rules for dead and false optional features. Our definitions are also influenced by those mentioned in [16].

We use the following FOL predicates to define the rules.

Table 1. Summary of Analysis Rules

	Scenario	**Analysis Rules**
False Optional		$\forall v, x, y \cdot variation\_point(v) \wedge mandatory\_variant(v, x)$ $\wedge\ optional\_variant(v, y) \ \wedge\ requires(x, y)$ $\wedge\ select(x) \Rightarrow select(y)$
		$\forall v, x, y, z \cdot variation\_point(v) \wedge mandatory\_variant(v, x)$ $\wedge\ optional\_variant(v, y) \ \wedge optional\_variant(v, z)$ $\wedge\ requires(x, y) \ \wedge\ requires(y, z)$ $\wedge\ select(x) \Rightarrow select(y) \ \wedge\ select(z)$
		$\forall v, x, y, z \cdot variation\_point(v) \wedge mandatory\_variant(v, x)$ $\wedge\ optional\_variant(v, y) \ \wedge\ variation\_point(y)$ $\wedge\ optional\_variant(y, z) \wedge requires(x, y)$ $\wedge\ select(x) \Rightarrow select(y) \ \wedge\ select(z)$
Dead Features		$\forall v, x, y \cdot variation\_point(v) \wedge mandatory\_variant(v, x)$ $\wedge\ optional\_variant(v, y) \ \wedge\ excludes(x, y)$ $\wedge\ select(x) \Rightarrow \neg select(y)$
		$\forall v, x, y, z \cdot variation\_point(v) \wedge mandatory\_variant(v, x)$ $\wedge\ optional\_variant(v, y) \ \wedge optional\_variant(v, z)$ $\wedge\ requires(x, y) \ \wedge\ excludes(y, z)$ $\wedge\ select(x) \Rightarrow select(y) \ \wedge\ \neg select(z)$
		$\forall v, x, y, z \cdot variation\_point(v) \wedge mandatory\_variant(v, x)$ $\wedge\ optional\_variant(v, y) \ \wedge\ variation\_point(y)$ $\wedge\ optional\_variant(y, z) \wedge excludess(x, y)$ $\wedge\ select(x) \Rightarrow \neg select(y) \ \wedge\ \neg select(z)$
		$\forall v, x, y, z \cdot variation\_point(v) \wedge mandatory\_variant(v, x)$ $\wedge\ optional\_variant(v, y) \wedge\ optional\_variant(v, w)$ $\wedge\ variation\_point(y) \ \wedge\ optional\_variant(y, z)$ $\wedge\ excludess(x, y) \wedge requires(w, z) \wedge\ select(x)$ $\Rightarrow \neg select(y) \ \wedge\ \neg select(z) \wedge\ \neg select(w)$

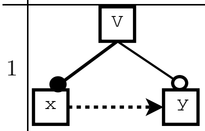
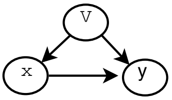
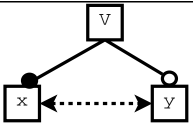
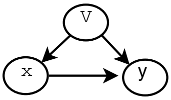
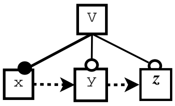
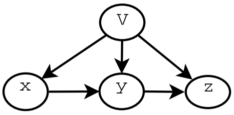
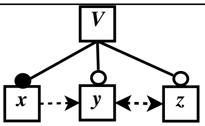
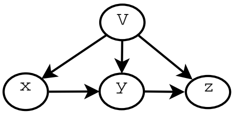
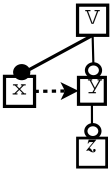
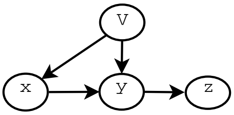
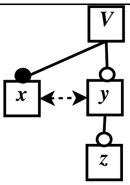
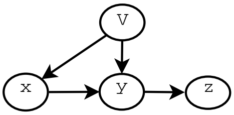
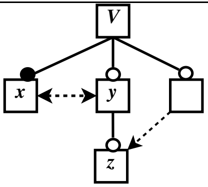
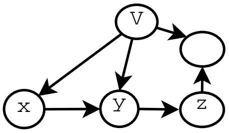
- *variation_point*(v): This predicate indicates that feature v is a variation point, i.e., feature v has child feature(s).
- *mandatory_variant*(v, x): This predicate indicates that feature x is a mandatory feature of feature v, i.e., x is a mandatory child of v.
- *optional_variant*(v, x): Here, x is an optional feature of v.
- $requires(x, y)$: It indicates that feature x requires feature y.
- $exclude(x, y)$: This predicate indicates that feature x and y are mutually exclusive.
- $select(x)$: This predicates the selection of a variant x.

The three analysis rules are defined for false optional features and four analysis rules for dead features. Table 1 illustrates the rules written using FOL. Later these definitions will be used during the analysis of Bayesian Networks based definitions.

4 Analysis Rules in BN

After analyzing the scenarios for false optional and dead features in a feature tree, we define Bayesian network representation of the scenarios. The key factor in defining the BN presentation is to identify the dependencies among the features. Our BN is heavily influenced by our logical representation shown in [17]. Table 2 illustrates the BN of the analysis rules mentioned earlier.

Table 2. BN representation of false optional and dead feature analysis rules

	False Optional Features		Dead Feature	
	Feature Tree	**Bayesian Networks**	**Feature Tree**	**Bayesian Networks**
1				
2				
3				
4				

False Optional, Rule 1: An optional feature becomes false optional when a mandatory feature requires that optional feature. In the Bayesian network (1st rule of false optional in Table 2), both x and y are two variants of the variation point v. Thus the probability of both of these features being selected are directly dependent on v. Feature x depends only on v, whereas y depends on both x and v. The NPT of the variable v, x and y is given in Fig. 2.

$\boldsymbol{v}$

T	F
1	0

$\boldsymbol{x}$

V	T	F
T	1	0
F	0	1

$\boldsymbol{y}$

V	x	T	F
F	F	0	1
F	T	I	I
T	F	I	I
T	T	1	0

Fig. 2. NPT for variables in false optional rule 1

In the NPT, I stands for inconsistent. From the NPT the probability of selecting the variable y can be calculated by using Bayesian rule,

$$\begin{aligned} P(y,x,v) &= P(y \mid x,v)P(x \mid v)P(v) \\ P(y=T \mid x=T) &= \frac{P(x=T,y=T)}{P(x=T)} = \frac{\sum_{v\in\{T,F\}} P(x=T,y=T,v)}{\sum_{v\in\{T,F\}} P(x=T,v)} \\ &= \frac{P(y=T,x=T,v=T)+P(y=T,x=T,v=F)}{P(x=T,v=T)+P(x=T,v=F)} \end{aligned}$$

$$\begin{aligned} P(y=T,x=T,v=T) &= P(y=T \mid x=T,v=T)P(x=T \mid v=T)P(v=T) \\ &= 1\times 1\times 1\times 1 = 1 \end{aligned}$$

Similarly, the probabilities of other variables can also be calculated. Finally we get the following conclusion,

$$P(y=T \mid x=T) = \frac{1+0}{1+0} = 1$$

The result indicates that the optional variant y will be always selected whenever the variant x is selected. As x is a mandatory feature, it will always be selected and thus y will be always part of a valid product, even though it is declared as optional. Hence, y is a false optional feature.

False Optional, Rule 2: An optional feature becomes false optional when it is required by another false optional feature. In the 2nd rule for false optional in Table 2, x is a mandatory feature which requires an optional feature y. Thus y becomes a false optional feature. Moreover, y also requires an optional feature z which implies that z is also a false optional feature. As the three variants x, y, and z are the variants of v, in BN all of these variables are directly dependent of v. Besides, y has a dependency on v and x and z has on v and y. Even there is no direct dependency between x and z, variant z is selected whenever x is selected. The NPT of the variables v, x, y and z are given in Fig 3.

From the NPT the probability of selecting the variable z can be calculated by using Bayesian rule,

v

T	F
1	0

x

v	T	F
T	1	0
F	0	1

y

v	x	T	F
F	F	0	1
F	T	I	I
T	F	I	I
T	T	1	0

z

v	y	T	F
F	F	0	1
F	T	I	I
T	F	I	I
T	T	1	0

Fig. 3. NPT for false optional rule 2

$$P(z,y,v) = P(z \mid y,v)P(y \mid x,v)P(x \mid v)P(v)$$

$$P(z=T \mid y=T) = \frac{P(z=T,y=T)}{P(y=T)} = \frac{\sum_{x,v\in\{T,F\}} P(z=T,y=T,x,v)}{\sum_{x,v\in\{T,F\}} P(y=T,x,v)}$$

$$= \frac{\begin{array}{c} P(z=T,y=T,x=T,v=T)+P(z=T,y=T,x=F,v=F)+ \\ P(z=T,y=T,x=T,v=F)+P(z=T,y=T,x=F,v=T) \end{array}}{\begin{array}{c} P(y=T,x=T,v=T)+P(y=T,x=F,v=F)+ \\ P(y=T,x=T,v=F)+P(y=T,x=F,v=T) \end{array}}$$

By using the values from NPT the following conclusion can be drawn,

$$P(z=T \mid y=T) = \frac{1+0+0+0}{1+0+0+0} = 1$$

The result states that the optional feature z is selected whenever feature y is selected. As feature y depends on feature x which is a mandatory feature, y is actually a false optional feature and hence z is also a false optional feature which will always be part of a valid product even after it is declared as an optional.

False Optional, Rule 3: A feature becomes false optional when its ancestor is a false optional feature. In the 3rd rule of false optional feature in Table 2, x is a mandatory feature and y is an optional feature of variation point v. Furthermore, z is an optional feature of y. As y is required by x, it becomes a false optional feature resulting z into a false optional feature as well. In the BN, y depends on both x and v and z only depends on y.

v

T	F
1	0

x

v	T	F
T	1	0
F	0	1

y

v	x	T	F
F	F	0	1
F	T	I	I
T	F	I	I
T	T	1	0

z

y	T	F
T	1	0
F	0	1

Fig. 4. NPT for false optional rule 3

$$\begin{aligned}
P(z,y) &= P(z \mid y)P(y \mid x,v)P(x \mid v)P(v)\\
P(z=T \mid y=T) &= \frac{P(z=T,y=T)}{P(y=T)} = \frac{\sum_{x,v\in\{T,F\}} P(z=T,y=T,x,v)}{\sum_{x,v\in\{T,F\}} P(y=T,x,v)}\\
&= \frac{\begin{array}{c}P(z=T,y=T,x=T,v=T)+P(z=T,y=T,x=F,v=F)+\\ P(z=T,y=T,x=T,v=F)+P(z=T,y=T,x=F,v=T)\end{array}}{\begin{array}{c}P(y=T,x=T,v=T)+P(y=T,x=F,v=F)+\\ P(y=T,x=T,v=F)+P(y=T,x=F,v=T)\end{array}}
\end{aligned}$$

By using the values from NPT in Fig. 4 the following conclusion can be drawn,

$$P(z=T \mid y=T) = \frac{1+0+0+0}{1+0+0+0} = 1$$

The result shows that z is a false optional feature.

Dead Feature, Rule 1: An optional feature becomes dead feature when it is excluded by a mandatory feature. In first rule for dead features in Table 2 the graph x is a mandatory feature which excludes optional feature y. Variant y can never be part of a valid configuration and thus its a dead feature. In the BN, the probability of both of these feature being selected are directly dependent on v. The probability of y being selected is also dependent on x.

v

T	F
1	0

x

v	T	F
T	1	0
F	0	1

y

v	x	T	F
F	F	0	1
F	T	I	I
T	F	I	I
T	T	0	1

Fig. 5. NPT for dead feature rule 1

From the NPT in Fig. 5 the probability of selecting the variable y can be calculated by using Bayesian rule,

$$\begin{aligned}
P(y,x,v) &= P(y \mid x,v)P(x \mid v)P(v)\\
P(y=T \mid x=T) &= \frac{P(x=T,y=T)}{P(x=T)} = \frac{\sum_{v\in\{T,F\}} P(x=T,y=T,v)}{\sum_{v\in\{T,F\}} P(x=T,v)}\\
&= \frac{P(y=T,x=T,v=T)+P(y=T,x=T,v=F)}{P(x=T,v=T)+P(x=T,v=F)}
\end{aligned}$$

The probabilities of variables can be calculated from NPT as shown for other rules. Finally we get the following conclusion,

$$P(y=T \mid x=T) = \frac{0+0}{1+0} = 0$$

Thus, y becomes a dead feature.

Dead Feature, Rule 2: An optional feature becomes dead when it is excluded by a false optional feature. In the 2nd rule for dead feature in Table 2, mandatory feature x requires an optional feature y. Thus y becomes a false optional feature. Moreover, y excludes the optional feature z which implies that z is a dead feature. In the BN all of the features x, y and z are directly dependent of v. Besides, y has a dependency on x and z has a dependency on y.

$\boldsymbol{v}$

T	F
1	0

$\boldsymbol{x}$

v	T	F
T	1	0
F	0	1

$\boldsymbol{y}$

v	x	T	F
F	F	0	1
F	T	I	I
T	F	I	I
T	T	1	0

$\boldsymbol{z}$

v	y	T	F
F	F	0	1
F	T	I	I
T	F	I	I
T	T	0	1

Fig. 6. NPT for dead feature rule 2

From the Fig. 6 the probability of selecting the variable z can be calculated by using Bayesian rule,

$$P(z, y, v) = P(z \mid y, v)P(y \mid x, v)P(x \mid v)P(v)$$

$$P(z = T \mid y = T) = \frac{P(z = T, y = T)}{P(y = T)} = \frac{\sum_{x,v \in \{T,F\}} P(z = T, y = T, x, v)}{\sum_{x,v \in \{T,F\}} P(y = T, x, v)}$$

$$= \frac{\begin{array}{c} P(z = T, y = T, x = T, v = T) + P(z = T, y = T, x = F, v = F) + \\ P(z = T, y = T, x = T, v = F) + P(z = T, y = T, x = F, v = T) \end{array}}{\begin{array}{c} P(y = T, x = T, v = T) + P(y = T, x = F, v = F) + \\ P(y = T, x = T, v = F) + P(y = T, x = F, v = T) \end{array}}$$

By using the values from NPT the following conclusion can be drawn,

$$P(z = T \mid y = T) = \frac{0 + 0 + 0 + 0}{1 + 0 + 0 + 0} = 0$$

The result states that the optional feature z is not selected whenever feature y is selected. y is a false optional feature and z is a dead feature as it will never be part of a valid product even after it is declared as an optional.

Dead Feature, Rule 3: A feature becomes a dead feature when its ancestor is a dead feature. In the dead feature rule 3 in Table 2, x is a mandatory feature of variation point v and y is an optional feature of v. Furthermore, y itself is a variation point and z is its optional feature. As y is excluded by x, it becomes a dead feature resulting z into a dead feature as well. In the BN x and y are dependent of v and z is only dependent of y.

$\boldsymbol{v}$

T	F
1	0

$\boldsymbol{x}$

v	T	F
T	1	0
F	0	1

$\boldsymbol{y}$

v	x	T	F
F	F	0	1
F	T	I	I
T	F	I	I
T	T	0	1

$\boldsymbol{z}$

y	T	F
T	1	0
F	0	1

Fig. 7. NPT for dead feature rule 3

$$P(z, y) = P(z \mid y)P(y \mid x, v)P(x \mid v)P(v)$$

$$P(z = T \mid y = F) = \frac{P(z = T, y = F)}{P(y = F)} = \frac{\sum_{x,v\in\{T,F\}} P(z = T, y = F, x, v)}{\sum_{x,v\in\{T,F\}} P(y = F, x, v)}$$

$$= \frac{\begin{array}{c} P(z = T, y = F, x = T, v = T) + P(z = T, y = F, x = F, v = F)+ \\ P(z = T, y = F, x = T, v = F) + P(z = T, y = F, x = F, v = T) \end{array}}{\begin{array}{c} P(y = F, x = T, v = T) + P(y = F, x = F, v = F)+ \\ P(y = F, x = T, v = F) + P(y = F, x = F, v = T) \end{array}}$$

By using the values from Fig. 7 the following conclusion can be drawn,

$$P(z = T \mid y = T) = \frac{0 + 0 + 0 + 0}{0 + 0 + 0 + 0} = 0$$

The result shows that z is a dead feature.

Dead Features, Rule 4: A feature becomes dead when it requires another dead feature. In the 4th rule of dead feature in Table 2 v is a variation point of three variant x, y and w where x is a mandatory feature and the other two are optional. Feature x and y are mutually exclusive. Thus y becomes a dead feature. Moreover, y is the variation point for another optional feature z. Since y is a dead feature, z is also a dead feature. On the contrary, the optional feature w requires z which is already a dead feature. Hence w is also a dead feature.

$\boldsymbol{v}$

T	F
1	0

$\boldsymbol{x}$

v	T	F
T	1	0
F	0	1

$\boldsymbol{y}$

v	x	T	F
F	F	0	1
F	T	I	I
T	F	I	I
T	T	0	1

$\boldsymbol{z}$

y	T	F
T	1	0
F	0	1

$\boldsymbol{w}$

v	z	T	F
F	F	0	1
F	T	I	I
T	F	I	I
T	T	1	0

Fig. 8. NPT for dead feature rule 4

$$P(w, y, z) = P(w \mid z, v)P(z \mid y)P(y \mid x, v)P(x \mid v)P(v)$$

$$P(w = T \mid z = F) = \frac{P(w = T, z = F)}{P(z = F)} = \frac{\sum_{x,y,v\in\{T,F\}} P(w = T, z = F, x, y, v)}{\sum_{x,yv\in\{T,F\}} P(z = F, x, y, v)}$$

All the above probabilities are calculated from Fig. 8. Finally, we derive the following conclusion which shows that z is a dead feature.

$$P(w = T \mid z = F) = \frac{0+0+0+0}{0+0+0+0} = 0$$

5 Conclusion

This paper presented an approach to define and verify software product line feature analysis rules by using Artificial Intelligence-based technique. During any software development project, handling of requirement is inherently difficult and uncertain, which make it amenable for the application of AI technique. In this work our focus was in a particular area of SE, Software Product Line. We defined a set of rules for both dead and false optional features of SPL feature diagram. BN is used to model and verify the analysis rules. In comparison to our earlier work on applying logic [17] and semantic web-based approach [18] to verify analysis rules, BN-based analysis not only manage to solve uncertain situation but also establishes a synergy between AI techniques and SPL feature analysis. Our future plan includes the extension of the analysis rules for various other analysis operations. We are also interested to apply any BN tool to perform the verification mechanically to avoid human error.

References

1. Barry, P.S., Laskey, K.B.: An application of uncertain reasoning to requirements engineering. In: Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence, UAI 1999, pp. 41–48. Morgan Kaufmann Publishers Inc., San Francisco (1999)
2. Benavides, D., Segura, S., Cortés, A.R.: Automated analysis of feature models 20 years later: A literature review. Inf. Syst. 35(6), 615–636 (2010)
3. Bosch, J.: Design and use of software architectures - adopting and evolving a product-line approach. Addison-Wesley (2000)
4. Clements, P.C., Northrop, L.: Software Product Lines: Practices and Patterns. SEI Series in Software Engineering. Addison-Wesley (August 2001)
5. Czarnecki, K., She, S., Wasowski, A.: Sample spaces and feature models: There and back again. In: International Software Product Line Conference, pp. 22–31 (2008)
6. Fenton, N.E., Neil, M., Marsh, W., Hearty, P., Marquez, D., Krause, P., Mishra, R.: Predicting software defects in varying development lifecycles using bayesian nets. Information & Software Technology 49(1), 32–43 (2007)
7. Jensen, F.V.: Bayesian Networks and Decision Graphs. Springer-Verlag New York, Inc., Secaucus (2001)
8. Jensen, F., Nielsen, T.: Bayesian Networks and Decision Graphs, 2nd edn. Springer, New York (2007)
9. Kang, K.C., Cohen, S.G., Hess, J.A., Novak, W.E., Peterson, A.S.: Feature-oriented domain analysis (foda) feasibility study. Tech. rep., Carnegie-Mellon University Software Engineering Institute (November 1990)

10. von der Massen, T., Lichter, H.: Deficiencies in Feature Models. In: Mannisto, T., Bosch, J. (eds.) Workshop on Software Variability Management for Product Derivation - Towards Tool Support (2004)
11. de Melo, A.C.V., de J. Sanchez, A.: Software maintenance project delays prediction using bayesian networks. Expert Syst. Appl. 34(2), 908–919 (2008)
12. Meziane, F., Vadera, S.: Artificial intelligence applications for improved software engineering development: New prospects. IGI Global, Hershey (2009)
13. Pearl, J.: Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference. Morgan Kaufmann Publishers Inc., San Francisco (1988)
14. Pendharkar, P.C., Subramanian, G.H., Rodger, J.A.: A probabilistic model for predicting software development effort. IEEE Trans. Software Eng. 31(7), 615–624 (2005)
15. Radliński, L., Fenton, N., Neil, M., Marquez, D.: Improved decision-making for software managers using bayesian networks. In: Proceedings of the 11th IASTED International Conference on Software Engineering and Applications, SEA 2007, pp. 13–19. ACTA Press, Anaheim (2007)
16. Rincón, L.F., Giraldo, G., Mazo, R., Salinesi, C.: An ontological rule-based approach for analyzing dead and false optional features in feature models. Electr. Notes Theor. Comput. Sci. 302, 111–132 (2014)
17. Ripon, S., Azad, K., Hossain, S.J., Hassan, M.: Modeling and analysis of product-line variants. In: de Almeida, E.S., Schwanninger, C., Benavides, D. (eds.) SPLC (2), pp. 26–31. ACM (2012)
18. Ripon, S., Piash, M.M., Hossain, S.A., Uddin, S.: Semantic web based analysis of product line variant model. International Journal of Computer and Electrical Engineering (IJCEE) 6(1), 1–6 (2014)
19. del Sagrado, J., del Águila, I.M., Orellana, F.J.: Architecture for the use of synergies between knowledge engineering and requirements engineering. In: Lozano, J.A., Gámez, J.A., Moreno, J.A. (eds.) CAEPIA 2011. LNCS (LNAI), vol. 7023, pp. 213–222. Springer, Heidelberg (2011)