Abstract

Managing knowledge is both a challenging and complex task. There is a number of techniques for making decisions in today's knowledge-based economies. Decision support systems (DSS) have been developed to aid the decision-making process to find solutions for multiple problems. One aspect that hinders successful decision making is the issue of variability definition. The aim of this paper is to define variability by proposing an intelligent method. Knowledge representation is based in two layers: 1) the upper layer i.e. graphical representation and 2) the lower layer i.e. a mathematical algorithm. We present a method that defines and provides autosupport for five operations in knowledge validation particularly dependency constraint rules, propagation, delete-cascade, logical inconsistency and dead choice detection.

1. Introduction and Motivations

Knowledge has become the main value driver for modern organizations and has been described from among different assets as a critical competitive asset of organizations. An important feature in the development and application of knowledge-based systems is the knowledge representation techniques used. A successful knowledge representation technique provides a means for expressing knowledge in a way that can be interpreted and reasoned with by both humans and machines.

For any decision, there are many choices that the decision maker can select, one or more from them [1]. Due to shortage in variability definition, this paper defines decision choices as variability in DSS. Although many approaches exist for knowledge representation in DSS, the design and implementation for a useful method that considers variability in DSS is very much desired. The problem of representing variability in DSS requires complex representation that captures static and dynamic aspects of choices that can be encountered during the decision process. We believe that the key feature of such knowledge representation (for variability in DSS) is its capability of precise representation of diverse types of choices and associations between them. This involves: i) qualitative or quantitative description of choices and their classification, ii) representation of causal relationships between choices and iii) the possibility of computerized representation.

The limitation of symbolic knowledge representation has led to the study of more effective models for knowledge representation [2]. Variability (choices) is one of the most valuable aspects in DSS therefore; an effective knowledge representation technique is required.

Malhotra [3] defines information sharing culture challenge of Knowledge Management systems as integration of decision-making and actions across inter-enterprise boundaries. Therefore, existence of a method to validate a DSS is highly recommended. Task variability defined in [4], highlights the number of exceptions encountered in the characteristics of the work. This study tests the importance of variability in system satisfaction. In [5] the source of logical inconsistency is defined from a skill-based or rule-based errors that would include errors made in touch-typing, in copying values from one list to another, or other activities that frequently do not require a high level of cognitive effort. One of the main drawbacks coming from the fusion of several different and partial views is a logical inconsistency [6].

Nowadays, Feature Model (FM) [7] and Orthogonal Variability Model (OVM) [8] are regarded as well-known techniques to represent variability. Inconsistency detection is defined as a challenging operation to validate variability in [9]. Dead feature defined in [10], is a frequent case of error in feature-model based variability. This can also be coined as a dead choice. Modeling variability methods must be considered as constraint dependency rules to ensure the correctness of the decision. Two operations (propagation and delete...
cascade) are proposed in this paper to support auto
selection of choices in the decision making process.
This structure of this paper will be as follows: section
two defines variability in general, section three
defines variability, section four and five highlights
representation and validation and lastly section six
discusses related work. Section seven concludes the
paper and describes future work.

2. Definitions

2.1. Variability

The term variability generally refers to the ability
to change, to be more precise; the kind of variability
we are interested in does not occur by chance but is
brought about on purpose. In other words we mean
alternate ways to represent choices. Pohl suggests
three questions to define variability [8].

- **What does vary?** : identifying precisely the
  variable item or property of the real world. The
  question leads us to the definition of the term
  variability subject. A variability subject is a
  variable item of the real world or a variable
  property of such an item.
- **Why does it vary?** : There are different reasons for
  an item or property to vary: different stakeholder
  needs, different laws, technical reasons, etc.
  Moreover, in the case of interdependent items, the
  reason for an item to vary can be the variation of
  another item.
- **How does it vary?** : This question deals with the
  different shapes a variability subject can take. To
  identify the different shapes of a variability subject
  we define the term variability object (a particular
  instance of a variability subject).

**Example of variability Subject and Objects for**
“Car”: The variability subject “car” identifies a
property of real-world items. Examples of
variability objects for this variability subject are
Toyota, Nissan and Proton.

3. Variability in DSS

In this section, variability is DSS is described. By
variability in DSS, we mean choices. In the choice
phase, the decision maker chooses a solution to the
problem or opportunity DSS help remind the decision
maker what methods of choice are appropriate for the
problem and by helping to organize and present the
information [1]. We can look for choices as a term of
between terms of knowledge such as kind-of, and
part-of become more important than the term itself",
the proposed techniques define and deal with these
types of relationship and with constraints between
choices such as requires and excludes. However,
there is no standardization to represent variability in
DSS [12].

Our proposed method can enhance both readability
and clarity in representation of variability in DSS.
Consequently, it also represents variability in high
degree of visualization considering the constraints
between choices. As mentioned earlier, in definition
of variability, there are two items of variability
subject and variability object, we suggest variability
subject as a decision point and variability objects and
the choices. For example the variability subject
“Promotion” identifies a decision point. Examples of
variability objects for this variability subject are
Experience, Qualifications, or Special Report. This
example illustrates three choices Experience,
Qualifications, or Special Report) that the decision
maker can select from them in the decision point
Promotion.

**A reward system example:** Rewards systems can
range from simple systems to sophisticated ones in
which many alternatives are there. It is closely in
relations to performance management. Both
rewarding and performance measurement are difficult
tasks not only because the area of human resource is
changing fast but also due to the decision variability
that takes place in different activities of a HR cycle as
it shown in figure 1.

4. Representing Variability in DSS Using
First Order Logic

In this section, the notations of the proposed
method are explained. First, the upper layer is
described and later decision point, choices, and
constraint dependency rules are represented using
predicates as a lower layer. The output of the process
is a complete modeling of variability in DSS as a
knowledge- base.
4.1. Upper Layer Representation of the proposed method

Decision points, choices and constraint dependency rules describe variability. Constraint dependency rules are: Decision point requires or excludes Decision point, choice requires or excludes choice, and choice requires or excludes Decision point. We combined FM diagram with OVM notations. Figure 1 represents the upper layer of our proposed method. Optional and mandatory constraints are defined in figure 1 by original FM notations [7], and constraint dependency rules are described using OVM notations. OVM and FM can easily become very complex for validating a medium level system, i.e., several thousand decision points and choices are required. This reason motivated us to develop a more intelligent approach to represent and validate variability in DSS.

4.2. Lower layer of the proposed method

In this sub-section, decision points, choices, and dependency constraint rules are described using predicates as a low level of the proposed method: (Examples are based on figure 1. Terms that start by capital letters represent variables and terms that start by lower letters represents constants):

4.2.1. Decision Points

The following five predicates that are used to describe each decision point:

i. type(Name1, decisionpoint). Define the type, e.g. type (promotion, decisionpoint).

ii. choiceof(Name1, Name2). Identifies the choice of specific decision point. Name1 represents name of decision point and Name2 represents name of choice, e.g. choiceof(promotion, promotion decision).

iii. max(Name, int). Identifies the maximum number allowed to be selecting of specific decision point. e.g. max(positive performance, 2).

iv. min(Name, int). Identifies the minimum number allowed to be selecting of specific decision point. e.g. min(positive performance, 1). The common choices(s) in a
decision point is/are not included in maximum-minimum numbers of selection.

v. Common (Name1, yes). Common (Name2, no). Describes the commonality of decision point, e.g. common (promotion, yes) If the decision point is not common, the second slot in the predicate will become no -as example- common (punishment, no).

4.2.2. Choice:
The Following two predicates are used to describe a choice

i. type (Name1, choice). Define the type, e.g. type (recognition, choice).

ii. common (Name1, yes). common (Name2, no). Describes the commonality of a choice, e.g. common (first reminder, yes). If the choice is not common, the second slot in the predicate will become no -as example- common (time on, no).

4.2.3. Constraint Dependency Rules

The following six predicates are used to describe constraint dependency rules:

i. requires_c_c (Name1, Name2). Choice requires choice, e.g. requires_c_c (promotion decision, recognition).

ii. excludes_c_c (Name1, Name2). Choice excludes choice, e.g. excludes_c_c (decrease level, high level).

iii. requires_c_dp (Name1, Name2). Choice requires decision point, e.g. requires_c_dp (promotion decision, positive performance).

iv. excludes_c_dp (Name1, Name2). Choice excludes decision point, e.g. excludes_c_dp (non promotion decision, positive performance).

v. requires_dp_dp (Name1, Name2). Decision point requires decision point, e.g. requires_dp_dp (negative performance, punishment).

vi. excludes_dp_dp (Name1, Name2). Decision point excludes decision point, e.g. excludes_dp_dp (negative performance, positive performance).

5. Validating Variability in DSS

5.1. Constraint Dependency rules

To validate the DM process, the proposed method triggers rules based on constraint dependencies. With regard to validation process result, the choice is added to knowledge-base or rejected, then an explanation of rejection reason is provided and correction actions are suggested.

When a new choice is selected there is a new predicate (select or notselect) would be added to knowledge-base and the backtracking mechanism validates the entire knowledge-base. At the end of the process, select and not notselect predicates represent the selection. Table 1 shows the abstract representation of the main rules in the knowledge-base. The proposed method contains thirteen main rules to validate the selection process based on constraints dependency.

Table 1: Abstract representation of the main rules in the proposed method
Rule 1:
For all choice \( x \) and choice \( y \); if \( x \) requires \( y \) and \( x \) is selected, then \( y \) is selected.

Rule 2:
For all choice \( x \) and choice \( y \); if \( x \) excludes \( y \) and \( x \) is selected, then \( y \) is assigned by \( \text{notselect} \) predicate.

Rule 3:
For all choice \( x \) and decision point \( y \); if \( x \) requires \( y \) and \( x \) is selected, then \( y \) is selected. This rule is applicable as well, if the decision point is selected first and it requires a choice:
\[
\forall \ x, \ y: \text{type}(x, \text{choice}) \land \text{require}_c_{dp}(x, y) \land \text{select}(y) \Rightarrow \text{select}(x)
\]

Rule 4:
For all choice \( x \) and decision point \( y \); if \( x \) excludes \( y \) and \( x \) is selected, then \( y \) is assigned by \( \text{notselect} \) predicate. This rule is applicable as well, if the decision point is selected first and it requires a choice:
\[
\forall \ x, \ y: \text{type}(x, \text{choice}) \land \text{require}_c_{dp}(x, y) \land \text{select}(y) \Rightarrow \text{notselect}(x)
\]

Rule 5:
For all decision point \( x \) and decision point \( y \), if \( y \) requires \( y \) and \( y \) is selected, then \( x \) is assigned by \( \text{notselect} \) predicate.

Rule 6:
For all decision point \( x \) and decision point \( y \), if \( x \) excludes \( y \) and \( x \) is selected, then \( y \) is assigned by \( \text{notselect} \) predicate.

Rule 7:
For all choice \( x \) and decision point \( y \), where \( x \) belongs to \( y \) and \( x \) is selected, that means \( y \) is selected. This rule determines the selection of decision point if one of its choices was selected.

Rule 8:
For all decision point \( y \) there exists of choice \( x \), if \( y \) selected and \( x \) belongs to \( y \), \( x \) is selected. This rule states that if a decision point was selected, then there is choice(s) belong to this decision point must be selected.

Rule 9:
For all choice \( x \) and decision point \( y \); where \( x \) belongs to \( y \) and \( y \) defined by predicate \( \text{notselect}(y) \), then \( x \) is assigned by \( \text{notselect} \) predicate. This rule states that if a decision point was excluded, then none of its choices is selected.

Rule 10:
For all choice \( x \) and decision point \( y \); where \( x \) is a common, \( x \) belongs to \( y \), and \( y \) is selected, then \( x \) is selected. This rule states that if a choice is common and its decision point selected then it is selected.

Rule 11:
For all decision point \( y \); if \( y \) is common, then \( y \) is selected. This rule states that if a decision point is common then it is selected in any decision process.

Rule 12:
For all choice \( x \) and decision point \( y \); where \( x \) belongs to \( y \) and \( x \) is selected, then the summation of \( x \) must not be less than the maximum number allowed to be selected from \( y \).

Rule 13:
For all choice \( x \) and decision point \( y \); where \( x \) belongs to \( y \) and \( x \) is selected, then the summation of \( x \) must not be greater than the minimum number allowed to be selected from \( y \). Rules 12 and 13 validate the number of choices' selection considering the maximum and minimum conditions in decision point definition (cardinality definition). The predicate \( \text{sum}(y, (x)) \) returns the summation number of selected choices belongs to decision point \( y \).

5.2. Propagation

This operation define how some choices are select automatically based on select other choices.

Definition 1: Selection of the choice \( n \), \( \text{select}(n) \), is propagated from selection of the choice \( x \), \( \text{select}(x) \), in three cases:

\[
\begin{align*}
\text{i.} & & \forall x, y, z: \text{type}(x, \text{choice}) \land \text{choicedef}(y, x) \land \text{select}(x) \land \text{require}_c_{dp}(y, z) \\
& & \land \text{type}(n, \text{choice}) \land \text{choicedef}(z, n) \\
& & \land \text{common}(n, \text{yes}) \Rightarrow \text{select}(n)
\end{align*}
\]

If \( x \) is a choice and \( x \) belongs to the decision point \( y \) and \( x \) is selected, that means \( y \) is selected (rule 7), and the decision point \( y \) requires a decision point \( z \), that means \( z \) is selected also (rule 5), and the choice \( n \) belongs to the decision point \( z \) and the choice \( n \) is common that means the choice \( n \) is selected (rule 10).

\[
\begin{align*}
\text{ii.} & & \forall x, n: \text{type}(x, \text{choice}) \land \text{type}(n, \text{choice}) \land \text{select}(x) \land \\
& & \text{require}_c_{c}(x, n) \Rightarrow \text{select}(n)
\end{align*}
\]

If the choice \( x \) is selected and it requires the choice \( n \), that means the choice \( n \) is selected, (rule 1). The
selection of choice \( n \) propagated from the selection of \( x \).

iii. \( \forall x,y,z,n: \text{type}(x, \text{choice}) \land \text{select}(x) \land \text{type}(y, \text{choice}) \land \text{type}(z, \text{decisionpoint}) \land \text{requires}_c \text{-} \text{dp}(x, z) \land \text{type}(n, \text{choice}) \land \text{choiceof}(z, n) \land \text{common}(n) \Rightarrow \text{select}(n) \).

If the choice \( x \) is selected and it requires the decision point \( z \) that means the decision point \( z \) is selected (rule 3), and the choice \( n \) is common and is belongs to the decision point \( z \) that means the choice \( n \) is selected (rule 10). The selection of the choice \( n \) propagated from the selection of \( x \).

5.3. Delete-cascade operation

In this sub-section, new operation is illustrated. We called it delete-cascade. This operation validates the automated decision making process in the execution time. The following scenario describes the problem:

If the choice \( x \) is selected in time \( N \) and the two choices \( y \) and \( k \) are propagated due to \( x \) selection, then the decision list (at time \( N \)) = \{ \( x \), \( y \), \( k \) \}. In time \( (N + 1) \), the choice \( m \) is selected, and \( m \) excludes \( x \), then \( x \) is remove from the decision list. The decision list at time \( (N + 1) \) = \{ \( m \), \( y \), \( k \) \}. The presence of the choice \( y \), and \( k \) is not a real choices. The following rule implements delete-cascade.

\( \forall x,y: \text{type}(x, \text{choice}) \land \text{type}(y, \text{choice}) \land \text{select}(x) \land \text{notselect}(y) \Rightarrow \text{notselect}(x). \)

For all choices \( x \), and \( y \); if the choice \( y \) requires \( x \) and \( x \) is selected and \( y \) assigned by \text{notselect} predicate, that means \( y \) is excluded within the configuration process, and \( x \) was selected according to selection of \( y \) (\( y \text{ requires } X \)), then the presence of \( x \) after exclusion of \( y \) is not true. The output for this operation is the assigning of the choice \( x \) with \text{notselect} predicate. The assigning permits a backtracking mechanism to perform delete-cascade operation to verify the selections.

5.4. Logical Inconsistency Detection

Inconsistency occurs from contradictions in constraint dependency rules. It is very complicated because it has different formats and it can occur between groups of choices or between individual choices. As an example: choice \( A \) requires choice \( B \) and choice \( B \) requires choice \( C \) and choice \( C \) excludes choice \( A \).

**Definition 1**
The inconsistency (error) can be detected in five cases:

i. \( \forall x,y: \text{type}(x, \text{choice}) \land \text{type}(y, \text{choice}) \land \text{requires}_c(x, y) \land \text{excludes}_c(y, x) \Rightarrow \text{error}. \)

If choice \( x \) requires choice \( y \) that means selection of \( x \) leads to selection of \( y \) (rule 1). In addition, the choice \( y \) excludes choice \( x \) that means if \( y \) selected, \( x \) must not be selected (rule 2), this is an error.

ii. \( \forall x,y: \text{type}(x, \text{decisionpoint}) \land \text{type}(y, \text{decisionpoint}) \land \text{requires}_d(x, y) \land \text{excludes}_d(y, x) \Rightarrow \text{error}. \)

If decision point \( x \) requires decision point \( y \) that means selection of \( x \) leads to selection of \( y \) (rule 5), and decision point \( y \) excludes decision point \( x \) means if \( y \), selected \( x \) should not be selected (rule 6), this is an error.

iii. \( \forall x,y,z,n: \text{type}(x, \text{choice}) \land \text{type}(y, \text{choice}) \land \text{type}(z, \text{decisionpoint}) \land \text{type}(n, \text{choice}) \land \text{choiceof}(y, x) \land \text{choiceof}(z, n) \land \text{requires}_c(x, n) \land \text{excludes}_d(y, z) \Rightarrow \text{error}. \)

If the choice \( x \) belongs to the decision point \( y \), and the choice \( n \) belongs to the decision point \( z \), and \( x \) excludes \( n \) that means if \( x \) selected \( n \) should be selected (rule1). Selection of the choice \( x \) means selection of the decision point \( y \), and selection of choice \( n \) means selection of decision point \( z \) (rule 7). The decision point \( y \) excludes the decision point \( z \) that means if one of the choices belongs to \( y \) is selected none belongs to \( z \) should be selected (rules 6, 7, and 9), this is an error.

iv. \( \forall x,y,z: \text{type}(x, \text{choice}) \land \text{type}(y, \text{decisionpoint}) \land \text{type}(z, \text{decisionpoint}) \land \text{requires}_c(x, y) \land \text{excludes}_d(x, z) \land \text{excludes}_d(z, y) \Rightarrow \text{error}. \)

If the common choice \( x \) belongs to the decision point \( y \), and \( x \) excludes the decision point \( z \) that means if \( x \) selected no choice belongs to \( z \) should be selected ( rules 4, and rule 9), and the decision point \( y \) requires the decision point \( z \) that means if \( y \) is selected \( z \) should also be selected( rule 5). Selection of decision point \( y \) means selection of common choice \( x \) (rule 10) but \( x \) excludes \( z \), this is an error.

v. \( \forall x,y,z: \text{type}(x, \text{choice}) \land \text{type}(y, \text{decisionpoint}) \land \text{type}(z, \text{decisionpoint}) \land \text{requires}_c(x, y) \land \text{excludes}_d(x, z) \land \text{excludes}_d(z, y) \Rightarrow \text{error}. \)

If the common choice \( x \) belongs to the decision point \( y \), selection of \( x \) means selection of \( y \) (rule 7), and \( x \) requires the decision point \( z \) that means selection of \( x \) leads to selection of \( z \) (rule 3); but the decision point \( y \) excludes the decision point \( z \) which means if \( y \) is selected \( z \) must not be selected (rule 6), this is an error.
5.5. Dead choice detection

The dead choice that is the choice never used in any decision making process.

Definition 2

A choice $x$ can be a dead choice in 3 cases:

i. $\forall x, y, z: type(x, choice)\& type(y, decision point)\& type(z, decision point)
   \& choiceof(y, x)\& choiceof(z, x)\& choiceof(n, x)\& choiceof(n, z)\& choiceof(n, y)
   \& excludes_c_dp(z, y) \Rightarrow dead_choice(x)$.

ii. $\forall x, y, z: type(x, choice)\& type(y, decision point)\& choiceof(y, x)
   \& choiceof(z, decision point)\& choiceof(n, y)\& choiceof(n, z)\& choiceof(n, x)
   \& excludes_dp_dp(z, y) \Rightarrow dead_choice(x)$.

iii. $\forall x, y, n: type(n, choice)\& type(y, decision point)\& choiceof(y, n)
    \& choiceof(y, n)\& choiceof(n, y)\& choiceof(n, x)\& choiceof(n, x)
    \& excludes_c_c(n, x) \Rightarrow dead_choice(x)$.

If the common choice $n$ belongs to the common decision point $y$ that means $n$ must be selected in any selection (rule 11), when $y$ is selected $n$ must be selected (rule 10), which means $n$ must be selected in any selection. The choice $n$ excludes choice $x$ that means $n$ must not be selected in any selections (rule 2).

6. Related Work

The aim of knowledge representation is to use multiple methods to replicate the human way of thinking and human manipulation with information and knowledge [13]. Major prior research works are summarized below.

Haas[14] investigated the feasibility of developing an overarching knowledge representation for Bureau of Labor Statistics information that captured its semantics, including concepts, terminology, actions, sources, and other metadata, in a uniformly applicable way. Haas suggested the (ISO/IEC 11179) standard for metadata, as knowledge representation techniques. Molina [15] reported the advantages of using a knowledge modeling software tool to help developers build DSS. Molina describes the development of DSS system called SAIDA where knowledge represented as components which designed by KSM. Knowledge Structure Manager (KSM) is a knowledge modeling tool that includes and extends the paradigm of taskmethod-domain followed by different knowledge engineering methodologies. KSM provides a library of reusable software components, called primitives of representation that offer the required freedom to the developer to select the most convenient representation for each case (rules, frames, constraints, belief networks, etc.).

Froelich and Wakulicz-Deja [16] investigated that the problems of representing knowledge for DSS in the field of medical diagnosis systems. They suggest a new model of associational cognitive maps (ACM). The ability to represent and reason with the structures of causally dependant concepts is the theoretical contribution of the proposed ACM. Bayesian network proposed as a knowledge representation to represent multiple-point-of-view [17]. The proposed technique can serves as refection of multiple points of view and it surpasses Bayesian network by both, describe dependency constraint rules and provides auto-explanation mechanism. Lu et al. [18] developed a Knowledge-Based Multi-Objective DSS, they represented two types of knowledge declarative and procedural. Declarative knowledge is a description of facts concerns with information about real-world objects and their properties; Procedural knowledge encompasses problem-solving strategies, arithmetical and inferential knowledge. They used text, tables and diagrams to represent these knowledge, the proposed technique solve this problem by suggests standard representation.

Brewster and O’Hara[19] proves difficulties of represent skills, distributed knowledge, or diagrammatic knowledge using Ontologies. Pomerol, et.al [20] used artificial intelligent decision tree to represent operational knowledge in DSS. Christiansson [21] proposed semantic web and temporal databases as knowledge representation techniques for new generation of knowledge management systems. One of the most sophisticated knowledge modeling methodologies is Common KADS [22] Common KADS explains how to model knowledge application through a structural top-down analysis of a problem domain. The outcome of modeling process according to Common KADS consists of three layers that are called contextual
model, conceptual model and design model. This model did not provide mechanism to define relation between objects or between layers. As the best of our knowledge, no specific method introduced variability as a knowledge representation technique in DSS.

7. Scalability Testing

Scalability is approved as a key factor in measuring applicability of the techniques dealing with variability modeling [23]. In this section, we discuss the experiment, which is developed to test the scalability of logical inconsistency detection operation.

7.1. The Experiment

In the following, we describe the process method of our experiment:

- **Generate the decision repository**: repository is generated in terms of predicates (decision point, and choice). We generated four sets contains 1000, 5000, 15000, and 20000 choices consecutively.

- **Create Assumption**: We have three assumptions: i) each decision point and choice has unique name, ii) each decision point is orthogonal, and iii) all decision points have the same number of choices.

- **Define the parameters**: The main parameters are the number of choices and the number of decision points. The remaining eight parameters (common choices, common decision points, choice requires choice, choice excludes choice, decision point requires decision point, decision point excludes decision points, choice requires decision point, and choice excludes decision point) are defined as a percentage. The number of choice-related parameters (such as; common choice) is defined as a percentage of the number of choices. The number of decision point-related parameters (such as; choice requires decision point) is defined as a percentage of the number of decision points.

- **Calculate output**: for each set, we made ten experiments, and calculated execution time as average. The experiments were done with the range (1000-20000) choices, and percentage range of 10%, 25%, and 50%. Figure 2 shows the results of logical inconsistency detection experiment.

8. Conclusion and Future Work

By introducing variability to represent knowledge in DSS we can get both formalized knowledge representation and support decision-making process by validation operations. An intelligent method for knowledge representation and validation is proposed. Decision selection processes is validated by constraint dependency rules, propagation and delete cascade operations. Two of the common drawbacks of variability modeling (dead decision and logical inconsistency) are validated using detective rules. In [5] state “developing and using a mathematical model in a DSS, a decision maker can overcome many knowledge-based errors. For this reason, the proposed method is supported by FOL rules. We plan to test and validate this work using real data real life case studies from industry. In addition, for representation a more definition data about each choice is recommended.

9. References