K-DSS: A Decision Support System for Identifying and Evaluating Crucial Knowledge

Inès Saad, Camille Rosenthal-Sabroux, Salem Chakhar
LAMSADE, University of Paris Dauphine
Place du Maréchal Lattre de Tassigny
75775 Paris Cedex 16, Paris, France
{saad, sabroux, chakhar}@lamsade.dauphine.fr

Abstract

The objective of this paper is to introduce K-DSS, a decision support system for identifying and evaluating crucial knowledge. K-DSS is an implementation of a two phases-based methodology. Attention is especially devoted to present conceptual and functional architectures of K-DSS. The implementation of K-DSS is also addressed.

Keywords: Decision support system, Knowledge representation, Knowledge management, Crucial Knowledge.

1 Introduction

To optimize the capitalizing on knowledge operation, one should focalize on only the so called “crucial knowledge”, that is, the most valuable knowledge. In practice, decision makers use tacit and explicit knowledge available in various forms in the organization to select, from a set of options, the alternative(s) that better response(s) to organization objectives. The main objective of capitalizing is to extract tacit knowledge [6], that are not explicitly defined and which are considered crucial for improving decision making [1]. Thus, companies should invest in engineering methods and tools in order to preserve the knowledge, especially of tacit nature, related to the decision making process. K-DSS, a decision support system for identifying and evaluating crucial knowledge, is one of such tools.

2 Methodology

The methodology is composed of two phases. A detailed description of it is available in [8]. The first phase is relative to constructive learning devoted to infer the preference model of the decision makers. Practically, it consists in inferring, through the DRSA (Dominance-based Rough Set Approach) [2] method—which is an extension of rough set theory [7] and which is devoted to multi-criteria sorting problems—of a set of decision rules from some holistic information—in terms of assignment examples—provided by the decision makers. This phase includes also the identification, using GAMETH (Global Analysis METHodology) framework [4], of a set of “knowledge of reference” and their evaluation with respect to a set of criteria.
We have constructed three sub-families of criteria [8]: (i) knowledge vulnerability family that is devoted to measure the risk of knowledge lost and the cost of its (re)creation; (ii) knowledge role family that is used to measure the contribution of knowledge in project objectives. Each criterion of this family corresponds to an objective; and (iii) use duration family that is devoted to measure the use duration of the knowledge basing on the company average and long term objectives. 

To evaluate each knowledge $K_i$ in respect to the each objective $O_j$, we have developed an oriented four levels graph computing model [9]. The first level corresponds to the piece of knowledge $K_i$ to be evaluated. The second level corresponds to processes. The third level corresponds to projects $R_1 \cdots R_{n_2}$; $n_2$ is the number of projects. The last level corresponds to the objective $O_j$. 

Once all knowledge are evaluated, the next step is an iterative procedure permitting to conjointly infer the decision rules. Two decision classes have been defined: $Cl1$: “non crucial knowledge” and $Cl2$: “crucial knowledge”. This procedure is composed of four substeps. Basing on the set of “knowledge of reference” and the decision classes, the first substep looks to determine with each decision maker the assignment of these “knowledge of reference” to the decision classes $Cl1$ and $Cl2$. The second substep permits to infer a set of decision rules for each assignment example determined in the previous substep. The third substep looks to modify the assignment examples or the evaluation with the concerned decision maker. This substep is an iterative one and is devoted to resolve inconsistency problems. Finally, we identify, with the help of the decision makers, a subset of collectively accepted decision rules.

In the second phase, the analyst uses the preference models of the different stakeholders defined in the first phase to assign new knowledge, called “potential crucial knowledge”, to classes $Cl1$ or $Cl2$. More specifically, a multi-criteria classification of “potential crucial knowledge” is performed on the basis of the decision rules that have been collectively identified by the decision makers in the first phase. The generated “potential crucial knowledge” are analyzed and then evaluated against the criteria. Then, they are assigned to one of two decision classes $Cl1$ or $Cl2$.

3 Architecture of K-DSS

K-DSS contains four main components: (i) graphical interface; (ii) model base which is the repository of all the algorithms needed to implement the proposed methodology; (iii) database which is the repository of data and eventually the parameters needed for executing the algorithms; and (iv) knowledge base which is the repository of all the pieces of knowledge represented in terms of facts and rules. All of these components, except the graphical interface, are briefly described here.

3.1 Model base

The model base of K-DSS regroups all the algorithms required to implement the methodology. More specifically, it contains: (i) the algorithms for computing the contribution degrees of the knowledge into the objectives, and (ii) the algorithms used to infer decision rules.

3.1.1 Algorithms for computing contribution degrees

The system contains three algorithms: (i) $Max_{p \in P} Min_{e \in p} Min_{d \in D} v^d(e)$; (ii) $Max_{p \in P} Min_{e \in p} Median_{d \in D} v^d(e)$; and (iii) $Max_{p \in P} Min_{e \in p} Max_{d \in D} v^d(e)$. $P$ is the set of paths from $K_i$ to $O_j$; $p$ is a path from the set of paths $P$; $D = \{d_1, \cdots, d_r\}$ is the set of decision makers; and $v^d(e)$ is the evaluation of the vertex $e$ from path $p$ designing the contribution degree of a knowledge to a process, a process to a project or a project to an objective, according to decision maker $d$.

Incorporating these three algorithms into the system enhances the flexibility of K-DSS by offering to decision makers the possibility to select the most appropriate algorithm. Other algorithms may also be added to the system.
3.1.2 Algorithms for the inference of decision rules

The model base contains two algorithms for decision rules induction. Generally, the induction algorithms permit to produce either (i) a minimal covering set of decision rules, i.e., a subset of non-redundant and complete decision rules as for example the DOMLEM algorithm [3]; or (ii) a set containing all the decision rules as for example the algorithms LEM2 (Learning from Examples Module, version 2; which is a part of the data mining system LERS—Learning from Examples based on Rough Sets [5]) or Explore [10]. Here, we have used the DOMLEM and Explore algorithms. These two algorithms use the rough set theory [7].

3.2 Database

The conceptual schema of the database is shown in Figure 1. The central class in the model is the class “Knowledge”. It is described with an unique number (K-Num), a name (K-Name), a description (K-Description), eight attributes (Complexity-Level, Substitutability-Level, Validation-Level, Transferability-level, Rarety-Level, Acquisition-Cost, Production-Time, Accessibility-Level) corresponding to the eight criteria $g_1,\ldots, g_8$ composing knowledge vulnerability family, use duration (Use-Duration) corresponding to the only criterion, $g_{15}$, of use duration family, (Knowledge-Type). Note finally that a piece of knowledge may be composed of several elementary knowledge. This is enhanced with the aggregation relation defined on the class “Knowledge”. The classes “Explicit-Knowledge” and “Tacit-Knowledge” are specializations of the class “Knowledge”. The “Explicit-Knowledge” class permits to identify for each explicit knowledge the set of supports on which this knowledge is inscribed. If the knowledge is tacit, it is characterized with the person who gathers it. This information is deduced from the relationship “Gathers-By” between “Tacit-Knowledge” and “Actor”. The class “Actor” contains the information relative to the different actors (Name, Telephone, Email, Role, Service-Length). The class “Actor” is specialized into three classes: “Supplier”, “Collaborator” and “PSA Actor”. The three classes “Process”, “Project” and “Objective” permit to handle the information relative to the names and descriptions of processes, projects and objectives, respectively.
The association class “Evaluate-K-P” between “Actor”, “Process” and “Knowledge” stores the contribution degree of a knowledge into a process (Contribution-Degree-K-P) attributed by a given actor.

As shown in Figure 1, an actor evaluates zero, one or many knowledge regarding one or many processes. The association classes “Evaluate-P-R” between the classes “Process”, “Project” and “Actor”; and “Evaluate-R-O” between the classes “Objective”, “Project” and “Actor” store the contribution degrees “Contribution-Degree-P-R” and “Contribution-Degree-R-O”, given by an actor to measure the contribution of a process into a project; and of a project into an objective, respectively.

Equally, an actor evaluates one or many processes according to one or many projects. Similarly, it evaluates one or several projects according to one or several objectives. For a given project and a given process, it exists zero, one or several evaluations provided by zero, one or several actors. This is also true for a given project and a given objective. The association class “Decision” contains the decision given by an actor concerning a given knowledge. An actor assigns one or several knowledge to the classes $Cl_1$ or $Cl_2$. A given knowledge can not be assigned to different categories for the same decision maker. Due the fact that the same knowledge may be evaluated by different actors, the creation of class “Decision” is necessary.

### 3.3 Knowledge base

To construct the knowledge base, we have used JESS. Since we are interested only with crucial knowledge, the rules base contains only the rules permitting to assign with certainty “potential crucial knowledge” to the class “$Cl_2$: crucial knowledge”. This because in our application only two classes have been defined and the rules relative to the class “crucial knowledge” will be redundant. However, if several classes have been defined, we should maintain all the rules.

A rule in JESS is defined through the function defrule. An example relative to our application is given in Figure 2. The fact base contains the initial facts relative to knowledge of reference issued from the decision table. A fact in JESS is defined through the function defacts. Figure 3 gives a JESS definition of a fact relative to the application.

```jess
(defrule rule1
  (Knowledge (K-Num ?K)
   (K-Description ?KD)
   (K-Name ?KN)
   (K-Description ?KD)
   (Complexity-Level ?CL)
   (Substitutability-Level ?SL)
   (Validation-Level ?VL)
   (Transferability-level ?TL)
   (Accessibility-Level ?AL)
   (Rarety-Level ?RL)
   (Acquisition-Cost ?AC)
   (Acquisition-Time ?PT)
   (Use-Duration ?UD)
   (> (printout outfile "crucial knowledge")) )
)
Figure 2: An example of a rule definition

(defacts knowledge
  (Knowledge (K-Num K1)
   (K-Name "knowledge relative to additive dosage")
   (K-Description )
   (Complexity-Level complex)
   (Substitutability-Level substitutable)
   (Validation-Level experimental)
   (Transferability-level hardly transferable)
   (Accessibility-Level easy)
   (Rarety-Level rare)
   (Acquisition-Cost low)
   (Acquisition-Time high)
   (Use-Duration high)
}
Figure 3: An example of a fact definition

### 4 Functional architecture of K-DSS

Figure 4 describes the functional architecture of K-DSS. Two phases may be distinguished in this figure. The first phase is relative to the construction of the preference model. The preference model is represented in terms of decision rules. The second phase concerns
the classification of potential crucial knowledge by using the rules collectively identified in the first phase.

4.1 Phase 1. Construction of the preference model

The first step consists in identifying, from the ones proposed, an algorithm for computing the contribution degrees. The selection is collectively established by all the decision makers with the help of the analyst. Whatever the selected algorithm, it uses the matrices Knowledge-Process (K-P), Process-pRoject (P-R) and pRoject-Objective (R-O) extracted from the database—more specifically from the three association classes “Evaluate-K-P”, “Evaluate-P-R ” and “Evaluate-R-O ”—to compute the contribution degree of each piece of knowledge into each objective. To avoid data redundancy, these matrices are not explicitly stored in the database but generated during processing. Only their intentional definitions are permanently stored in the system. Once these matrices are generated, the contribution degrees are first stored (temporally) in a decision table and then introduced in the database. The structure of the decision table is shown in Figure 5. As for matrices, only the intentional definition of the decision table is maintained in the system. The decision table contains the evaluation of the “knowledge of reference” concerning the vulnerabilities and use duration criteria extracted from the database (from the class “Knowledge” precisely). These evaluations are collectively defined and introduced by the analyst in the database. The analyst should introduce in the decision table, and for each decision maker \( k \), the decisions concerning the assignment of “knowledge of reference” into the classes \( C1 \) and \( C2 \). The decision table contains, in addition to the columns relative to vulnerability and those relative to contribution degree and use duration criteria, as many columns as decision makers. Once the decision table is generated, it will be used as the input of the induction algorithm selected by the decision makers (DOMLEM or Explore). This algorithm permits to generate the list of the initial rules for each decision maker \( k \). It is important to mention again that only rules relative to class \( C2 \) are stored. Then, each decision maker should select a subset from these initial rules. The next step in this phase consists to collectively select, from the set of decision rules individually identified by the different decision makers, a subset of decision rules that will be used latter by JESS for the classification phase.

4.2 Phase 2. Evaluation of potential crucial knowledge

The second phase consists in classifying the new knowledge called “potential crucial knowledge”. As the previous one, this phase starts by identifying the algorithm to use to compute the contribution degree of each piece of knowledge into each objective. This algorithm uses as input the information relative to the performances of potential crucial knowledge previously introduced in the matrices K-P, P-R and R-O. The results are stored in a performance table. The structure of the performance table is shown in Figure 6. The information contained in the performance table are then transformed into facts. The inference engine incorporated in JESS verifies first if exists at least one rule that verifies the different facts and if this holds, the knowledge is classified as crucial; otherwise the piece of knowledge is considered non crucial. An update of rule and fact bases any time a fact is verified by at least one rule is performed.

5 Implementation

In this section we provide a brief description a prototype implementing K-DSS. K-DSS was
implemented with Visual Basic. The user can use the different capabilities of GUI interfacing system to, among others, introduce required data, infer decision rules, classify knowledge into $C11$ or $C12$.

Figures 7, 8 and 10 presents three printed screens from K-DSS. The screen in Figure 7 permits to generate Matrix K-P containing the evaluation of each knowledge in respect to each process. As it is shown in this screen, the user selects the piece of knowledge to evaluate and then introduces the evaluation directly or by selecting the desired evaluation from the provided drop-down list. The user may also add/remove a process from the list initially shown. Similar interfaces are used for process-project and project-objectives evaluations. They permit to generate Matrix P-R and Matrix R-O, respectively.

Once all the data are introduced, the user may use the interface shown in Figure 8 to compute the contribution degrees of each knowledge to each objective. First, s/he should select the computing algorithm. As mentioned earlier, three algorithms are provided by the system: (i) $\max_{p \in P} \min_{e \in P} \min_{d \in D} \psi^d(e)$, (ii) $\max_{p \in P} \min_{e \in P} \text{Median}_{d \in D} \psi^d(e)$; and (iii) $\max_{p \in P} \min_{e \in P} \max_{d \in D} \psi^d(e)$.

To infer decision rules, we have used JESS. To incorporate JESS in our system, we have developed an executable file (inference.exe) in JAVA to import JESS DLLs (see Figure 9). K-DSS and JAVA dialogue is completely transparent to users. As shown, in Figure 9, K-DSS automatically generates an input
text file (input.txt) which is used by inference.exe. The results generated by JAVA are then stored, by inference.exe in an output text file (output.txt). This last one is then used by K-DSS to provide results (in terms of decision rules) to the user.

The decision rules are first generated by DOMLEM or Explore. These rules are initially expressed in the following mathematical form:

If \( f(x, g_1) \geq 2 \wedge f(x, g_6) \geq 2 \wedge f(x, g_{12}) \geq 4 \wedge f(x, g_{15}) \geq 2 \)  
Then \( x \in C12 \)

The rules are automatically traduced, by K-DSS, to apply to the syntax of JESS. For example, the rule cited above will be traduced as follows:

IF \( K_i \) Substitutable-Level is “at least weak” and \( K_i \) Rarety-Level is “at least rare” and \( K_i \) Competitivity is “at least high” and \( K_i \) use-duration is at least “average”  
THEN \( K_i \) is at least in C12

This rule means that knowledge \( K_i \) is considered to be crucial (i.e. \( K_i \) belongs to the class C12), if it is difficult to replace it, it is scarce, has an important impact on commercial position of the company and with a convenient use duration.

Once all the decision rules are generated, the user may use the interface shown in Figure 10 to visualize the evaluation of each knowledge in respect to each criteria. Then, s/he should assign these knowledge into classes C11 or C12. Naturally, human may provide some incoherent information when classifying these knowledge. To illustrate this fact, consider knowledge \( K7 \) and \( K8 \) in the list shown in Figure 10 and suppose that \( K7 \) and \( K8 \) have the same evaluation in respect to all criteria. In this case, they should normally be assigned to the same class and not to different classe—as it is shown in Figure 10. It is possible that the evaluation of a knowledge in respect to a given criterion is unavailable. This lack of information is designed by “?” symbol in the screen of Figure 10. This fact was one of many reasons for adopting DRSA instead of several other classification techniques as those based on outranking relation (e.g. Electre Tri), additive utility function (e.g. UTDAS) or hierarchical process discrimination (e.g. MHDIS) or other methods proposed in artificial intelligence where incoherences are eliminated before analysis. In fact, DRSA, which is based on rough set theory, is able to detect incoherence in the decision table, which are latter taken into account in the final decisions and not eliminated in early steps as the case with the above-cited techniques.
6 Enhancing the model with fuzzy logic

To enhance the capabilities of our system, in this section we propose an ongoing work aiming to take into account imprecision and uncertainty at the database level. Fuzzy set seems to be a natural way to cope with this problem. Indeed, class Knowledge may be defined as fuzzy concept with two extent properties: $P_{\text{knowledge}} = \{p_1, p_2\}$ where $p_1$ and $p_2$ are based on level-of-tacit and degree-of-maturity attributes, respectively—these two attributes are not defined in the original model. By associating appropriate weights $w_1$ and $w_2$ to the extent properties $p_1$ and $p_2$, the degree of membership of a piece of knowledge $K_i$ to fuzzy class Knowledge may be computed as follows:

$$
\mu_K(K_i) = \frac{\sum_{i=1}^{n} \rho_{P_{X}^{K}(v_i)} \cdot w_i}{\sum_{i=1}^{n} w_i},
$$

where the number $v_i$ is the value of the attribute of $K_i$ on which the extent property $p_i$ is defined and $\rho_{P_{X}^{K}(\cdot)}$ represents the extent to which entity $K_i$ verifies property $p_i$ of fuzzy class $K$. The idea may easily be generalized to other classes of the model.

7 Conclusion

We have introduced K-DSS, a Decision Support System for identifying and evaluating crucial Knowledge. K-DSS is an implementation of a two phases-based methodology conducted and validated in the PSA Peugeot Citroën automobile company in France. Here, attention is especially devoted to present the conceptual and functional architectures of K-DSS. The implementation of K-DSS is also addressed.

References


