Decision Tree Transformation for Knowledge Warehousing

Rim Ayadi[1,2], Yasser Hachaichi[1,2], Saleh Alshomrani[3] and Jamel Feki[1,3]
[1]*Multimedia, InfoRmation Systems and Advanced, Computing Laboratory, Sakiet Ezzit, Tunisia*
[2]*University of Sfax, Sfax, Tunisia*
[3]*Faculty of Computing and IT, University of Jeddah, Jeddah, Saudi Arabia*

Keywords: Heterogeneous Knowledge, Knowledge Harmonization, Knowledge Warehouse, Data Mining, MOT, Metamodels, Transformation Rules, Source Model, Target Model.

Abstract: Explicit knowledge extracted from data, formalized tacit knowledge from experts or even knowledge existing in business sources may be in several heterogeneous formal representations and structures: as rules, models, functions, etc. However, a knowledge warehouse should solve this structural heterogeneity before storing knowledge. This requires specific tasks of harmonizing. This paper first presents our proposed definition and architecture of a knowledge warehouse, and then presents some languages for knowledge representations as particular the MOT (Modeling with Object Types) language. In addition, we suggest a metamodel for the MOT, and a metamodel for the explicit knowledge obtained using decision trees technique. As we aim to represent knowledge having different modeling formalisms into MOT, as a unified model, then we suggest a set of transformation rules that assure the move from the decision tree source model into the MOT target model. This work is still in progress, it is currently completed with tranformations for additional.

1 INTRODUCTION

In recent years, there has been a growing interest in knowledge both for personal and professional life. Thus, many companies have devoted time and efforts to discover relevant information either on themselves or touching their customers and competitors.

Hence, knowledge is a power means which helps companies to improve their decision making process and to better manage competitive situations. Furthermore, the evolution of the company in its environment, in addition to its external challenges, pushes it to capitalize knowledge in order to be more efficient and then to increase profitability. From this situation was born the need to gather knowledge into a repository commonly called knowledge warehouse (KW). A KW has a specific architecture providing the necessary infrastructure to extract, store and handle explicit knowledge (Nemati et al., 2002).

In fact, explicit knowledge existing in the company and used in decision-making process has initially heterogeneous formats. In order to put different types of knowledge together in the KW, we need to harmonize and homogenize. A means to do this we have elected the usage of the semi-formal graphical language for knowledge modeling called MOT (Modeling with Object Types (Paquette, 2002)). The MOT language relies on a set of *typed knowledge units* and *links* allowing homogeneous modeling of heterogeneous explicit knowledge. Our objective is to represent knowledge having different models from source models (i.e., decision tree, association rules, clustering) into the unified target model which is the MOT. In this paper we focus on the transformation of decision tree knowledge into the MOT model.

The rest of this paper is organized as follows: Section 2 introduces the context of this work and our definition and architecture for a KW. Section 3 presents some languages for knowledge representations and then presents the MOT language and its metamodel. Section 4 proposes the metamodel for models of explicit knowledge obtained by decision tree technique. In addition, it defines transformation rules from this metamodel into the MOT metamodel. Finally, section 5 concludes the paper and gives some relevant perspectives.

2 CONTEXT

For business intelligence purposes, we suggest to gather knowledge (i.e., tacit and explicit knowledge)

Ayadi R., Hachaichi Y., Alshomrani S. and Feki J..
Decision Tree Transformation for Knowledge Warehousing.
DOI: 10.5220/0005380506160623
In *Proceedings of the 17th International Conference on Enterprise Information Systems* (ICEIS-2015), pages 616-623
ISBN: 978-989-758-096-3

into a knowledge warehouse (KW) for which we give our definition and architecture. Our suggestions rely on several works of the literature (Kerschberg, 2001), (Dymond, 2002), (Nemati et al., 2002), (Qing-lan and Zhi-jun, 2009) and (Irfan and uddin Shaikh, 2010) that have tried to introduce the KW concept and propose a basic architecture for a KW. However, these authors did not provide a precise definition of a KW and did not detailed its architecture. Indeed, in their works, a KW is considered as a data warehouse (Dymond, 2002) (Nemati et al., 2002) described with three layers: capture, storage and access to content. For example, (Dymond, 2002) considers that the storage structure is referred to as a knowledge base and is constructed as a tree with objects at the nodes. Objects are packages containing data in "attributes" and blocks of program code in "methods".

In addition, most of proposed architectures are presented in an abstract way and are composed of three layers (Data source layer, Knowledge management layer and Knowledge presentation layer) (Kerschberg, 2001) (Dymond, 2002) (Nemati et al., 2002) (Irfan and uddin Shaikh, 2010). These architectures i) do not present the interaction between decision makers and the KW (i.e., How to exploit and update stored knowledge?) and ii) do not specify the sources of knowledge: are they exist in business sources of companies, are they extracted from existing data or are they acquired through the formulation of tacit knowledge from individuals working in these companies.

Furthermore, in the absence of a complete definition of KW, we rely on the following extracts so that we can give our definition of a KW:

- The knowledge can be found in multiple repositories under multiple heterogeneous representation (Kerschberg, 2001) such as databases, documents, computer programs and even in people's heads (Dymond, 2002);
- The knowledge extracted and stored in the KW should be explicit (Kerschberg, 2001) (Nemati et al., 2002), i.e., formal and systematic in order to be easily communicated and shared (Nonaka and Takeuchi, 1995);
- The KW can be used as a clearinghouse of knowledge to be used throughout the organization by the employees to support their knowledge intensive decision-making activities (Nemati et al., 2002).

Based on these three extracts that we consider incomplete and often reflecting a particular viewpoint, we provide our definition. In our viewpoint, a *Knowledge Warehouse gathers explicit knowledge that may come from multiple sources having heterogeneous formats and relating to several business activities within a domain. This knowledge is unified and integrated in order to support an intelligent decision-making process* (Ayadi et al., 2013).

Based on this definition, we draw multi-layer core architecture for a KW (cf., Figure 1). In this architecture, we find the following tasks located at three layers (Ayadi et al., 2013).

Data Acquisition: This step is interested in collecting the initial data from companies belonging to one or more sectors.

Knowledge Extraction: Extracts the hidden knowledge from the initial data by using multiple knowledge extraction techniques such as data mining techniques (Zaki and Meira, 2014).

Tacit Knowledge Explicitness: This step provides experts with knowledge models to express their tacit knowledge into explicit knowledge in order to be used by a computerized decision process.

Knowledge Harmonization: The harmonization aims to standardize knowledge expressed in heterogeneous formats before being loaded into the KW. It is based on a transformation process that transforms knowledge from a source model into a common target model (i.e., MOT).

Knowledge Storage: Storing knowledge according to a KW model is crucial for the computerized decision process. Naturally, stored knowledge could be later updated by the KW administrator in order to manage knowledge evolution over time.

Knowledge Exploitation: Once the KW is loaded, the usage of its content is the ultimate step for intelligently solving decision problems.

As a further step in the KW life cycle is the KW maintenance. After a decision was taken and evaluated, if the decision maker is satisfied then he validates it, otherwise he can inform the KW administrator to update knowledge that led to invalid decisions.

In the remaining of this paper, we focus on the knowledge harmonization as a keystone task for knowledge warehousing. Knowledge has often several representation forms: tacit knowledge of experts or knowledge extracted through data mining techniques. For this reason, we elected the MOT language as a unified language for knowledge representation.

3 KNOWLEDGE REPRESENTATIONS

In the literature, there are several languages for representing knowledge (Fensel et al., 1994): i) the informal language allows knowledge explicitness in form of sentences, their specifications contain ambiguity and contradictions and lack precision, ii) the formal

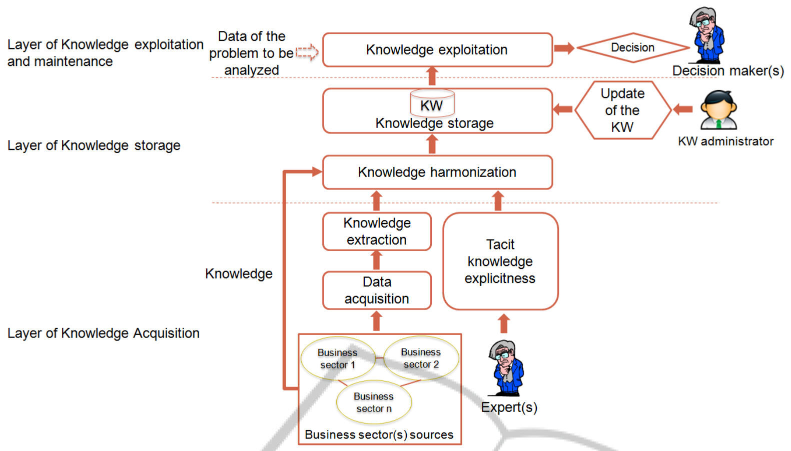


Figure 1: Basic architecture for a knowledge warehouse (Ayadi et al., 2013).

language is a set of strings of symbols that may be constrained by rules that are specific to it. It is used, among other things, for the precise definition of data formats and the syntax of programming languages. These languages are hard to understand and it is very difficult to extract an intuition about the functionality of a system given the huge amount of details of a formal specification only and iii) the semi-formal language is a knowledge expression formalism whose grammatical and semantic flexibility makes it user-friendly (Héon, 2012). It has advantages compared to informal language because it provides a representational guidance which structures the modeling process. On the other hand, compared to the formal language, the semi-formal language offers greater ease of use and offers the possibility of expanding the number of people qualified to represent their knowledge, without the help of knowledge engineers.

Since our work aim to model knowledge for decision makers and due to the advantages of semi-formal languages we choose using a semi-formal language that allows a graphical representation of knowledge and that facilitate the knowledge modeling. In the remainder, we will study the best known semi-formal languages to choose the most appropriate to our work (Paquette, 2002).

Semantic Trees are used to categorize concepts and provide instances for these concepts. The most general concepts are on top of the tree and the most specific concepts are arranged hierarchically below. The upper links indicate a relationship between a class and its subclasses while the lower links are links of belonging of an instance to a class. This type of graphs leaves several possible interpretations since it does not specify the nature of the link.

Concept Maps were developed by Joseph D. Novak in the 70s and they are also used to represent relationships between concepts. They also have the form of a graph, but not necessarily of hierarchical nature. This time, the nature of each link is specified by a label. These maps use non-oriented links that leads to complicate interpretation. Then, the imprecise choice of labels can make it difficult the knowledge transfer.

Semantic Networks were invented by Allan Collins and Ross Quillian in 1969. These networks are graphical representations analogous to concept maps but they use oriented links. Semantic networks use two types of nodes i) nodes representing taxonomic knowledge (concepts and their instances) and ii) others representing the properties of these concepts. Furthermore, they also employ two types of links i) structural links (e.g., is-a, has, part-of) and ii) links specific to the domain semantic. However, these networks complain about the lack of standard terminology and standard graphical representation.

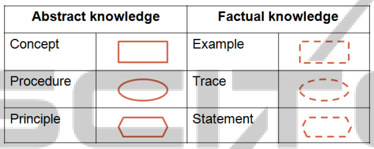
MOT (Modeling with Object Types) Language, designed by Paquette ((Paquette, 2002), (Paquette, 2010)) in collaboration with researchers at the research center LICEF[1], provides a graphical formalism in order to facilitate the representation of explicit knowledge of various formats as well as the articulation of experts' tacit knowledge. The MOT language allows for a quality gain in knowledge representation, since it has greater expressiveness to represent procedural and strategic knowledge other than declarative knowledge (Héon et al., 2010) (Héon, 2012). It enables to represent abstract and factual knowledge as well as the relationship between them using oriented typed links. Thus, MOT is used to represent several types of models: facts systems, taxonomies, procedural systems, methods, processes, multi-agent

[1]Research Center: Laboratory of Cognitive Informatics and training environments (LICEF) of the Quebec Teleuniversity.

systems, etc (Paquette, 2002). Being a language that overcomes problems of the others semi-formal languages, we elected the MOT to represent knowledge in the KW. The next sub-sections describe the alphabet, semantics, grammar and meta-model of MOT.

3.1 The MOT Alphabet, Semantics and Grammar

The MOT alphabet includes two types of graphical symbols for knowledge and relationships (i.e., links). It allows representing knowledge by geometrical figures that differentiate their types and their abstraction levels (cf., figure 2):



Abstract knowledge		Factual knowledge	
Concept	▭	Example	⬚
Procedure	⬭	Trace	⬭
Principle	⬡	Statement	⬡

Figure 2: MOT shapes used for knowledge representation.

- *Abstract Knowledge*: knowledge that represent classes of objects (Paquette, 2002). There are three types of abstract knowledge: Concept, Procedure and Principle;
- *Factual Knowledge (Facts)*: refers to a tangible entity to describe concrete objects (Paquette, 2002) (Héon, 2011). There are three types of facts: Example, Trace and Statement.

From the semantic viewpoint, different types of abstract and factual knowledge are expressed as follows (Paquette, 2002) (Héon, 2012):

- *Concept* (declarative/conceptual knowledge) represents a class of objects. Indeed, it is the abstraction of a concrete object called *Example*;
- *Procedure* (procedural knowledge) describes the sets of operations for acting on the objects. The instantiation of a procedure gives a *Trace*;
- *Principle* (strategic knowledge) makes causal links between objects, determining conditions that may apply to the implementation of actions and representing agents that act on something. Instantiating a principle gives a *Statement*.

Relationships are represented by directional links between knowledge. MOT covers a set of seven typed links (Paquette, 2002) (Héon et al., 2010):

- *Link I*: Instantiation link connects an abstract knowledge to a fact;
- *Link C*: Composition link connects knowledge to one of its components or its constituent parts;
- *Link S*: Specialization link connects two abstract knowledge of the same type, one of which is a-kind-of the other;
- *Link P*: Precedence link associates two procedural knowledge or strategic knowledge such as the first must be completed and evaluated before the second starts;
- *Link I/P*: Input/Product link used to associate a procedural knowledge and a conceptual knowledge in order to represent the input or the product of procedural knowledge;
- *Link R*: Regulation link from strategic knowledge to another knowledge specifies a constraint, restriction or rule that governs knowledge;
- *Link A*: Application link is an association between two domains: the first domain plays the role of meta-knowledge for the second. Thus, the link A allows the combination of a fact to a knowledge.

In addition to rules of good practices, the MOT language offers integrity rules for each type of link (cf., Table 1) (Paquette, 2002) (Héon et al., 2010) (Héon, 2011) . These rules define the valid relationships between different types of knowledge.

3.2 The MOT Metamodel

Recall that our goal is the representation of explicit knowledge expressed in heterogeneous formats into the MOT language; we propose the metamodel of this language (cf., Figure 3) based on its descriptions in (Paquette, 2002). The *MOT model* is composed of *n* *(n≥0) Links* and *m (m≥0) Knowledge unit(s)* (KU). Each *KU* has a name *name_KU* and can be either an *Abstract knowledge unit* (AKU) either a *Factual knowledge unit* (FKU). In turn, an *AKU* can be a *Concept*, *Procedure* or *Principle*. Each *AKU* can be instantiated to many *FKU* which can be an *Example*, *Trace* or *Statement*. In MOT, a *Link* can be of type {I, C, P, S, I/P, R, A}. It links a *KU* source to a *KU* destination. we note that the *Link C* may have a cardinality represented as a value written in the arrow head and which is 1 by default.

To meet our needs and as we aspire that the MOT meta-model covers all types of explicit knowledge, we extend its meta-model by adding two new classes called *Position* and *Principleprocedure* depicted in red color in Figure 3. The class *Position* has the attribute *number* which represents the position of a

[2]The link R between a concept and a principle is an Add in (Héon et al., 2010) (Héon, 2011) (Héon, 2012). It is not part of the original grammar MOT proposed by Paquette (Paquette, 2002).

Table 1: Integrity rules for MOT links.

Destination	Abstract knowledge			Factual knowledge		
Origin	Concept	Procedure	Principle	Example	Trace	Statement
Concept	C, S	I/P	R^2	I, C		
Procedure	I/P	C, S, P	C, P		I, C	
Principle	R	C, R, P	C, S, P, R			I, C
Example	A	A	A	A,C	A,I/P	A
Trace	A	A	A	A, I/P	A, C, P	A, C, P
Statement	A	A	A	A, R	A, C, R, P	A, C, R, P

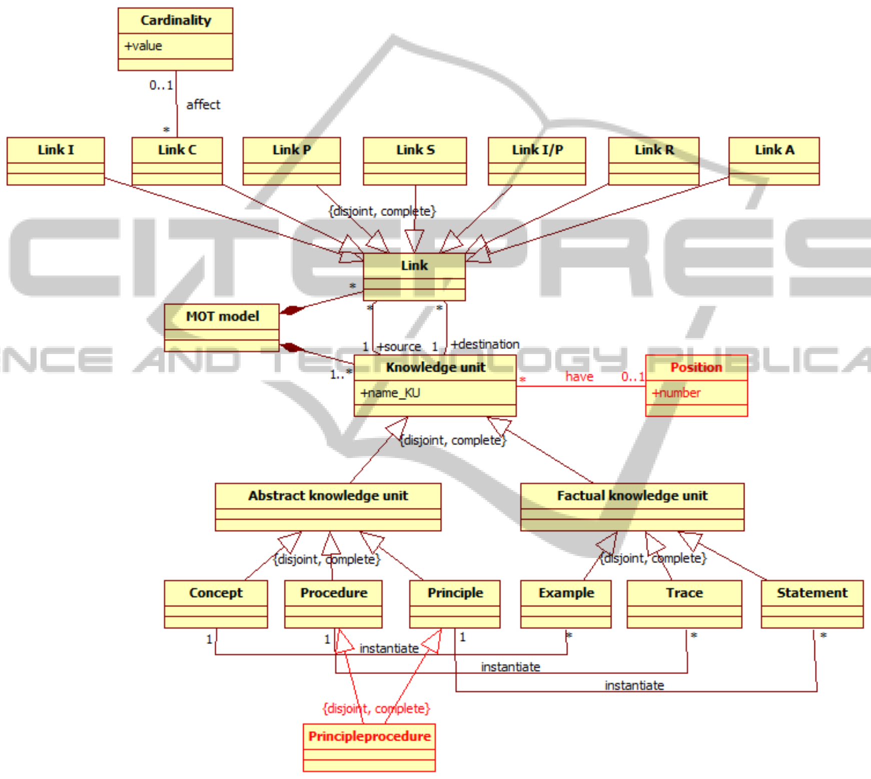


Figure 3: The extended MOT metamodel.

KU. This class is useful when defining transformation rules from the decision tree metamodel where each decision node of the tree has a position (cf., section 4.1). The class *Principleprocedure* represents a *KU* that can be at the same time a Principle and Procedure. This new *AKU* will be useful to transform knowledge from the association rules model into the MOT model. In this MOT metamodel, we have taken into account transformations for three models of the three most used techniques (ie, decision trees, association rules, clustering). Note that we are working on how to integrate knowledge coming from other data mining techniques (e.g., neural networks, support vector machines), therefore, the presented meta-model can be extended by incorporating more concepts (i.e., attributes, classes).

Our objective is to define transformation rules that assure the move from a decision tree source model into the MOT target model. Instead of defining the rules at the model level we position ourselves at a higher abstraction level so then we define rules at the metamodel level (MM); thus we gain genericity of rules. Indeed, our transformation rules associate a concept of the source MM with its corresponding concept in the target MM (Van Sinderen et al., 2012). In the next section we limit ourselves to present transformation rules from the decision tree MM.

4 DECISION TREE TECHNIQUE

In this section we are interested in decision tree as a knowledge source and its transformation into MOT.

4.1 Decision Tree Metamodel

Figure 4 presents the decision tree metamodel (DT-MM). In this MM, a *Decision tree* is composed of *Decision node*(s), *Branch*(es) and *Leaves*. Each *Decision node* has a name *name_DN* and a position *positionN*. Each *Branch* is a path for a possible decision or occurrence and is labeled with a *value*. Each *Leaf* has a position *positionF* and represents the class label of the object to be classified. This class can be of type *Conclusion* or *Action*. Each *Decision node* receives a single *Branch* except the root node, and may have several outgoing *Branches*. Each *Branch* associates with its source one *Decision node* and with its destination either a *Decision node* or a *Leaf*. A *Leaf* is the destination of a single *Branch*.

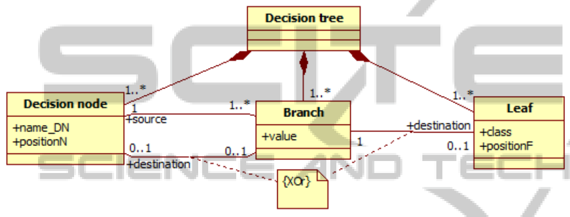


Figure 4: Decision tree metamodel (DT-MM).

Figure 5 shows a sample decision tree built on a set of 14 observations to forecast the behavior of sports described with 5 attributes (Quinlan, 1993), by using the decision tree construction technique in data mining (Althuwaynee et al., 2014). Indeed, the behavior of sports is predicted by determining the Play attribute (represents the leaf of the DT having the yes or no class). This prediction is based on the weather data (Outlook of the sky, Temperature, Humidity and Wind) representing decision nodes of the DT.

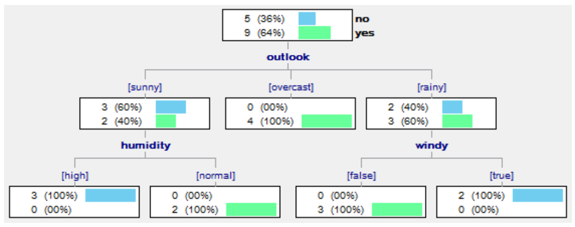


Figure 5: A sample decision tree.

4.2 Transformation Rules from DT to MOT

The transformation rules link between the DT-MM elements and those of the MOT metamodel. Let's note that some elements of the DT-MM transforms to AKU (e.g., principle, procedure) whereas others transforms to links (e.g., link P, link C) between knowledge units. Thus, we distinguish two types of rules.

4.2.1 Transformation Rules to Generate Abstract Knowledge Units (AKU)

In the following transformation rules the term root of a decision tree (DT) refers the Decision node located at position zero (highest level).

RD1. The root *R* of a *DT* becomes an *AKU* of type *Principle Pr1*. The name of *Pr1* is *R*; its position is that of *R*;

RD2. The value of a branch *Br1 (n1, n2)*; i.e., a branch from node *n1* to node *n2*, of a *DT* becomes an *AKU* of type *Principle Pr2*. The name of *Pr2* is the concatenation of *n1* (name of the source node) with the value of *Br1*. The position of *Pr2* is that of the destination node *n2* (decision node or leaf) of *Br1*;

RD3. A leaf *F1* in a *DT* transforms into i) an *AKU* of type *Principle Pr3* when the class of *F1* is a *Conclusion*, and ii) to an *AKU* of type *Procedure Proc* when the class of *F1* is an *Action*. The name of *Pr3* or *Proc* is the class of the leaf *F1*. The position of *Pr3* or *Proc* is *positionF* of the leaf *F1* concatenated with ".1".

Note that in this rule, the type of the resulted *AKU* (i.e., *Principle* or *Procedure*) depends on the semantics of the leaf class *F1* (i.e., *Conclusion* or *Action*).

4.2.2 Rules for Linking AKU

In a decision tree modeled in MOT each *AKU-node* (i.e., *Principle* or *Procedure*) is spotted by its position. The root has position 0 (zero), its immediate descendants are numbered sequentially from left to right including their ancestor number (zero) starting from 0.1. Further descendants are numbered relative to their ancestors as 0.1.2.3 for the third descendant of the second child of the first descendant of the root (cf., Figure 6).

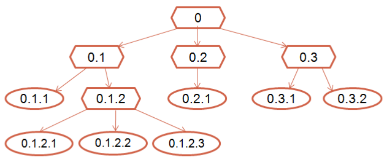


Figure 6: Position of AKU (Abstract Knowledge Unit).

In order to link all *AKU-nodes* obtained by rules RD1 to RD3 we rely on their positions as stated in rule RD4.

RD4. Two AKU *n1* and *n2* having their positions *x* and *x.i* respectively will be related by a link from *n1* to *n2*; where *i* is a single digit. As an example, if *x* is **0.1.2** (position of *n1*) and *n2* position is

0.1.2.4 then a link is constructed starting from *n1* towards *n2*. The type of the link being constructed is determined by the following two rules:

RD4.1. If *n1* is a principle *Pr1* issued from rule RD1 and *n2* is a principle *Pr2* (obtained with rule RD2) then their link is of type *link-C* and has the default cardinality (one);

RD4.2. If *n1* is a principle *Pr2* issued from RD2 and *n2* is an *AKU* issued from rule RD2 or RD3 then the type of their link is *link-P*.

In the next section we develop the DT2MOT algorithm based on the four above rules and we present its application on the example of figure 5.

4.3 The DT2MOT Algorithm

In this section we develop the DT2MOT algorithm that transforms a DT to MOT. This algorithm is not optimized; it shows step-by-step the transformation of a given DT to MOT and indicates in comments of each method the used transformation rule (cf., Algorithm 1). Figure 7 shows AKU obtained from the DT of figure 5 by applying the DT2MOT algorithm from line 3 to line 13. In this figure, the position and the rule of each extracted AKU are depicted.

Rule	List of AKU	Position
RD1	Outlook	0
RD2	Outlook sunny	0.1
	Outlook overcast	0.2
	Outlook rainy	0.3
	Humidity high	0.1.1
	Humidity normal	0.1.2
	Windy false	0.3.1
	Windy true	0.3.2
RD3	yes	0.2.1
	no	0.1.1.1
	yes	0.1.2.1
	yes	0.3.1.1
	no	0.3.2.1

Figure 7: AKU and their positions (obtained from tree of figure 5).

Figure 8 shows the MOT model obtained from the DT of figure 5. In this model, AKU of figure 7 are linked by applying the DT2MOT algorithm from line 14 to line 19.

Algorithm 1: DT2MOT algorithm.

```
   Input: Decision Tree DT
   Output: MOT model
1  Declaration: j ∈ N*, i is a position of a KU
   conform to notation used in RD4
2  begin
3      foreach decision node dn ∈ DT do
4          if (positionN of dn = 0) then
               /* node located at position
                  zero is the Root                    */
5              Transform_root_to_principle (dn,
                  Pr1) /* this method
                  corresponds to rule RD1
                  where dn and Pr1 are
                  respectively the input and
                  the output                          */
6              Break;
7      foreach branch br ∈ DT do
8          Transform_branch_to_principle (br, Pr2)
           /* rule RD2; br: input and
              Pr2: output                             */
9      foreach leaf f ∈ DT do
10         if (class of f is a Conclusion) then
11             Transform_leaf_to_principle (f, Pr3)
                /* rule RD3.i; f: input
                   and Pr3: output                    */
12         else
               /* class of f is Action                */
13             Transform_leaf_to_procedure (f,
                  Proc) /* rule RD3.ii; f:
                  input and Proc: output
                  */
       /* Note that the calls for the four
          previous methods could be
          executed in parallel, they
          generate AKU with their
          positions                                   */
       /* Linking the AKU task                        */
14     foreach AKU n1 do
15         foreach AKU n2 ≠ n1 do
16             if (position of n1 = 0 and position of
               n2 = 0.j) then
17                 Link_by_linkC (n1, n2) /* rule
                      RD4.1; n1: source of
                      linkC and n2:
                      destination of linkC */
18             else if (position of n1 = i and
               position of n2 = i.j) then
19                 Link_by_linkP (n1, n2) /* rule
                      RD4.2; n1: source
                      and n2: destination
                      */
```

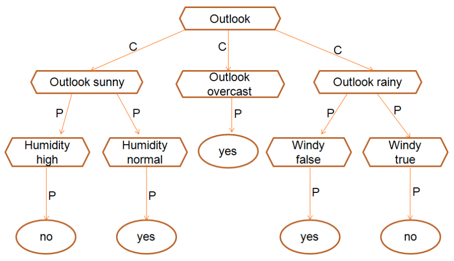


Figure 8: Decision tree of figure 5 transformed into MOT.

5 CONCLUSION AND PERSPECTIVES

In order to load explicit knowledge in the knowledge Warehouse (KW) we need to unify their modeling and solve structural heterogeneity. To represent knowledge having different models into a unified one, we have elected the semi-formal language for graphical knowledge representation called Modeling with Object Types (MOT) for which we have designed its metamodel. In addition, we have defined in this paper the Decision Tree (DT) metamodel and we have suggested a set of appropriate rules that allows transforming DT model into MOT model. Furthermore, we have developed an algorithm describing the principle of these transformations that we have illustrated with an example.

We are working on several extensions: how to transform into MOT language, other kinds of extracted knowledge by using other techniques such as associations rules, ECA rules, non-monotonic knowledge, clustering, neural networks, support vector machines, etc. Then, how to integrate these many MOT models produced from different source models (e.g., Decision tree, association rules, clustering) and what are the operators that should be used to query and maintain an MOT knowledge Warehouse. Using the KW during an intelligent decision-making process is the ultimate objective of this research.

REFERENCES

Althuwaynee, O., Pradhan, B., Park, H.-J., and Lee, J. (2014). A novel ensemble decision tree-based chi-squared automatic interaction detection (chaid) and multivariate logistic regression models in landslide susceptibility mapping. *Landslides*, 11(6):1063–1078.

Ayadi, R., Hachaichi, Y., and Feki, J. (2013). Vers des entrepôts de connaissances : Définition et architecture. In *Conférence sur les Avancées des Systèmes Décisionnels ASD'2013*.

Dymond, A. (2002). The knowledge warehouse: The next step beyond the data warehouse. In *Data Warehousing and Enterprise Solutions*. SAS Users Group International 27.

Fensel, D., Landes, D., Neubert, S., and Studer, R. (1994). Integrating semiformal and formal methods in knowledge-based systems development. In *Proceedings of the 3rd Japanese Knowledge Acquisition Workshop JKAW'94 (Hatoyama, Japan, Nov. 7-9)*, pages 73–87.

Héon, M. (2011). Guide du langage de modélisation par objets typés mot. Cotechnoe inc, Québec, Canada.

Héon, M. (2012). Ontocase, une approche d'élicitation semi-formelle graphique et son outil logiciel pour la construction d'une ontologie de domaine. In *5ième Gestion des connaissances dans les sociétés et les organisations*, Montréal (Québec), Canada.

Héon, M., Basque, J., and Paquette, G. (2010). Validation de la sémantique d'un modèle semi-formel de connaissances avec ontocase. In *Acte des 21èmes Journées Francophones d'Ingénierie des Connaissances*, page 55 à 66, Nîmes, France.

Irfan, R. and uddin Shaikh, M. (2010). Enhance knowledge management process for group decision making. In *Proceedings of the 2010 Second International Conference on Computer Engineering and Applications - Volume 01*, ICCEA '10, pages 66–70. IEEE Computer Society.

Kerschberg, L. (2001). Knowledge management in heterogeneous data warehouse environments. In *International Conference on Data Warehousing and Knowledge Discovery*, pages 1–10. Springer.

Nemati, H. R., Steiger, D. M., Iyer, L. S., and Herschel, R. T. (2002). Knowledge warehouse: an architectural integration of knowledge management, decision support, artificial intelligence and data warehousing. *Decision Support Systems*, 33(2):143 – 161.

Nonaka, I. and Takeuchi, H. (1995). *The Knowledge-Creating Company: How Japanese Companies Create the Dynamics of Innovation*. Oxford University Press, New York.

Paquette, G. (2002). *Modélisation des Connaissances et des Compétences: Un Langage Graphique Pour Concevoir et Apprendre*. Presses de l'Université du Québec, Sainte-Foy.

Paquette, G. (2010). *Visual Knowledge Modeling for Semantic Web Technologies: Models and Ontologies*. IGI Global, Hershey, PA.

Qing-lan, H. and Zhi-jun, H. (2009). Research on cost control dss based on knowledge warehouse. In *Proceedings of the 2009 Sixth International Conference on Fuzzy Systems and Knowledge Discovery - Volume 07*, pages 357–361. IEEE Computer Society.

Quinlan, J. R. (1993). *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Van Sinderen, M., Johnson, P., Xu, X., and Doumeingts, G. (2012). *Enterprise Interoperability: 4th International IFIP Working Conference, IWEI 2012, Harbin, China, September 6-7, 2012. Proceedings*. Springer.

Zaki, M. J. and Meira, W. (2014). *Data Mining and Analysis: Fundamental Concepts and Algorithms*. Cambridge University Press.