Semantic Decision Tables: Self-organizing and Reorganizable Decision Tables

Yan Tang[1], Robert Meersman[1], and Jan Vanthienen[2]

[1] Semantic Technology and Application Research Laboratory (STARLab),
Department of Computer Science,
Vrije Universiteit Brussel, Pleinlaan 2 B-1050 Brussels, Belgium
{yan.tang,robert.meersman}@vub.ac.be
[2] Katholieke Universiteit Leuven, Faculty of Business and Economics
Department of Decision Sciences and Information Management
Naamsestraat 69, 3000 LEUVEN Belgium
jan.vanthienen@econ.kuleuven.be

Abstract. A Semantic Decision Table (SDT) provides a means to *capture* and *examine* decision makers' concepts, as well as a tool for *refining* their decision knowledge and facilitating *knowledge sharing* in a *scalable* manner. One challenge SDT faces is to organize decision resources represented in a tabular format based on the user's needs at different levels. It is important to make it *self organized* and automatically *reorganized* when the requirements are updated. This paper describes the ongoing research on SDT and its tool that supports the self organizations and automatic reorganization of decision tables. We argue that *simplicity*, *precision*, and *flexibility* are the key issues to respond to the paper challenge. We propose a novel combination of the principles of Decision Support and Database Modeling, together with the modern technologies in Ontology Engineering, in the adaptive **s**elf-**o**rganization and **a**utomatic **r**eorganization procedures (SOAR).

1 Introduction

Sharing decision resources efficiently is mandatory for group decision making. The problems of *ambiguity*, *inconsistency* and *scalability*, which occur while drawing a decision table amongst a decision group, are tackled by Semantic Decision Table (SDT, [20]). SDT provides a means to *capture* and *examine* decision makers' concepts, as well as a tool for *refining* their decision knowledge and facilitating *knowledge sharing* in a *scalable* manner. An SDT is the result of annotating a set of decision tables (or any well structured decision resources) with ontologies. It contains *richer* decision rules than a mere decision table, as it specifies the hidden decision rules and meta-decision rules of a decision table. We guide a decision group to construct an SDT using an efficient stepwise methodology described in [19]. Note that the term "decision table" used in this paper is a table that contains decision rules, which can be an incomplete rule set.

In our current research projects, such as the EC Prolix project[1], SDT is used as a tool embedded in decision processes of a system in order to improve its flexibility and effectiveness, such as in [21]. An important feasibility provided by SDT is to visualize the results in the form of decision tables when the decisions are taken in every micro process. Recently, we get increasing requirements of managing the decision tables at a high level; enabling them *self organized* and automatically *reorganized* when the user queries are updated. We consider this kind of decision tables as an extension to SDT. It needs to automatically check the dependencies of different knowledge blocks, quickly adapt to dynamic inputs, and accurately generate the decision tables. These requirements become the challenges of this paper.

A traditional decision table takes the form of a 'flat' reasoning structure with three basic constituents [3]. One constituent is the *condition stubs* and *action stubs*; the second one holds the *condition entries* and the *action entries*; the third one includes the *decision rules*, each of which corresponds to a combination of the elements in the above two constituents. The only constituent type used for reasoning is the third one, which is physically represented as the table columns.

SDT, in general, also contains these three constituent parts. In addition, SDT provides three types of sub-elements for reasoning: 1) the one that corresponds to the dependencies between the conditions (or between the actions); 2) the hidden decision rules, constraints or operational dependencies between the conditions and the actions; and 3) the (possible) meta-rules of a set of decision tables.

The three extra elements of SDT are the key approaches to the paper challenges. In this paper, we propose a novel combination of the principles of Decision Support and Database Modeling, together with the modern technologies in Ontology Engineering, in the procedures called SOAR. SOAR is the abbreviation of the collection of the adaptive Self-Organization and Automatic Reorganization procedures for SDT. In this paper, we propose to use the principles of data dependencies in Database Modeling to constrain the output of SOAR, and thus improve its *precision*.

In our early paper [19], we are careful to stress that SDT, as a sort of group decision support system, has a natural connection with ontology engineering. Ontologies [6, 7], in modern computer science realm, are used to model a domain so far as a universe of discourse. An ontology, by definition, is supposed to be *consistent*. Seeing the reasoning feasibility provided by modern ontology engineering, we store the decision rules at different levels, including the meta-decision rules and other constraints, as a set of *axioms* in an ontology. In this paper, SOAR contains the checksum of the ontological constraints before generating the outputs. By doing so, we can ascertain that its outputs are consistent. We argue that *simplicity*, *precision*, and *flexibility* are the key issues to respond to the algorithm.

The approach of considering SDTs as self-organizing and reorganizable decision tables is based on the characteristics of semantics stored in SDTs. SDT is defined as a decision table with appropriate semantics, containing the constraints at different level, as well as a system that supports data learning. The remainder of this paper is structured as follows. In section 0, we present a grounded understanding of Semantic

[1] The objective of PROLIX is to align learning with business processes in order to enable organizations to faster improve the competencies of their employees according to continuous changes of business requirements. URL: http://www.prolixproject.org/

Decision Tables (SDTs, section 0) and the paper motivation (section 0). We design the procedures in SOAR in section0. SOAR checks whether all the constraints represented by an SDT are satisfied before the outputs are generated. It also provides the outputs at different levels. Section 0 details the main constraints used for SDT. An SDT tool called "SDT SOAR Plug-in" that supports SOAR is demonstrated in section 0. We present our experimental analysis in section 0. We compare our work with the existing technologies, and discuss both the advantages and disadvantages of our work in section 0. Section 0 contains the paper conclusion and the future work.

2 Background

Based on the de-facto standard [3], there are three basic elements in a decision table: the *condition*s, the *action*s (or decisions), and the *rule*s that describe which actions might be taken based on the combination of the conditions. A condition is described by a *condition stub* and a *condition entry*. A condition stub contains a statement of a condition. Each condition entry indicates the relationship between the various conditions in the condition stub. An action (or decision) contains an *action stub* and an *action entry*. Each action stub has a statement of what action to be taken. The action entries specify whether (or in what order) the action is to be performed for the combination of the conditions that are actually met in the rule column.

Table 1. A simple example of a traditional decision table[2], which is used to decide whether we hire a driver or not

	1	2	3	…
Condition				
Has driver's license	Yes	Yes	Yes	…
Previous job	Bus driver	N/A	N/A	…
Language	French, Dutch	French	English	…
Action				
Hire	*			…
Hire and train		*		…

Table 1 presents a part of a simple decision table with three conditions: "Driver's license type", "Previous job" and "Language"; and two actions: "Hire" and "Hire and train". The condition "Has driver's license" has two condition entries - "Yes (the person has a driver's license)" and "No (the person doesn't have a driver's license)". The rule column with ID '1' expresses a decision rule as "If one person has a driver's license, his previous job is a bus driver and he speaks French and Dutch, then hire him".

[2] A traditional decision table is often used as a complete set of decision rules in computer science, e.g. decision tables as a programming tool [2]. Strictly speaking, if Table 1 only contains three decision columns, it is not called a traditional decision table. Rather, it is a table consists of three decision rules.

2.1 SDT: Semantic Decision Table

The notion of Semantic Decision Table (SDT, [20]) was initially introduced to tackle the following problems in a traditional decision table: 1) *ambiguity* in the information representation of the condition stubs or action stubs, 2) *conceptual duplication* amongst the conditions, 3) *uncertainty* in the condition entries, and 4) difficulties in managing *large* tables (also known as the *scalability* problem). What makes an SDT different from a traditional decision table is its *semantics*. Unlike traditional decision tables, the concepts, variables and decision rules are explicitly defined.

An SDT is modeled in three-layer format: 1) the layer of the decision *binary fact types* called SDT *lexons*, 2) the SDT commitment layer that contains the constraints and axioms of these fact types; and 3) the layer of decision tasks or applications. The three-layer format is designed based on the principles of Developing Ontology-Grounded Methods and Applications approach to ontology engineering (DOGMA, [17]), which has been the main research topic at the VUB STARLab over ten years.

An SDT *lexon* is a quintuple $<\gamma, t_1, r_1, r_2, t_2>$, where γ is a context identifier. γ is assumed to point to a resource, and serves to disambiguate the terms t_1, t_2 into the intended concepts. r_1, r_2, which are "meaningful" in this specific context γ, are the roles referring to the relationships that the concepts share with respect to one another. For example, a lexon <γ*, driver, has, is issued to, driver's license*>[3] explains a fact that "a driver has a driver's license", and "a driver's license is issued to a driver". The linguistic nature of a lexon represents that a fundamental DOGMA characteristic is its grounding in the linguistic representation of knowledge. The community of decision makers chooses (or has to agree on) a given (natural) language, e.g. English, to store and present lexon terms and roles.

An SDT *commitment* corresponds to an explicit instance of an intentional interpretation by a decision task. It contains a set of rules in a given syntax, and describes a particular application view of reality, such as the use by the application of the (meta-) lexons in the lexon base. The commitments need to be expressed in a *commitment language* that can be easily interpreted. Suppose that the above lexon - <*driver, has, is issued to, driver's license*> - has the constraint as "*EACH driver should have AT LEAST ONE driver's license*". We apply the *mandatory* constraint on the lexon written as below:

```
P1 = [driver, has, is issued to, driver's license]:    (1)
MAND (p1).4
```

The decision rules in a decision table can be equivalently mapped into a set of SDT commitments. For example, the following commitment is the decision rule in column 1 of Table 1.

In this use case, SDT is the result of annotating a decision table with ontologies. The goal of using SDT is to tackle the problems, such as the ambiguity problem

[3] In this paper, we do not focus on the discussion of the context identifier γ, which is omitted in other lexons. E.g. < γ*, driver, has, is issued to, driver's license*> is thus written as <*driver, has, is issued to, driver's license*>.

[4] The syntax can be found at: http://www.starlab.vub.ac.be/website/SDT.commitment.example

```
(P2 = [Has driver's license, has, is of, value],
 P3 = [Previous job, has, is of, value],
 P4 = [Language, has, is of, value],                    (2)
 P5 = [action, is about, is a, Hire])
 : IMP5 (AND (P2 (value) = 'Yes', P3 (value) = 'Bus
driver', P4 (value) = 'French, Dutch'), P5).
```

and the conceptual duplication problem, early discussed in this section. During the an notation process, the decision makers need to specify all the hidden rules, such as "*EACH driver should have AT LEAST ONE driver's license*" shown above. Thus, an SDT contains *richer* decision rules than a mere decision table.

There are many other interesting use cases of SDT. One of them is to embed SDT in a process, separate decision rules from the process in order to improve the system flexibility [21]. An important feasibility provided by SDT is to visualize the process results in the form of decision tables when the decisions are taken in every micro process. A detailed explanation is given in the next subsection.

2.2 A Use Case of SDT and Motivation

Suppose we have a training process in the domain of human resource management. We want to train the employees from different companies, e.g. MIVB[6]. Firstly, we collect the data of the employees from the company. The data can be personal information or professional background, e.g. the name, the address and the driving skills. Then, we decide which courses he should take. The decisions are drawn based on many decision rules, which can be modeled and embedded in various approaches, such as business process models[7] (BPM, [16]).

This use case is a simplified one we encounter in the EC Prolix project. We are motivated to use SDT because of its advantages. An SDT is a subtype of a decision table. It has all the advantages of a decision table. E.g. a decision table is extremely convenient and user-friendly for non-technical people. It can be easily imported to their workbench, such as Excel[8]. An SDT is a decision table enhanced by semantics, which makes it better than a mere decision table. As early discussed at the beginning of section 0, SDT has many advantages over a decision table. For example, an SDT doesn't contain any ambiguities in the decision items, as they are properly annotated with ontologies. We refer to [19, 20, 21] for more details.

In addition, we're motivated to use SDT because of the feasibility of building SDTs. We build an SDT with the method in [20], which requires domain ontologies.

[5] IMP is the implication operator. AND is the conjunction operator. This SDT commitment is verbalized as: IF the value of 'Has driver's license' is 'Yes', AND the value of 'Previous job' is 'Bus driver', AND the value of 'Language' is 'French, Dutch', THEN the action is about (to) 'Hire'".

[6] It is a public transport company in Belgium. http://www.mivb.be

[7] Nevertheless, the decision rules are separated from the processes.

[8] Excel is part of the Microsoft® Work Suit, which are widely used by many enterprises. An Excel file contains a spreadsheet, which is used to design informal decisions. http://office.microsoft.com/en-us/excel/

In the Prolix project, one training center, such as GENO[9] in the project, is responsible for training the employees from many companies. Different companies can have different database systems and applications; therefore, the domain ontologies are constructed to improve the *interoperability*. We can use the available ontologies to build an SDT.

Recently, we get detailed requirements as follows:

- The first (and probably the most important) requirement is to automatically check the constraints, quickly adapt to dynamic inputs, and accurately (re-)generate the SDT. As soon as a user adds a new constraint, the SDTs earlier generated should be rechecked.
- The second requirement is to present SDT at different levels when needed. A user may have a question on a specific decision in an SDT. For example, he wants the explanation of the first column in Table 1. He sees the formal SDT commitments bundled with the SDT. Unfortunately, he is not familiar with the syntax of the commitments. In a worse case, he even didn't contribute to the SDT commitment writing in the decision group. A simple solution is to provide the verbalization of the SDT commitments in a natural language. For example, "*EACH driver should have AT LEAST ONE driver's license*" is the verbalization of the commitment *P1 = [driver, has, is issued to, driver's license]: MAND (p1)*. The user is happy when there are only a few sentences. He gets nervous when he sees a big bunch of text. Therefore, we need a better solution to categorize the information. In practice, we observe that SDTs are often layered. One decision rule presented in a SDT can be propagated in another SDT. It gives us a hint to present SDT at different levels. Whenever a user wants to know a specific detail level, the system needs to automatically generate another SDT at required level.

The above requirements are the paper challenge and the main motivation: SDT needs to be *self organized* and automatically *reorganized* when the user queries are updated. We have been working on a tool called SDT Plug-in[10] for more than two years. The plug-in implements many user scenarios of SDT, such as the SDT annotation scenario demonstrated in [20]. In this paper, we focus on the above requirements, design a collection of the adaptive *self organized* and automatically *reorganized* procedures (SOAR), which is introduced in section 0 and implemented in section 0.

3 SOAR

SOAR is a collection of the adaptive ***S***elf-***O***rganization and ***A***utomatic ***R***eorganization procedures used for SDT. Fig. 1 shows the pseudo code for three main procedures.

[9] http://www.geno-stuttgart.de/

[10] It is a Java plug-in used in the DOGMA Studio Workbench, which is an ontology engineering tool developed by VUB STARLab. It collects the implementations of all the researching efforts at the lab, e.g. the implementation of the ontology creation methodologies, domain ontology modeling and visualization. A detailed explanation of DOGMA Studio Workbench and the plug-ins can be found at: http://www.starlab.vub.ac.be/website/tools.

```
Generate_1(condition stub[], action
stub[], SQL query){
  load data from database based on SQL
query;
  generate key rows in decision table DT;
  complete DT;
  load ontology constraints set ONT[];
  load SDT commitments SDTC[];
  while (consistent(DT, SDTC[], ONT[][])
        is false){
     user edits SDTC[];
     generate_1(condition stub[], action
stub[]);
  }
  generate SDT;
}
```

(a) Pseudo code for generating SDT from the database and user defined table layout

```
Generate_2 (action, column ID){
  load   relevant ontological constraints
set ONT[];
  load relevant SDT commitment set
SDTC[] ;
  while (SDT of next level exists){
    visualize SDT of next level;
    visualize ONT[];
    generate the verbalization;
  }
}
```

(c) Pseudo code for generating SDT of all levels

```
Boolean consistent(DT, SDTC[],
ONT[][]){
  for all constraints in ONT[]{
     if(DT satisfies ONT[]){
        for all constraints in SDTC[]{
          DT satisfies SDTC[];
          return true;
        }
     }
  }else
     return false;
}
```

(b) Pseudo code for the SDT consistency checking

```
reorganize(SDT, commitment[]){
  load ontology constraints set ONT[];
  for all constraints in commitment[]{
     if (database is not consistent
with new
          constraint){
        propose to delete this con-
straint;
        user deletes the constraint;
     }
  }
  while(consistent(SDT, commitment[],
ONT[][])
        is false){
    delete inconsistent SDT column;
  }
  generate SDT;
}
```

(d) Pseudo code for reorganizing an existing SDT when new commitments are added

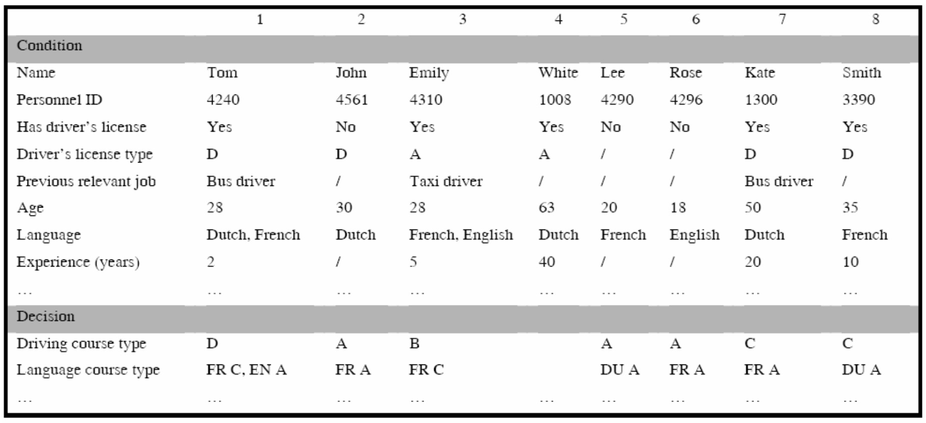
Fig. 1. Pseudo code for SOAR

We explain Fig. 1 as follows:

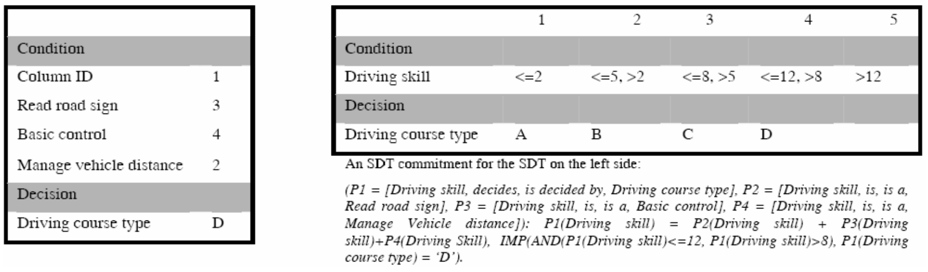
- The procedure generate_1 () is executed to generate SDT from the data stored in the database. First, users need to provide condition stubs (e.g. “Name” and “Has driver’s license” in Table 2) and action stubs (e.g. “Driving course type” and “Language course type” in Table 2) for the table layout, which are the input of the procedure. Second, users need to provide at least one key condition stub of the table. For example, “Personnel ID” is the key condition stub in Table 2. Other data is automatically filled in Table 2 by looking up in the database system. This process is based on the unique key and the foreign keys in DB models. Third, users need to provide an SQL query to select a few records from the database. In a big company, the database can be rather big; therefore, we need the select query to ensure the size of the generated SDT under control. For example, we select our data in department X for Table 2. Fourth, when a temporary generated SDT is inconsistent with the ontologies, users need to edit[11] the SDT commitments, which are often predefined and stored as a set of business rules.

[11] Based on the requirement in practice, users are not allowed to change the ontologies, but they can override the ontological constraints in the SDT commitments.

Table 2. A decision table that decides which training courses are suitable for an employee in department X



	1	2	3	4	5	6	7	8
Condition								
Name	Tom	John	Emily	White	Lee	Rose	Kate	Smith
Personnel ID	4240	4561	4310	1008	4290	4296	1300	3390
Has driver's license	Yes	No	Yes	Yes	No	No	Yes	Yes
Driver's license type	D	D	A	A	/	/	D	D
Previous relevant job	Bus driver	/	Taxi driver	/	/	/	Bus driver	/
Age	28	30	28	63	20	18	50	35
Language	Dutch, French	Dutch	French, English	Dutch	French	English	Dutch	French
Experience (years)	2	/	5	40	/	/	20	10
…	…	…	…	…	…	…	…	…
Decision								
Driving course type	D	A	B		A	A	C	C
Language course type	FR C, EN A	FR A	FR C		DU A	FR A	FR A	DU A
…	…	…	…	…	…	…	…	…



Condition	
Column ID	1
Read road sign	3
Basic control	4
Manage vehicle distance	2
Decision	
Driving course type	D

	1	2	3	4	5
Condition					
Driving skill	<=2	<=5, >2	<=8, >5	<=12, >8	>12
Decision					
Driving course type	A	B	C	D	

An SDT commitment for the SDT on the left side:

(P1 = [Driving skill, decides, is decided by, Driving course type], P2 = [Driving skill, is, is a, Read road sign], P3 = [Driving skill, is, is a, Basic control], P4 = [Driving skill, is, is a, Manage Vehicle distance]): P1(Driving skill) = P2(Driving skill) + P3(Driving skill)+P4(Driving Skill), IMP(AND(P1(Driving skill)<=12, P1(Driving skill)>8), P1(Driving course type) = 'D').

Fig. 2. Two SDTs that shows a decision rule at two different levels

- The procedure reorganize () is executed when a user adds new commitments to an existing SDT. The system first checks the consistency of the existing database with the commitment set. It proposes to the user to delete this constraint when there is a conflict. For example, if the user wants to add a commitment as "*each* user has *at least one* previous relevant job". The existing database may not satisfy his mandatory constraint (see the "Language" data that is automatically filled in the condition entries in Table 2). The solution proposed in this procedure is to delete[12] this constraint in the commitment set. Then, the system checks the consistency of the commitment set with the existing ontology. Users need to edit the commitment set when the conflicts happen.
- The procedures generate_2 () is executed when a user wants to visualize a decision rule of all levels. First, a user provides an action/decision stub in an existing SDT, e.g. "Driving course type" in Table 2, and a column number of an SDT, e.g. column "1" in Table 2. Then, the system loads all the relevant SDT commitments and ontological constraints. An SDT commitment or an

[12] In practice, it costs too much if users change the database system in a company just for one SDT. Therefore, they are required to delete the constraint in SDT commitments when the conflicts happen.

ontological constraint is relevant when it contains this action. Then, the system finds a set of condition stubs needed by this action. It generates another SDT by filling the actual data, which are retrieved from the database system, in the conditions. This process is repeated until no more SDT can be generated (see two SDTs in Fig. 2). The SDT on the left side explains the decision rule with column ID “1” in Table 2 (see the case of “Tom”). It shows all the relevant conditions for “Driving course type D”, such as “(the ability to) Read road sign”. The SDT on the right side (Fig. 2) shows a more general decision rule about “Driving course type”. Relevant SDT commitments are listed. In the meanwhile, necessary verbalizations of the SDT commitments and ontological constraints are generated. For example, the SDT commitment in Fig. 2 is verbalized as: the value of “Driving skill” is the total number of “Read road sign (skill level)”, “Basic control (skill level)” and “Manage vehicle distance (skill level)”; if the value of “Driving skill” is less than or equal to 12, and it is larger than 8, then the value of “Driving course type” is “D”.

All the procedures in SOAR contain the consistency checking. We have defined 22 SDT constraint types in 6 categories. In the next subsection, we explain how to check the consistency of a few SDT constraints, which are mostly used in SOAR.

3.1 Constraints in Semantic Decision Tables

A traditional decision table takes the form of a ‘flat’ reasoning structure represented by three basic constituents. The first one contains the *stubs* of conditions and actions/decisions (e.g. “Has driver’s license” is a condition stub in Table 2. “Driving course type” is a decision stub); the second one holds the *entries* of the conditions and actions/decisions (e.g. “Yes” is a condition entry for the condition “Has driver’s license” in Table 2. “D” is a decision entry for the decision “Driving course type”); the third one includes the *decision rules*, each of which corresponds to a combination of the elements in the above two constituents. The only constituent used for reasoning is the third one, which is physically represented as the table columns (e.g. column 1 in Table 2).

SDT, in general, also contains these three constituent types. In addition, SDT provides three sub-element types for reasoning[13]:

1) The one that corresponds to the dependencies between the conditions (or between the actions). For example, the condition of “Experience (years)” partly depends on the condition “Previous relevant job” in Table 2.
2) The one that represents the hidden decision rules, constraints or operational dependencies between the conditions and the actions. For example, the rule “if a person is about to be retired, then he doesn’t need to be trained” can be specified for Table 2.
3) The (possible) meta-rules of a set of decision tables. For instance, we can specify a meta-rule for Table 2 as “if a column doesn’t contain any decisions, then the table should not contain this column[14]”.

[13] Note that all the constraints used for reasoning are stored as SDT commitments.

The three extra elements of SDT contain the main constraints in SOAR. The SDT constraints mostly used in SOAR are the constraints of *dependencies*, such as subset dependencies, and *logical operators*, such as implication.

As discussed in [13], dependencies in the most general sense are constrained relations in database modeling. Among all kinds of dependencies, *multivalued dependencies*, *subset dependencies*, and *mutual dependencies* are the mostly used. Based on the work in [8, 13], we carefully bring the database modeling principles into the ontology engineering and decision engineering. We mainly use multivalued dependencies, equality, subset, exclusion, mandatory, uniqueness and value constraints in [8].

The types of constraints depend on the requirements in practice. According to Halpin, the total number of constraint types, in theory, is infinite [8]. Including ORM, the various constraints among data have been extensively studied in the literature [1, 5, 8]. The specification illustrated in this section can be further translated into first-order-logic. The translation is useful for reasoning.

With regard to SOAR, it takes the knowledge of data in the database into account. Every record has its meaning. In other words, *data* has its *semantics*. It recalls the debates on whether separate data from knowledge or not, which has been carried on for a long time, e.g. in [14]. We argue that every data has its semantics. It is comparable to the fact that the content in a webpage has its meaning in the context of Semantic Web. By doing so, our approach can benefit from the modern technologies of semantics and ontologies. A drawback can be the difficulties at the implementation level.

By now, we have designed the self-organizing and reorganizing procedures and explained main constraints used in SOAR. In the next subsection, a tool that supports SOAR will be demonstrated.

3.2 SDT SOAR: A Tool to Support Self-organizing and Reorganizable Decision Tables

SOAR (Fig. 1) is developed as SDT SOAR Plug-in in DOGMA Studio Workbench 1.0[15]. The Workbench is constructed according to the plug-in architecture in Eclipse[16]. There, plug-ins, being loosely coupled ontology viewing, querying or editing modules support the different ontology engineering activities and new plug-ins continuously emerge. MySQL Server[17] is used as the database management system to store the employee information.

There are five main views in the SDT SOAR Plug-in as indicated in Fig. 3. The top view is the SDT tabular view. The bottom view in the left corner represents a tree

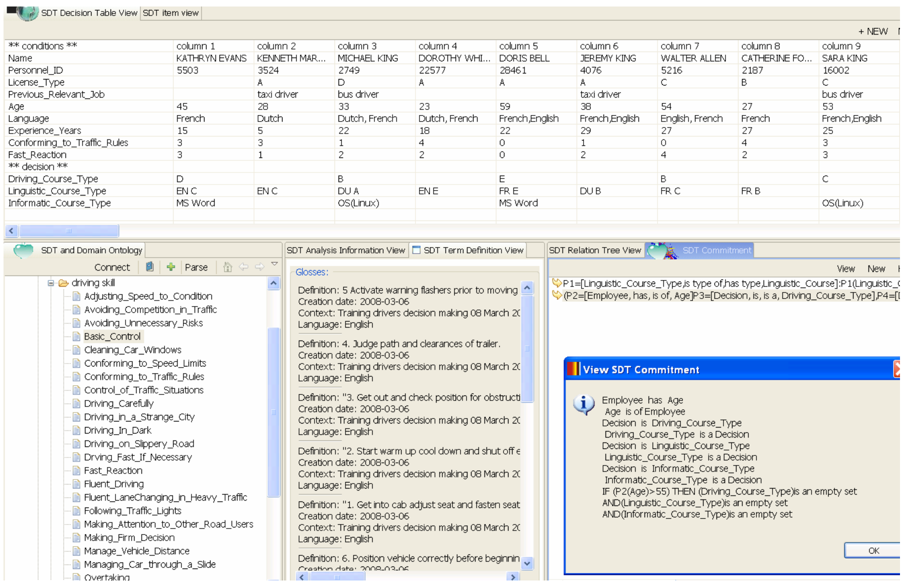
---

[14] It is not necessary to delete such columns in many cases. Otherwise, the debate on the completeness of decision table may arise. However, we put this meta-rule here, because our intension is to use it as an example to demonstrate the meta-rules of a decision table.

[15] DOGMA Studio is a tool suite, which contains both a Workbench and a Server, to support DOGMA ontology engineering approaches. http://www.starlab.vub.ac.be/website/dogmastudio

[16] Eclipse is an open development platform, which supports Java language (http://java.sun.com/). It is mainly used for enterprise development, embedded device development, rich client platform, application frameworks and language IDE. http://www.eclipse.org/

[17] MySQL is a multithreaded, multi-user SQL database management system (DBMS). http://www.mysql.com/



** conditions **	column 1	column 2	column 3	column 4	column 5	column 6	column 7	column 8	column 9
Name	KATHRYN EVANS	KENNETH MAR...	MICHAEL KING	DOROTHY WHI...	DORIS BELL	JEREMY KING	WALTER ALLEN	CATHERINE FO...	SARA KING
Personnel_ID	5503	3524	2749	22577	28461	4076	5216	2187	16002
License_Type		A	D	A	A	A	C	B	C
Previous_Relevant_Job		taxi driver	bus driver			taxi driver			bus driver
Age	45	28	33	23	59	38	54	27	53
Language	French	Dutch	Dutch, French	Dutch, French	French,English	French,English	English, French	French	French,English
Experience_Years	15	5	22	18	22	29	27	27	25
Conforming_to_Traffic_Rules	3	3	1	4	0	1	0	4	3
Fast_Reaction	3	1	2	2	0	2	4	2	3
** decision **									
Driving_Course_Type	D		B		E		B		C
Linguistic_Course_Type	EN C	EN C	DU A	EN E	FR E	DU B	FR C	FR B	
Informatic_Course_Type	MS Word		OS(Linux)		MS Word				OS(Linux)

driving skill
- Adjusting_Speed_to_Condition
- Avoiding_Competition_in_Traffic
- Avoiding_Unnecessary_Risks
- Basic_Control
- Cleaning_Car_Windows
- Conforming_to_Speed_Limits
- Conforming_to_Traffic_Rules
- Control_of_Traffic_Situations
- Driving_Carefully
- Driving_in_a_Strange_City
- Driving_In_Dark
- Driving_on_Slippery_Road
- Drving_Fast_If_Necessary
- Fast_Reaction
- Fluent_Driving
- Fluent_LaneChanging_in_Heavy_Traffic
- Following_Traffic_Lights
- Making_Attention_to_Other_Road_Users
- Making_Firm_Decision
- Manage_Vehicle_Distance
- Managing_Car_through_a_Slide
- Overtaking

Glosses:

Definition: 5 Activate warning flashers prior to moving
Creation date: 2008-03-06
Context: Training drivers decision making 08 March 20
Language: English

Definition: 4. Judge path and clearances of trailer.
Creation date: 2008-03-06
Context: Training drivers decision making 08 March 20
Language: English

Definition: "3. Get out and check position for obstructi
Creation date: 2008-03-06
Context: Training drivers decision making 08 March 20
Language: English

Definition: "2. Start warm up cool down and shut off e
Creation date: 2008-03-06
Context: Training drivers decision making 08 March 20
Language: English

Definition: "1. Get into cab adjust seat and fasten seat
Creation date: 2008-03-06
Context: Training drivers decision making 08 March 20
Language: English

Definition: 6. Position vehicle correctly before beginnin
Creation date: 2008-03-06

```
P1=[Linguistic_Course_Type,is type of,has type,Linguistic_Course]:P1(Linguistic_
(P2=[Employee, has, is of, Age]P3=[Decision, is, is a, Driving_Course_Type],P4=[
```

View SDT Commitment

```
Employee has Age
 Age is of Employee
Decision is Driving_Course_Type
 Driving_Course_Type is a Decision
Decision is Linguistic_Course_Type
 Linguistic_Course_Type is a Decision
Decision is Informatic_Course_Type
 Informatic_Course_Type is a Decision
IF (P2(Age)>55) THEN (Driving_Course_Type)is an empty set
AND(Linguistic_Course_Type)is an empty set
AND(Informatic_Course_Type)is an empty set
```

Fig. 3. SDT SOAR Plug-in screenshot

view of the domain ontologies, e.g. the ontology of HRM of drivers, with which the SDT is built. The bottom view in middle is the concept definitions categorized in glosses. The concept definition view gives the definitions when a concept in the ontology tree is selected (see 'Basic_Control' in Fig. 3).The bottom views in the right corner are the views of formal SDT commitments and SDT commitments in pseudo natural language. Users can add a new commitment and visualize its verbalization. For example, the window with the title "View SDT Commitment" in Fig. 3 shows a new rule "if the age of an employee is more than 55, then he doesn't need to take any courses". SDT-SOAR will automatically check and regenerate the SDTs at different levels, such as shown in Fig. 2, when the new rule is added.

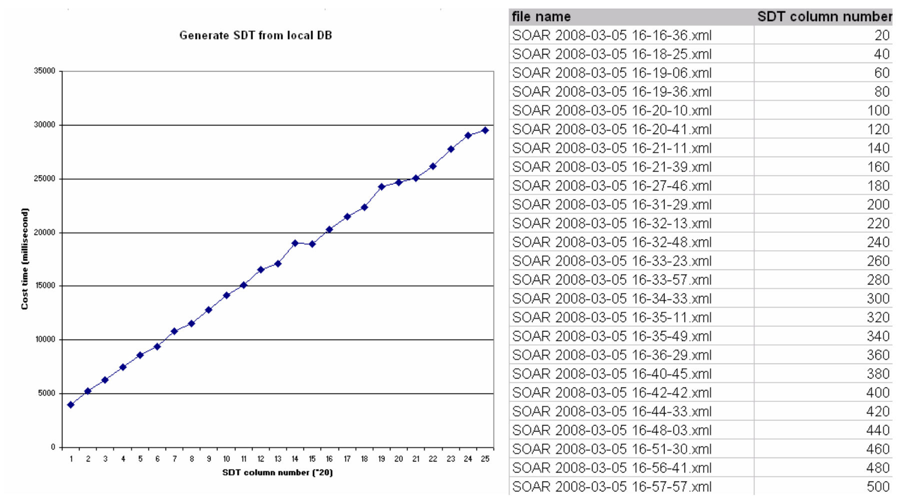
In this section, we focus on how the procedures in SOAR are designed and implemented. In the next section, we present experimental analysis of SOAR.

4 Experimental Analysis

We have conducted several experimentations to evaluate SOAR.

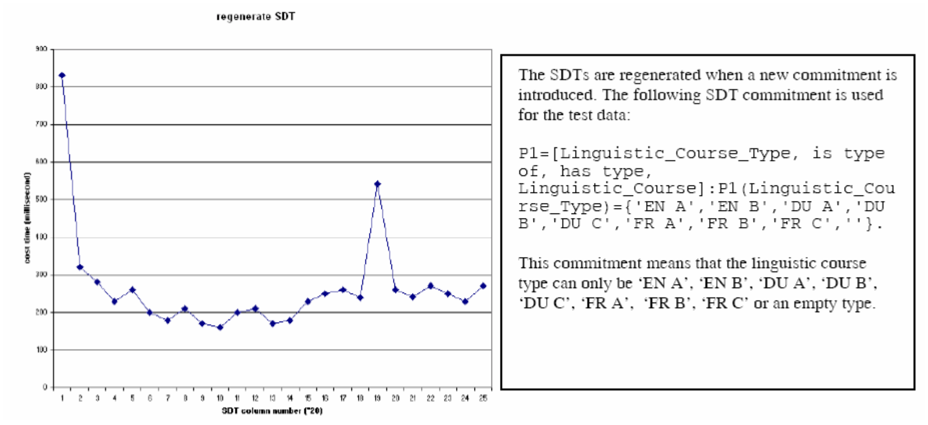
The experimental setup is as follows: We use Intel(R) Pentium(R) processor 1500MHZ with 2 GB memory running Microsoft Windows XP professional version 2002 with Service Pack 2. We implement SDT-SOAR using JRE 1.6.0_02. The employee information is stored in MySQL Sever of version 5.2.

Fig. 4 shows the cost in milliseconds for generating SDTs from the local database. We increase the SDT size by adding its decision columns. In our problem settings, every decision column in an SDT corresponds to an employee. The more employees are selected from the local database, the bigger the resulting SDT becomes. The



file name	SDT column number
SOAR 2008-03-05 16-16-36.xml	20
SOAR 2008-03-05 16-18-25.xml	40
SOAR 2008-03-05 16-19-06.xml	60
SOAR 2008-03-05 16-19-36.xml	80
SOAR 2008-03-05 16-20-10.xml	100
SOAR 2008-03-05 16-20-41.xml	120
SOAR 2008-03-05 16-21-11.xml	140
SOAR 2008-03-05 16-21-39.xml	160
SOAR 2008-03-05 16-27-46.xml	180
SOAR 2008-03-05 16-31-29.xml	200
SOAR 2008-03-05 16-32-13.xml	220
SOAR 2008-03-05 16-32-48.xml	240
SOAR 2008-03-05 16-33-23.xml	260
SOAR 2008-03-05 16-33-57.xml	280
SOAR 2008-03-05 16-34-33.xml	300
SOAR 2008-03-05 16-35-11.xml	320
SOAR 2008-03-05 16-35-49.xml	340
SOAR 2008-03-05 16-36-29.xml	360
SOAR 2008-03-05 16-40-45.xml	380
SOAR 2008-03-05 16-42-42.xml	400
SOAR 2008-03-05 16-44-33.xml	420
SOAR 2008-03-05 16-48-03.xml	440
SOAR 2008-03-05 16-51-30.xml	460
SOAR 2008-03-05 16-56-41.xml	480
SOAR 2008-03-05 16-57-57.xml	500

Fig. 4. Cost of generating SDTs from local DB



The SDTs are regenerated when a new commitment is introduced. The following SDT commitment is used for the test data:

```
P1=[Linguistic_Course_Type, is type
of, has type,
Linguistic_Course]:P1(Linguistic_Cou
rse_Type)={'EN A','EN B','DU A','DU
B','DU C','FR A','FR B','FR C',''}.
```

This commitment means that the linguistic course type can only be 'EN A', 'EN B', 'DU A', 'DU B', 'DU C', 'FR A', 'FR B', 'FR C' or an empty type.

Fig. 5. Cost of regenerating and reorganizing SDTs

generated SDTs are stored as XML files (see the table on the right hand in Fig. 4). The minimum size of the XML file is 18.6 KB (SOAR 2008-03-05 16-16-36.xml), which is generated in 4005 milliseconds. The maximum XML file size is 180 KB (SOAR 2008-03-05 16-57-57.xml) generated in 29533 milliseconds. It increases linearly when the SDT sizes up gradually.

Fig. 5 illustrates the cost in milliseconds while regenerating and reorganizing SDTs. Once a user introduces a new commitment, such as shown in the figure, SDT SOAR checks the consistency in an SDT. The inconsistent decision columns are removed. The cost shown in Fig. 5 has is an irregular line, which means that the cost of regenerating and reorganizing an SDT does not depend on the size of the original SDT.

5 Related Work and Discussion

In the past, decision tables mainly used for computer programming can be found in many literatures, such as [4, 12, 18]. The application area of decision tables have been gradually moved from computer programming to many other domains during the last 50 years. A renewed research interest of decision tables focuses on the construction of the table itself [22]. As Vanthienen indicated, the application field of decision tables is enlarged into knowledge engineering, especially in the contexts of verification and validation of knowledge based systems, efficient execution of knowledge based systems, knowledge base maintenance, knowledge acquisition and knowledge discovery.

The approach of this paper is in the application area of knowledge validation and knowledge discovery. We focus on the discussion of the self organization and automatic reorganization of Semantic Decision Tables (SDTs). A similar solution is SORCER introduced in [10]. SORCER is a learning system that induces *second-order* decision tables from a given data set. Each entry (a condition entry or a decision entry) of a *first-order* decision table corresponds to a single value; while each entry of a *second-order* decision table is a value set. The authors in [10] tend to enhance comprehensibility of a decision tables by transforming a first-order decision table into a second-order decision table. By doing so, they can also reduce the table size without losing the decision rules. Our approach goes further than their work. We call both a first-order decision table and a second-order decision table as 'traditional' decision tables. For example in Fig. 2, the table on the left hand is a first-order decision table and the table on the right hand is a second-order table. The work in [10] is restricted to the transformation of these two kinds of tables. We provide a more generic transformation algorithm described in the SOAR procedures. Moreover, the work in [10] only uses the 'if-then-else' deduction rules for the transformation. We use various constraints, such as the mandatory constraint, the subset constraint and the exclusion constraint[18], for the transformation. Similar debates can be applied to the approaches that are similar to SORCER, such as [9, 11].

Another interesting approach similar to ours is using decision tables in a decision table based development framework of decision support system [22, 23]. Decision tables are automatically created data patterns. We share the same comprehension of that fact that the decision logics behind a decision table are the key issues in the automatic decision table generation. The methods in [22, 23] use various classification techniques while generating the decision tables. For instance, classical neural networks, machine learning and classification tree algorithm. The generating rules of an applied domain are keyword (or label) based. Therefore, the resulting decision tables are not always accurate. In this paper, the semantics of SDT are from both the decision logics and the constraints in the domain ontologies. An ontology, by definition, deals with the concepts and their relations in a domain instead of the keywords. It has been proven that an ontology-based system can dramatically increase the accuracy of a process result, e.g. key words searching versus ontology-based searching [15]. Therefore, we argue that SOAR procedures in this paper, which are ontology based, can increase the accuracy of the generated decision tables.

[18] Note that those constraints are not used at the level of database but at the level of the decision table.

Comparing to all the research efforts of the related work listed above, SDT has many basic yet important characteristics provided by modern ontology engineering. An SDT is the result of annotating (a set of) decision table(s) with a domain ontology. It can be stored in computers (e.g. the SDT xml files shown in Fig. 4) and is *explicit*, *sharable*, *formal* and *conceptual*. Comparing to a traditional decision table, an SDT contains richer decision rules. All the verification and validation rules of a decision table are specified in the form of ontological commitments of SDTs. Based on the constraints in the SDT commitments, SOAR ensures the *precision* of the resulting reorganized SDTs.

In this paper, we use SDTs to learn rules from data and match new rules with existing data. Another simple yet important understanding of SDTs is as follows. An SDT can also be considered as a decision table with appropriate semantics in order to define the decision logic in a modeling setting. In this case, we don't need to involve the database or actual case as we do in this paper.

A disadvantage of SDT might be its dependency on the availability of the domain ontology. According to our experience, to create an ontology costs a lot of time. For example, we used to spend six man months to create a HRM (Human Resource Management) ontology based on O*NET[19] in PoCehrMOM Project[20]. Therefore, SDT is feasible when one of the following conditions is satisfied: 1) there exist domain ontologies; 2) there exists formal knowledge documentations, which can be easily converted into an ontology; 3) the domain is rather small.

6 Conclusion and Future Work

In this paper, we focus on the discussion of Semantic Decision Tables (SDTs) as self-organizing and reorganizable decision tables. SOAR is developed as a collection of the adaptive ***S***elf-***O***rganization and ***A***utomatic ***R***eorganization procedures used for SDT. SOAR is *precise*, *simple* and *flexible*. While reorganizing an SDT, SOAR contains the consistency checking based on the constraints in the SDT commitments. We introduce 7 constraints and 4 logical operators mainly used in formal SDT commitments. SOAR ensures the *precision* of the process of self-organization and reorganization by always satisfying these constraints. In our current projects (e.g. the EC Prolix project), we observe that it's rather easy to implement SOAR because the algorithm used in the SOAR procedures is rather *simple* (see the pseudo code in Fig. 1). The reasoning logics of SDTs are often layered. For example, the SDT on the right hand in Fig. 2 explains the SDT on the left hand in Fig. 2. The latter SDT represents part of the reasoning logics of Table 2. By using SOAR, end users can visualize SDTs at different levels. We call it *visualization flexibility*.

SOAR is implemented as a tool called *SDT-SOAR*. We have conducted several experiments to evaluate SDT-SOAR. The cost of generating an SDT from local database

[19] O*NET provides a full-access, online version of the occupational network database. http://online.onetcenter.org/

[20] PoCehrMOM Project (Project omtrent Competenties en functies in e-HRM voor technologische toepassingen op het Semantisch Web door Ontologie en Meertalige terminologie). The project is to use ontologies to enhance human resource management. http://cvc.ehb.be/PoCeHRMOM/Frameset.htm

server increases linearly when the size of the SDT grows. The cost of regenerating and reorganizing an SDT does not depend on the size of the original SDT.

Currently, the tool SDT-SOAR only supports a few constraints, such as the value constraint. In the future, we will implement all constraints discussed in the paper. One of our recent ongoing researches focuses on using RuleML[21] to store and interchange the SDT commitments. Later on, we will add a new SDT-SOAR function, which reads RuleML as the input and generates SDTs as the output.

Acknowledgments. The research is partly supported by the EC Prolix project. It is authors' pleasure to thank all the STARLab members for the paper discussion.

References

1. Camps Paré, R.: From Ternary Relationship to Relational Tables: A Case against. Common Beliefs, SIGMOD Record 31(20) (2002)
2. Cavouras, J.C.: On the Conversion of Programs to Decision Tables: Method and Objectives. Commun. ACM 17(8), 456–462 (1974)
3. CSA, Z243.1-1970 for Decision Tables, Canadian Standards Association (1970)
4. Geesink, L.H., van Dijk, J.E.M.: The construction of decision tables in PROLOG. Angewandte Informatik archive 30(7), 294–301 (1988)
5. Goelman, D., Song, I.-Y.: Entity-Relationship Modeling Re-revisited. In: Atzeni, P., Chu, W., Lu, H., Zhou, S., Ling, T.-W. (eds.) ER 2004. LNCS, vol. 3288, pp. 43–54. Springer, Heidelberg (2004)
6. Gruber, T.R.: Toward Principles for the Design of Ontologies Used for Knowledge Sharing. In: Workshop on Formal Ontology, Padva, Italy; In book Formal Ontology in Conceptual Analysis and Knowledge Representation. Kluwer Academic Publishers (1993)
7. Guarino, N., Poli, R.: Formal Ontology in Conceptual Analysis and Knowledge Representation. Special issue of the International Journal of Human and Computer Studies 43(5/6) (1995)
8. Halpin, T.: Information Modeling and Relational Database: from Conceptual Analysis to Logical Design. Morgan-Kaufmann, San Francisco (2001)
9. Han, J., Fu, Y.: Discovery of multiple-level association rules from large databases. In: Proc. of the 21st international conference on very large databases (VLDB 1995), Zurich, Switzerland, pp. 420–431. Morgan Kaufman, San Francisco (1995)
10. Hewett, R., Leuchner, J.H.: The Power of Second-Order Decision Tables. In: Proc. of the Second SIAM International Conference on Data Mining, Arlington, VA, USA. SDM 2002, April 11-13, 2002. SIAM, Philadelphia (2002)
11. Kohavi, R.: The Power of Decision Tables. In: Lavrac, N., Wrobel, S. (eds.) Proceedings of the European Conference on Machine Learning. Lecture note in Artificial Intelligence, vol. 914, pp. 174–189. Springer, Heidelberg (1995)
12. Langenwalter, D.F.: Decision tables - an effective programming tool. In: Proc. of the first SIGMINI symposium on Small systems, pp. 77–85. ACM, New York (1978)

[21] The Rule Markup Language (RuleML) is a markup language developed to store rules in XML. The Rule Markup Initiative has taken steps towards defining a shared Rule Markup Language (RuleML), permitting both forward (bottom-up) and backward (top-down) rules in for deduction, rewriting, and further inferential-transformational tasks. http://www.ruleml.org/

13. Sadri, F., Ullman, J.D.: Template dependencies: a large class of dependencies in Relational Databases and its complete approximatization. Journal of the ACM (JACM) 29(2), 363–372 (1982)
14. Sheth, A.: Data Semantics: What, Where and How? Database Applications Semantics. In: Proc. of the Sixth IFIP TC-2 Working Conference on Data Semantics (DS-6), Stone Mountain, Atlanta, Georgia, USA, Chapman & Hall, Boca Raton (1996)
15. Sheth, A.P., Ramakrishnan, C.: Semantic (Web) Technology In Action: Ontology Driven Information Systems for Search, Integration and Analysis. IEEE Data Engineering Bulletin, IEEE Data Engineering 26(4), 40–48 (2003)
16. Smith, H., Fingar, P.: Business Process Management: The Third Wave, 1st edn. Meghan-Kiffer, USA (2002)
17. Spyns, P., Meersman, R., Jarrar, M.: Data Modeling versus Ontology Engineering. SIGMOD Record: Special Issue on Semantic Web and Data Management 31(4), 12–17 (2002)
18. Sterbenz, R.F.: Tabsol decision table preprocessor. ACM SIGPLAN Notices archive 6(8) (September 1971); special issue on decision tables, pp. 33 – 40, B.F. Goodrich Chemical Company, Cleveland, Ohio. ACM, New York (ISSN:0362-1340)
19. Tang, Y.: On Conducting a Decision Group to Construct Semantic Decision Tables. In: Meersman, R., Tari, Z., Herrero, P. (eds.) OTM-WS 2007, Part I. LNCS, vol. 4805, pp. 534–543. Springer, Heidelberg (2007)
20. Tang, Y., Meersman, R.: On constructing semantic decision tables. In: Wagner, R., Revell, N., Pernul, G. (eds.) DEXA 2007. LNCS, vol. 4653, pp. 34–44. Springer, Heidelberg (2007)
21. Tang, Y., Meersman, R.: Organizing Meaning Evolution Supporting Systems Using Semantic Decision Tables. In: Meersman, R., Tari, Z. (eds.) OTM 2007, Part I. LNCS, vol. 4803, pp. 272–284. Springer, Heidelberg (2007)
22. Vanthienen, J.: Ruling the business: about Business Rules, Decision Tables and Intelligent Agents. In: Vandenbulcke, J., Snoeck, M. (eds.) New directions in Software Engineering, pp. 103–120, 160. Leuven University Press, Leuven (2001)
23. Wets, G., Vanthienen, J., Mues, C., Timmermans, H.: Extracting complete and consistent knowledge patterns from data. In: van Harmelen, F. (ed.) Proc. of Sixth International Conference on Principles of Knowledge Representation and Reasoning: V&V Workshop, Trento, Italy (1998) ISSN 1613-0073