

# **Ruling the Business: About Business Rules and Decision Tables**

**Jan Vanthienen**

Department of Applied Economic Sciences  
Katholieke Universiteit Leuven  
Naamsestraat 69, 3000 Leuven, Belgium  
Jan.Vanthienen@econ.kuleuven.ac.be

***Abstract.** Business rules have become a common approach to understanding, specifying and implementing business processes. This new approach adds to the work that has been done in knowledge-based systems technology in such areas as knowledge acquisition, modeling, representation, validation and verification, inferencing, etc. Also, the decision table technique is continuously being used (and rediscovered) in representing and validating the complex business logic to be dealt with in business rules.*

## **1. Introduction**

Some observations continue to plague ICT and information systems: too hard to develop, too hard to implement, and too hard to change. Especially with the rise of the Web, applications and services must be able to react in a fast and flexible way to ever changing business situations, policies and products.

Obtaining this flexibility is a challenge. It is hard to modify or even find the underlying rules of the business in existing systems, because they are frozen and buried in the code. And yet personalization, flexibility and the need to modify business processes by business managers are key issues nowadays.

Systems analysis and design often describe enterprises in terms of the structure of the data those enterprises use, the organization of the functions they perform, the operations that will be executed, but they have tended to neglect the rules and constraints under which the business operates. Frequently these are not explicitly documented, but hidden into program code. Communication between ICT professionals and business experts is difficult when a considerable amount of business knowledge is only represented in code or database constructions and is not modeled in an explicit way which is easy to understand.

## **2. Expliciting the Business Knowledge**

Business rules have become a common topic in systems development and enterprise modeling. Identifying business rules is now an accepted approach to understanding business processes. However, discussions on this topic frequently give little recognition to the amount of work that has been done in knowledge-based systems technology in such areas as knowledge acquisition, modeling, representation, validation and verification, inferencing, etc.

Even a classical technique, like decision tables, is continuously being used (and rediscovered) in representing and validating the complex business logic to be dealt with in the business rules approach.

### **2.1. Modeling Business Knowledge using Tables**

Although the decision table still looks almost the same as in the early days of its first developments, some profound changes can be noted (see also earlier descriptions of the stages in the history of decision tables, e.g. CODASYL (1982)):

- The **application area** has moved from computer programming towards various other domains with logical complexity. This extension has directed research efforts away from the efficient conversion of the table into program code towards the construction process of the table itself. In recent developments, this enlargement of the application field is illustrated through the use of decision tables in business knowledge engineering and validation.
- The **objective** has changed: the emphasis has moved towards the power of the decision table to model complex decision situations in a simple manner, easy to check for consistency, completeness and correctness.

Figure 1 shows the main points of attention in the evolution of decision table research and practice.

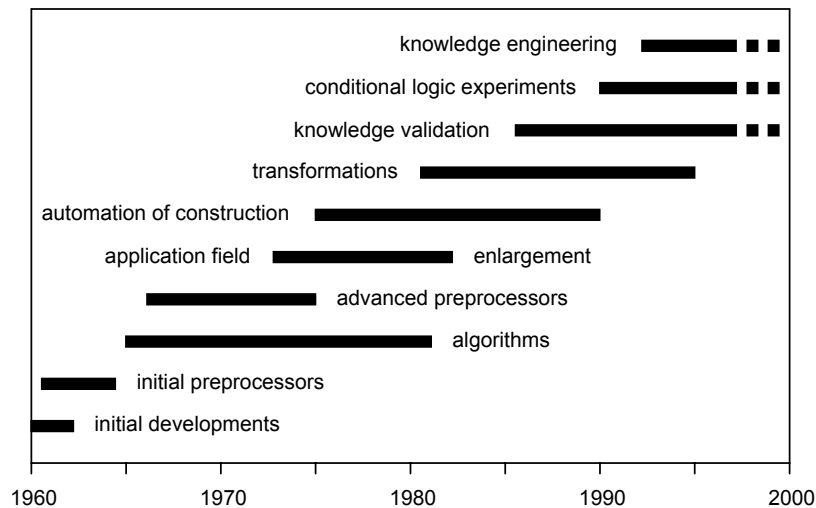


Figure 1. Evolution of the decision table technique

Though originally used as a technique to support programming, decision tables have proven a useful aid in modeling complex decision situations of various sorts. In the field of knowledge-based systems research, there has been a renewed interest in decision tables over the past years. Here, decision tables have been studied or applied in the following contexts:

- verification and validation of knowledge-based systems,
- efficient execution of knowledge-based systems,
- knowledge base maintenance,
- knowledge acquisition,
- knowledge discovery,
- several application domains such as medicine, law, etc.

Based on these observations, it is suggested that much is to be gained from the integrated use of decision table based techniques throughout the various stages in systems development. The focal point of this approach is the use of decision tables as a modeling formalism for expressing (at least part of) the business knowledge in the problem domain at hand.

## 2.2. A Renewed Interest

That the decision table is continuously being used (and rediscovered) in representing and validating complex business logic will show from the continuous attention it gets in the literature. In a recent overview of decision table publications (Moreno Garcia, Verhelle & Vanthienen (2000)), about nine hundred references were discovered and analyzed. However, this figure is far from exhaustive, given the limited availability of some publications. Even electronic search facilities (e.g. on the Web) are not able to find all documents, because not all of them have been indexed. Thus, the actual number of documents discussing decision tables will be much higher.

However, even taking into account the above-mentioned remarks, an effort to capture and classify the most important documents related to decision tables has been performed. This compilation of documents shows that decision tables involve a broad range of problem solving matters going as far as transport or

chemistry. To have a better understanding of the evolution of the literature on decision tables, the references are also classified per year. The following graph (Figure 2) presents the results.

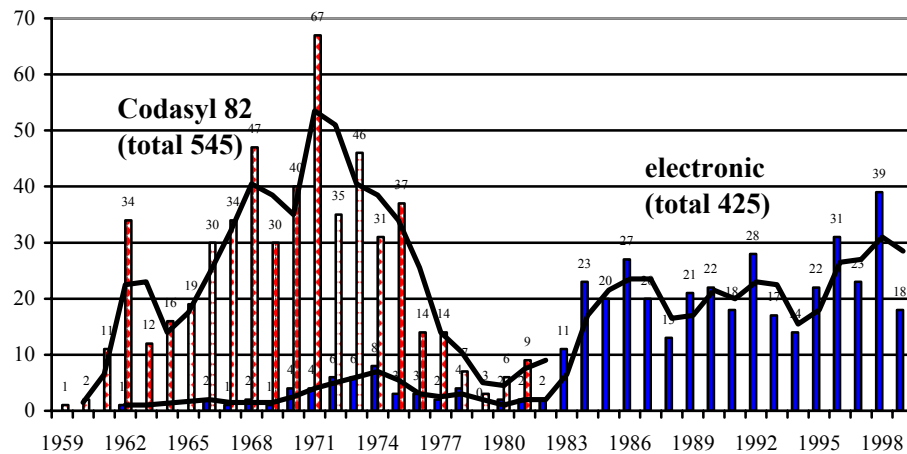


Figure 2. Number of decision table publications per year

Figure 2 shows publications on decision tables throughout the past forty years. The two colors correspond to two kinds of documents, where red (left) points to references that have been extracted from the Codasyl International Decision Table Task Group report of 1982, and blue (right) indicates documents located after an electronic search. The total number of publications referenced is 970, of which 545 related to the Codasyl report and 425 to the electronic research.

In the early years, publications were especially oriented towards decision tables as an interesting tool for programming. The period 1962-75 was very productive with an average of 30 publications per year, and a record of 67 publications in 1971. The number of publications over the years 1984 to 1999 is more or less constant, with an average of around 20 per year. The only exceptions are 1988 and 1994, but these deviations will be coincidental. The smaller number of publications from 1999 on can be explained by the earlier remark that not all references have been indexed yet, which is especially true for new ones. Similar observations, but to a smaller extent hold for 1983 and earlier where publications may not appear in indexing mechanisms because of their age.

In brief, we can confirm that publications are rising and there is evidence to confirm the renewed interest in decision tables in several domains.

### 2.3. The Business Rules Approach

Although traditional systems development methods, like structured analysis, information engineering, and also the newer object oriented analysis methods are an efficient way to model data, processes, entities and operations, they do not always capture the essence of the business in business terms.

Building systems requires flexibility and does not work well without business rules and the ability to create and manage business rules in a well organized setting, by bringing business rules out of obscurity, out of implementation and into the business management side (Morgan (2002), Ross (2003), Von Halle (2002), Date (2000)).

The essence of the business rules approach is to describe and automate aspects of the business function in a declarative instead of a procedural way (WHAT Not HOW, as it is called in Date (2000)). Odell (1995) defines business rules as "declarations of policy or conditions that must be satisfied". The author suggests the following taxonomy of business rules:

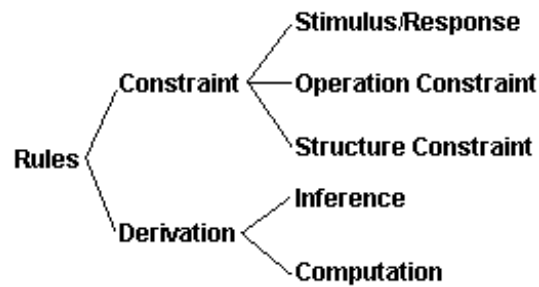


Figure 3. A Taxonomy of Business Rules

This taxonomy maps well to standard information concepts, including OO concepts. For example, operation constraints are pre- and post-conditions on object operations, and structure constraints are class invariants. However, stimulus/response rules, which state that when something occurs, something should be done, are not commonly supported by OO languages and are typically implemented outside of an OO application as triggers in the database. On the other hand, knowledge-based tools provide software support for stimulus/response rules using demons.

Derivation rules derive a fact about an object. Computational business rules state how something is calculated; they are typically allocated to procedural code in object operations. Inference rules may be associated with rules in knowledge-based systems. Object-oriented techniques often ignore inferencing and reduce inference rules to procedural code, thereby violating the declarativeness of business rules.

In order to separate code and data from rules, and add flexibility to systems development, rule engines and rule management systems have to be added. ("Knowledge-based systems did not really go away. They went undercover").

#### 2.4. The New Decision Table

A decision table is a tabular representation used to describe and analyze decision situations, where the state of a number of conditions determines the execution of a set of actions. Not just any representation, however, but one in which all distinct situations are shown as columns in a table, such that every possible case is included in one and only one column (completeness and exclusivity). Many variations of the decision table concept exist which look similar at first sight.

##### 2.4.1. Different Kinds of Tables

The most important criterion when distinguishing tables, is the question whether all columns are mutually exclusive (*single hit* versus *multiple hit*).

In a single hit table each possible combination of conditions matches exactly one (one and only one) column.

If the columns are non-exclusive (multiple hit table), some combinations of conditions match more than one column, which may lead to ambiguity or inconsistency. When consulting the table then, the first hit rule will often be used. This rule states that the *first hit* (when scanning the table from left to right) will determine which set of actions has to be executed, thus preventing contradictions. Another possibility is that *all hits* are used to determine the set of actions to be executed. In this case, each hit from left to right can add actions (not mentioned by previous columns) or overwrite actions. An interesting concept of this latter form is the so called decision grid chart, a tabular representation of a set of (action oriented) decision rules. In both multiple hit cases (first hit versus all hits) the same combination of conditions can occur in different columns. As a result the overview over the columns is lost, and with it, the simplicity of inspection. For these reasons we do not consider these tables to be real decision tables, even though they are frequently mentioned in practice.

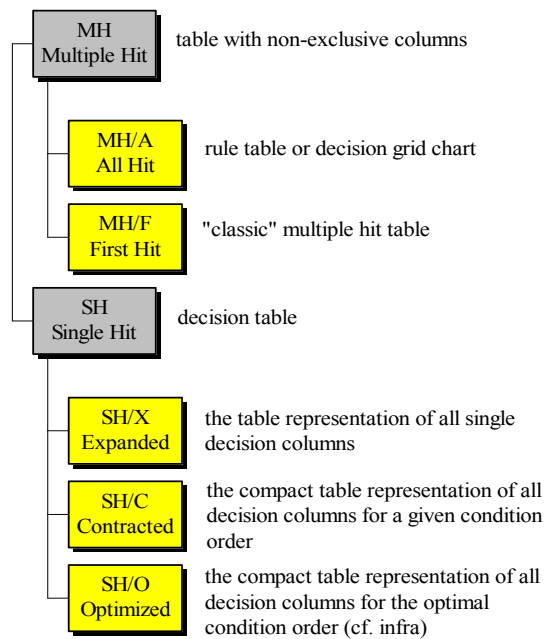


Figure 4. Kinds of tables

Based on these considerations, the subdivision in Figure 4 is put forward. This grouping, however, does not mean that one form is better than the others for all purposes. The decision grid chart (rule base) primarily has a specification function while the expanded single hit decision table has a verification function. When constructing decision tables, we will use both forms in this order, as steps in the process leading to the final contracted table. In this context Maes and Van Dijk (1988) discuss the *life cycle* of the decision table, distinguishing between 'construction time', 'test time' and 'interpretation time' decision tables (corresponding to our types MH/A, SH/X and SH/C respectively).

#### 2.4.2. A Proposed Standardization of Decision Tables

In the past, many variations of the decision table concept have been used, without making a proper distinction between them, thus leading to confusion and reduction of the applicability of the formalism. In order to deal with these problems, we suggest a number of constraints to the decision table formalism. These constraints deal with the form as well as with the contents of the table and emanate from the need to use the decision table as a well structured technique across application areas. The purposes served by this proposed standard have been validated by extensive use of the technique in a large number of environments and applications. **Ten commandments on decision table usage** are proposed (Figure 5):

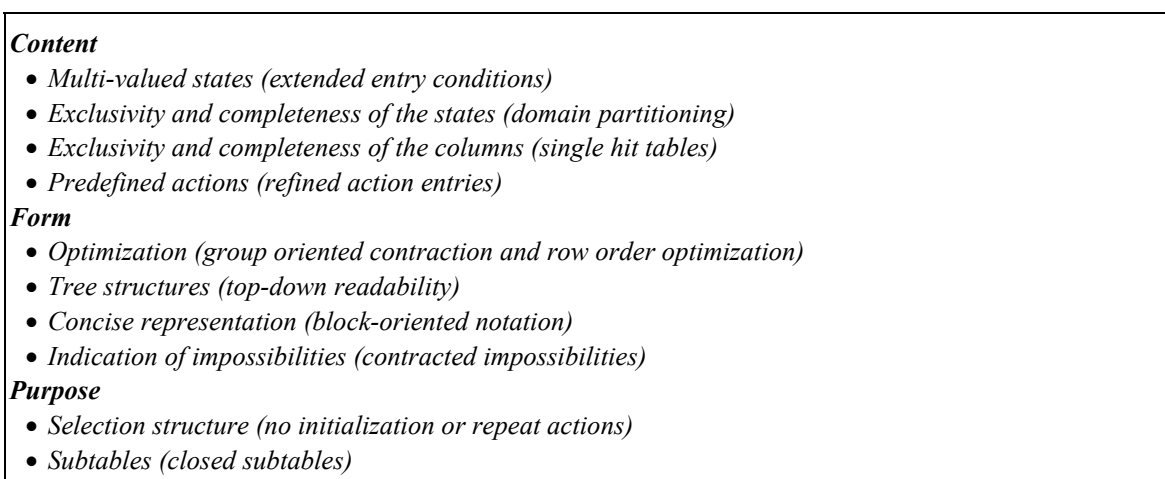


Figure 5. Ten commandments on decision table usage

### 3. The Quality of (Business) Rules

Identifying business rules is becoming an accepted approach to understanding and modeling business processes. It is clear that the quality of the set of business rules is essential and that rules should not duplicate or conflict other rules, and that they are correct, complete and simple.

Little recognition is often given to the amount of work that has been done on this subject in knowledge-based systems. Gathering the correct and complete knowledge is one of the main problems in building knowledge-based systems, and usually a lot of contradictions and insufficiencies have to be detected and solved. Also, maintaining the rule base is not a trivial task and often introduces unnoticed inconsistencies, contradictions or other anomalies. Verification and validation of knowledge based systems have been receiving considerable attention in the past.

Validation and Verification (V&V) occur at several instances during the building process:

- A first validation takes place during the *elicitation* stage. When acquiring knowledge, we will look for incomplete specifications, ambiguities, redundancies in order to direct and improve the elicitation process.
- In the *modeling* stage, the modeling tool or technique will (or should) verify the knowledge structuring effort (not only syntactical elements, but also semantics). Very often, graphical representations of links between elements are used in this stage.
- When the system has been designed, the knowledge has to be *tested* (using prototyping facilities, debugging, test data management, reasoning trees, tracing and logging).
- The V&V process resumes while *maintaining* the knowledge.

#### 3.1. Verification and Validation of Rules

Research in the verification & validation (V&V) community has focused on these anomalies. A classification which is nowadays commonly used in the V&V community of knowledge-based systems can be found in Preece & Shinghal (1994). It considers the following anomalies:

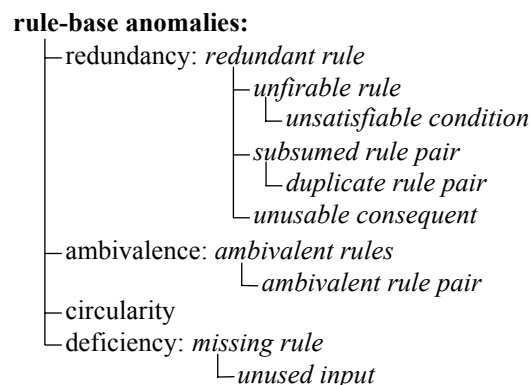


Figure 6. A classification of rule base anomalies

In order to detect these anomalies, many tools have been developed. An important problem of automatic rule base checking is that these tools often face combinatorial explosion as the rule base increases. Therefore, it may be necessary to split the rule base or use heuristics to circumvent this problem. Although anomaly detection tools can be effective, it would be wrong to rely completely on these tools to build high-quality systems. These tools are merely an add-on to the development environment, and mostly they possess no facilities to prevent anomalies during the design process.

The issues of validation and verification have led to the occasional use of schemes, tables or similar techniques in knowledge representation and validation. As has been reported earlier, e.g. in Cragun and Steudel (1987), most of these verification and validation problems (consistency and correctness of knowledge, non-redundancy of knowledge, completeness of knowledge) can benefit from the decision table technique.

Anomaly detection systems based on the decision tables or related formalisms, mainly suffer from the drawback that they fail to find anomalies in the rule base which occur over inference chains. Most of those systems can only detect anomalies between simple pairs of rules. Also most of these systems do not check for circularity. Therefore, in order to overcome these limitations, a different perspective in developing the anomaly detection facilities has been taken. Although in the literature it is argued that approaches which are based on some form of tables show many limitations in order to verify a knowledge-based system, most of these limitations do not hold anymore if the decision tables are properly defined.

Our viewpoint differs from these other approaches, because we advocate the use of decision tables as a modeling technique on its own, and not only as a means towards verification of rule-based systems.

### 3.2. Decision Tables and V&V

It is important that knowledge in specifications is correct, consistent, complete and non-redundant. During and after the building process, the specifications must be verified and validated, which proves to be one of the important concerns in designing business systems.

Starting from the classification by Preece it was investigated how these kinds of anomalies can be detected in a decision table based system (Vanthienen, Mues, et al. (1998)). A distinction is made between tabular anomalies (which might occur in a single table) and inter-tabular anomalies (which originate from interactions between several tables).

The following classification is used to describe (intra-)tabular anomalies. As only single hit tables are allowed, where every possible combination of condition values is represented in one (completeness) and only one column (exclusivity), most of the traditional anomalies cannot even occur or are easy to detect.

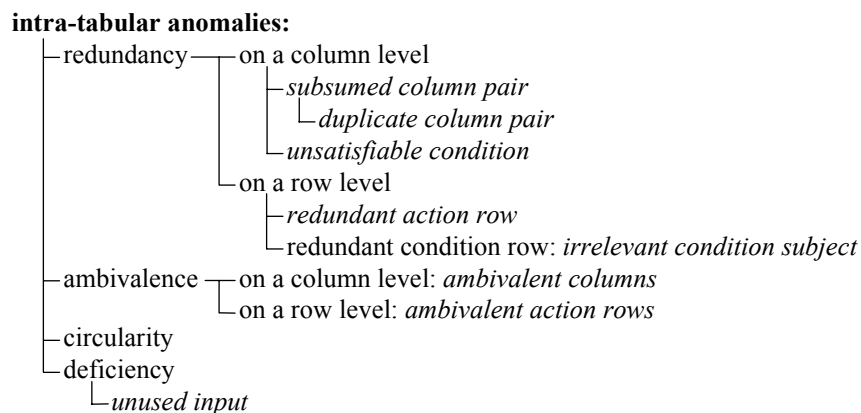


Figure 7. A classification of tabular anomalies

In a vast majority of cases, the decision table technique is able to provide for extensive validation and verification assistance. It easily enables the designer to check for contradictions, inconsistencies, incompleteness, redundancy, etc. in the problem specification. Moreover, the acquisition process is well served through the overview and communication abilities of well-structured decision tables, because they allow to solve (or avoid) the common problems in rule systems, e.g. redundant rules, conflicting rules, subsumed rules, unnecessary conditions, circular rules, missing rules or combinations, unreferenced attribute values, illegal attribute values, dead end clauses.

#### 3.2.1. Consistency and Correctness of Knowledge

Dividing the knowledge over a large number of rules, designed independently, may lead to problems of inconsistency, such as:

- **Conflict:** rules with the same premises (or containing overlapping combinations), but leading to contradictory conclusions.

In a decision table all columns are non-overlapping and each column refers to exactly one configuration of conclusions, therefore conflicting columns will not occur.

- **Cyclical rules:** a set of rules where a conclusion occurs somewhere as one of the premises. In a decision table context, conclusions occurring as premises lead to separate tables. The forward character of the decision table structure eliminates the problem of cyclical references;
- **Invalid attribute values:** a rule containing a nonexistent value of an attribute (e.g. because of a typing error). Once the domain of conditions is well defined, invalid values will not occur, as table entries can be generated automatically.

### 3.2.2. *Non-redundancy of Knowledge*

Redundancy usually does not lead to errors in the final system, although it may harm efficiency. The main problem with redundancy, however, is maintenance and the risk of creating inconsistencies when changing the specifications. Some common forms of redundancy:

- **Subsumption:** rules with the same conclusions but with one of them containing additional premises (and therefore being less general). Subsumption can not occur in the decision table, because columns do not overlap.
- **Redundant premises:** (partly) complementary rules with equal conclusions, which can be combined. In a contracted decision table, two or more complementary rules with equal action configurations are automatically combined, leading to irrelevant or partly irrelevant conditions. The number of distinct columns is thereby minimized.
- **Redundant rules:** rules with the same premises and (partly) equal conclusions. Because in the decision table every possible case is included in only one column (exclusivity), redundancies will not occur.

### 3.2.3. *Completeness of Knowledge*

No current system is able to incorporate all possible knowledge, but within the specific problem area, the following omissions often occur:

- **Missing knowledge:** the absence of some essential elements from the problem situation. Inconsistency often is a symptom of incompleteness because of missing premises.
- **Unused attribute values or combinations:** when attribute values (or combinations) never occur as premises, a number of rules may be missing. Detecting the completeness of all combinations of attribute values is not always simple. The nature of the decision table easily allows to check for completeness: the number of simple columns should equal the product of the number of states for every condition. This guaranty of completeness of condition combinations is one of the main advantages of decision tables.
- **Unreachable conclusions:** conclusions which are never deduced and cannot be asked. The format of the decision table easily shows unreachable conclusions.

Detecting these rule shortcomings (either by the designer or automatically), without some form of decision tables, is highly improbable, unless through excessive testing.

### 3.3. *Inter-tabular Anomalies*

To check a system of interrelated tables or sets of rules on anomalies, it is not sufficient to check each of the components separately. It also has to be checked whether there exist some inter-tabular anomalies. These anomalies can only be detected by inspecting the system as a whole, because these anomalies result

from the interactions between two or more tables. A classification of inter-tabular anomalies is depicted in the following figure:

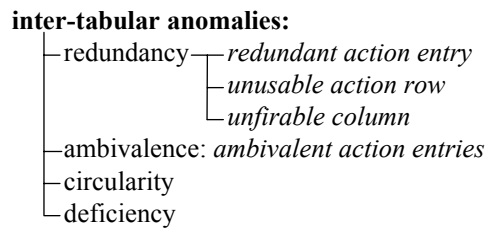


Figure 8. A classification of inter-tabular anomalies

Most of the limitations shown by anomaly checking systems based on decision tables are solved in our approach. These include the possibility to check for anomalies over chains of rules and the check for circularity.

#### 4. Holidays, A simple one-table example

The example is a regulation that is used by the Personnel Department of a small company. As the information is available in written form, capturing the business knowledge in decision tables can start from the initial text. This is often the case for existing regulations in a business environment. The text found in the rule book of the company is depicted in the following figure.

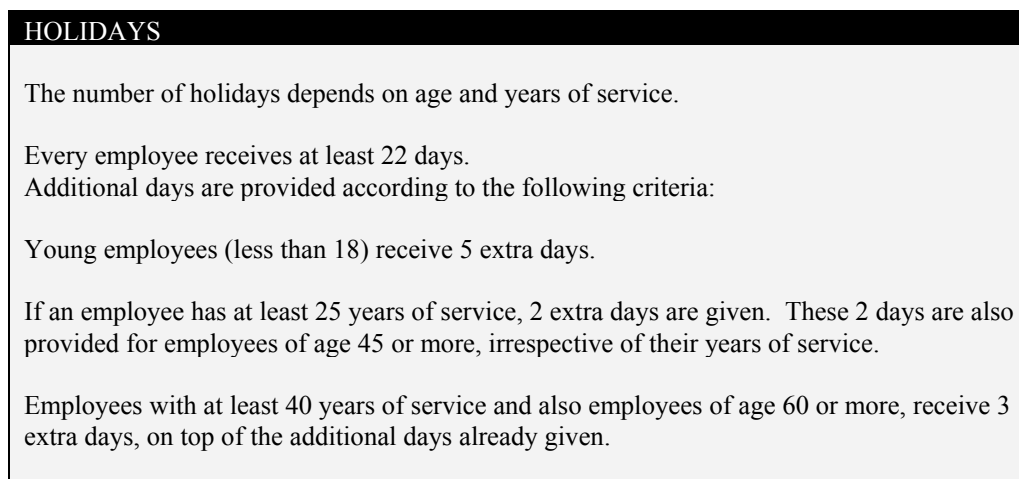


Figure 9. The regulations for holidays, as found in a rule book

We will go through the stages of building a decision table step by step.

##### 4.1. Defining the Concepts

When we draw up a list of all condition statements and actions that are mentioned in the text, it is clear that this example only uses the Age and Years of Service of an Employee in order to calculate the Number of Holidays. The following table lists all occurrences of these terms in the text.

Condition Statements	Action Statements
Age	Number of Holidays
Years of Service	At least 22 days
Every Employee	Additional days
Less than 18	5 extra days
>= 25 years of service	2 extra days
Age >= 45 years	2 extra days
irrespective of years of service	3 extra days on top
>= 40 years of service	
Age >= 60 years	

Table 1. Exhaustive List of Condition Statements and Actions in the text

If we delete the restatements and the complements from the list, and think about the domain of each attribute (completeness!), we obtain an exhaustive set of mutual disjoint states. This is done by putting the different states of a condition on a numerical scale, as can be seen in Figure 10.

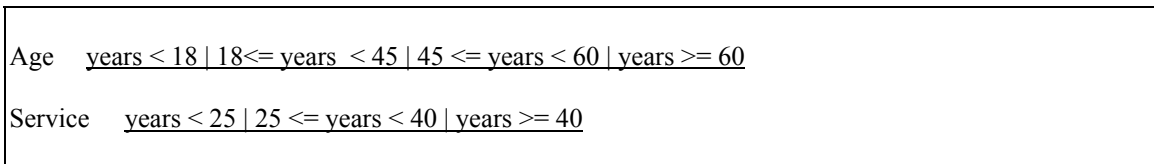


Figure 10. Condition Scales for the Holiday Regulations

Based on these enumerations, we can now fill out the conditions, condition states and actions of our decision table (Figure 11).

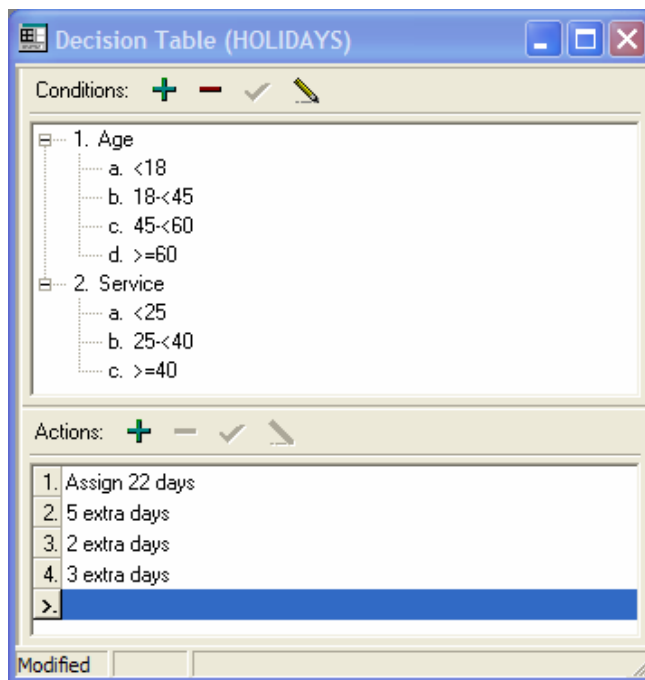


Figure 11. Conditions, states and actions

The product of the number of states for each condition (4 x 3) is the number of combinations that will be present in the (expanded) decision table. When we draw the (empty) table, each combination is included in one (completeness) and only one column (consistency and non-redundancy). This table is shown in Figure 12.

1. Age	<18			18-<45			45-<60			>=60		
2. Service	<25	25-<40	>=40	<25	25-<40	>=40	<25	25-<40	>=40	<25	25-<40	>=40
1. Assign 22 days	.	.	.	.	.	.	.	.	.	.	.	.
2. 5 extra days	.	.	.	.	.	.	.	.	.	.	.	.
3. 2 extra days	.	.	.	.	.	.	.	.	.	.	.	.
4. 3 extra days	.	.	.	.	.	.	.	.	.	.	.	.
	1	2	3	4	5	6	7	8	9	10	11	12

Figure 12. Empty decision table

**4.2. Defining the Decision Rules**

Based on the text of the regulations and on the conditions, the condition states and the actions, we can now proceed by defining the rules. We analyze each line in the regulation and translate it into a rule.

- *Every employee receives at least 22 days*  
Action 1 always has to be performed: there are no conditions.

**Rule 1:** *Assign 22 days* definitely if *always*

Definitely if means that exceptions to this rule will not be allowed.

- *Young employees (less than 18) receive 5 extra days*

**Rule 2:** *5 extra days* if  $Age < 18$

- *If an employee has at least 25 years of service, 2 extra days are given. These 2 days are also provided for employees of age 45 or more, irrespective of their years of service.*

**Rule 3:** *2 extra days* if  $(Service = 25 - < 40 \text{ or } Service \geq 40)$  or  $(Age = 45 - < 60 \text{ or } Age \geq 60)$

- *Employees with at least 40 years of service and also employees of age 60 or more, receive 3 extra days, on top of the additional days already given.*

**Rule 4:** *3 extra days* if  $Service \geq 40$  or  $Age \geq 60$

**4.3. Filling the Decision Table**

After specifying the decision rules, and filling them into the appropriate combinations in the decision table, the table looks as depicted in Figure 13.

1. Age	<18			18-<45			45-<60			>=60		
2. Service	<25	25-<40	>=40	<25	25-<40	>=40	<25	25-<40	>=40	<25	25-<40	>=40
1. Assign 22 days	x	x	x	x	x	x	x	x	x	x	x	x
2. 5 extra days	x	x	x	.	.	.	.	.	.	.	.	.
3. 2 extra days	.	x	x	.	x	x	x	x	x	x	x	x
4. 3 extra days	.	.	x	.	.	x	.	.	x	x	x	x
	1	2	3	4	5	6	7	8	9	10	11	12

Figure 13. A first View of the Decision Table

**4.4. Verifying Completeness and Consistency**

Empty columns, unreferenced actions or unreferenced conditions do not occur in this example. All condition combinations are included in one and only one column.

When, however, we take a good look at the preliminary decision table, we see that it should be impossible for an employee younger than 18 years to have 25 or more years of service. Of course, we would prefer to discard this impossible situation. This can be done by adding a new rule. Since children are not allowed to work, we can also add a rule that employees younger than 45 can not have 40 years of service in the company.

**Rule 5:** *impossible* if  $Age < 18$  and  $(Service = 25 - < 40 \text{ or } Service \geq 40)$

**Rule 6:** *impossible* if  $Age = 18 - < 45$  and  $Service \geq 40$

In fact, we could have defined these semantic impossibilities when stating the concepts. The impossible columns will be deleted from (or marked in) the expanded table.

#### 4.5. Validating Correctness

This last check is a double one. First we verify if the different columns of the decision table correspond with the described specifications. For this example this is the case. If this is not the case, it means that the interpretation of the text is wrong, or in other words one of the rules does not correspond to the text.

Second, one has to check if the decision table corresponds to the desired reality. For instance, it could be possible that management would prefer to give an extra day after 10 years of service, even if this is not mentioned in the text. This is exactly what has to be checked: (1) does the text fully represent the current situation?, and (2) does it represent the desired business situation?

#### 4.6. Simplifying the Decision Table

Once a complete validation of the decision table is finished, the table could be reduced to its minimal format.

##### 4.6.1. Contraction of the decision table

Columns (or groups of columns) leading to identical action configurations can be combined. The contracted decision table is shown in Figure 14.

1. Age	<18		18-<45		45-<60		>=60
2. Service	-	<25	25-<40 or >=40		<25 or 25-<40	>=40	-
1. Assign 22 days	x	x		x	x	x	x
2. 5 extra days	x	.	.	.	.	.	.
3. 2 extra days	.	.		x	x	x	x
4. 3 extra days	.	.	.	.	.	x	x
	1	2	3	4	5	6	

Figure 14. The final Decision Table

##### 4.6.2. Optimizing the condition order

The order of the conditions might influence the number of columns in the contracted table. For this example, the original condition order is already the optimal one.

## 5. Applications Areas

Decision tables were originally used as a technique in computer programming. Due to its representational capabilities, the application area has extended later on to several other domains with logical complexity. In this section, three important current application areas are shortly elucidated:

- *software engineering*: e.g. information systems analysis and description of systems requirements; design and programming; test case generation;
- *complex procedural decision situations in general*: e.g. management procedures; checking and visualizing technical, medical and legal specifications;
- *specification and validation of knowledge based systems*: e.g. legal knowledge based systems; help desk applications; verification and validation.

### 5.1. Decision Tables and Software Engineering

Ever since the birth of the decision table, its use in computer programs has been an important point of attention. In the past, numerous compilers, translators and interpreters were developed, allowing this use of decision tables in programs. However, two important aspects were overlooked:

- Almost all the attention went to an efficient processing of the table and few or none to its modeling and development.
- It was always assumed that the decision table has to be specified as a table, leading to a variety of syntactical problems (aligning columns, continuation indicators, etc.), especially if extended entries are used.

Automating the construction process of the decision table remedies these difficulties, because then a distinction can be made between the *contents* of the table (the decision logic) and the *representation* (which is not relevant to the computer). It then suffices to enclose the decision logic, e.g. expressed in decision rules, in order to use decision tables in programs, because the construction process of the table can be left to an intelligent editor and the translation into an efficient condition oriented selection is the task of the (pre)compiler.

## 5.2. Conditional Logic Representation

The ability of the decision table to represent logically complex decision problems in a readable manner is not limited to the world of programming. Examples of other application areas include: medical diagnosis, laws and regulations, management procedures, etc. Decision tables act as an *intermediate between the specification of a decision process and the real decision making act*, allowing overview and verification throughout the life cycle.

The decision table is **condition oriented** and therefore effective in representing procedural knowledge, with such advantages as: overview, readability, consistency, completeness and correctness.

This structured enumeration of decision columns, however, is not the way in which procedural knowledge is acquired or specified. Most texts, procedures, laws, etc. are described in **action oriented**, partial or modular decision specifications, not suited for fast and correct decision making or easy verification. To this end, the decision table construction process allows to transform the action oriented *specification* into a condition oriented *representation*. The decision table being a representation mechanism, (execution) *optimization* is not the main concern. But it may be, once the table has to be converted into an operational system or a manual decision making procedure, e.g. in order to minimize the total test time or the number of questions. The advantage of the decision table approach, however, is that implementation aspects can be separated from representation aspects through transformation.

The decision table approach, therefore, unifies these three complementary aspects of a decision situation (Figure 15): *specification*, *representation* and *implementation*.

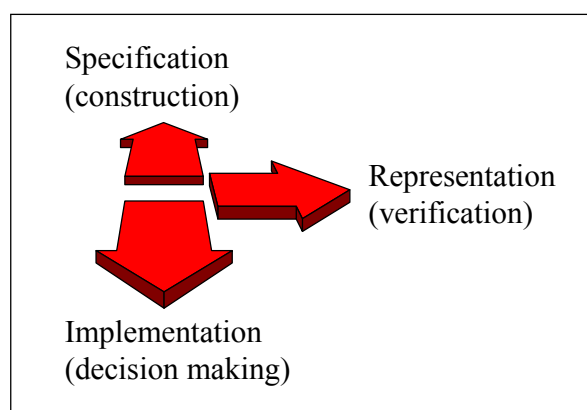


Figure 15. Three aspects of a decision situation

In the past, different experiments have been conducted in order to compare the effectiveness of the decision table as a conditional logic representation tool with other formalisms. Details on these experiments can be found in e.g. Vessey and Weber (1986).

### 5.3. Decision Tables and Knowledge-based Systems

In order to develop high quality knowledge-based systems, methods and techniques are needed to support different stages in the development life cycle. Three important stages can be distinguished: knowledge acquisition, validation & verification and implementation. It has been recognized in the literature that decision tables can play an important role in each of these stages.

Along with the representational properties of the decision table, its ability to be executed very quickly can be applied to increase the performance of knowledge-based systems. In Colomb and Chung (1995) a formal equivalence between propositional expert systems and decision tables is proved, and a procedure is given to perform the transformation to decision tables, thereby substantially increasing execution speed.

Instead of building or generating decision tables only during the validation and verification process or as fast way to execute a knowledge-based system, they can also be used with significant advantage in the knowledge acquisition phase itself when the domain knowledge is being modeled.

The role of the decision table formalism therefore is not limited to occasional and isolated use of tables or similar techniques and can be extended, starting from the early stages of knowledge acquisition and structuring, up to and including the final transformation of decision tables into existing knowledge base tools and products, thus covering the full trajectory of the development life cycle. In Wets (1998) this framework for the development of knowledge-based applications is described, which uses decision tables in several stages of the lifecycle. These decision tables are obtained through the combined use of techniques from knowledge acquisition and discovery. V&V activities are seen as an integral part of this modelling process. At a final stage, the resulting body of decision tables can serve as a basis for implementation.

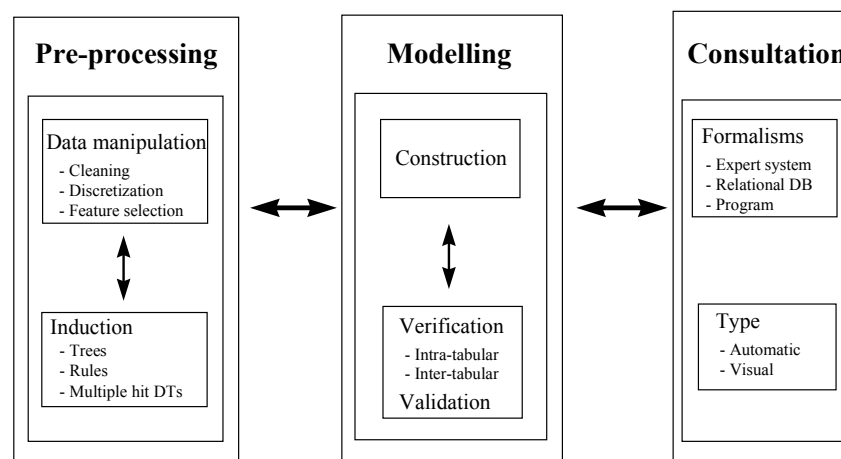


Figure 16. A decision table based development framework

### 5.4. Decision tables and Knowledge Discovery Results

There is an urgent need for techniques which can extract knowledge from data by discovering relations and patterns between the data elements. Due to the strong need for knowledge discovery, many techniques have been proposed to extract knowledge from data. Currently, however, research to verify the extracted knowledge is rather limited.

When machine learning techniques are adopted to mine large data volumes, the extracted information is often presented in the form of rules. Verification & validation (V&V) of the extracted knowledge can then be realized by importing these rules into decision tables. Also the decision table formalism can help the expert to understand the extracted knowledge by visualizing it in a proper way.

In traditional decision table modeling, an expert will provide the knowledge engineer with the necessary rules, where the connection is made between a combination of condition states and some actions that must

be executed. In a data mining application, however, decision rules are extracted from the dataset using some kind of classification techniques. For example, neural networks, machine learning techniques, etc..

The goal of the V&V process is not only to detect anomalies, but also to visualize the anomalies in a proper way. This will allow the expert to correct the anomalies much more easily. Furthermore, visualization of the extracted knowledge using decision tables makes it easier for the expert to consult and interpret the extracted information. The visualization step can be used for two reasons. Firstly, the decision table can be used effectively when the expert wants to inspect the knowledge manually, and secondly it can be used to explain the outcome generated by a consultation application to the expert.

In Vanthienen & Wets (1996) e.g., a data mining experiment at a mail order company was described. It was shown how the decision table formalism can be used effectively to verify the extracted information. Furthermore, it was shown that decision tables prove to be an interesting explanation formalism.

In another paper (see Baesens et al. (2001)), the results are presented from analyzing two real life credit-risk evaluation data sets using neural network rule extraction techniques. Clarifying the neural network decisions by explanatory rules that capture the learned knowledge embedded in the networks can help the human experts in explaining why a particular decision is made. Furthermore, it is also discussed how these rules can be visualized as a decision table in a compact and intuitive graphical format.

### **5.5. Decision Table Automation and Interfacing**

Because manual decision table construction can be a cumbersome process, a design tool for computer-supported construction, manipulation, validation and optimization of decision tables proves valuable. Experiences with such a tool, called PROLOGA (PROcedural LOGic Analyzer) (Vanthienen and Dries (1994) indicate its merits in numerous application areas. One of the cornerstones of the PROLOGA system is that the anomaly detection component is integrated in the modelling phase of the system. Verification takes place as much as possible on the modeling structure.

Also, the decision table formalism is not an isolated technique and shows a lot of *interfaces* to other representation formalisms such as code, trees, rules, etc. Making good use of these connections, however, is only possible through flexible computer support. Therefore a variety of bridges have been built between the decision table workbench and other representations, resulting in a large application domain for decision table modeling.

## **6. Conclusion**

The new realities of business have created new imperatives for business information systems. A major reason for the apparent reuse failure of OO is insufficient attention to the issue of domain understanding and the representation of this understanding in an unambiguous way so it can be validated by business experts. OO notions such as class, association, and message are biased toward the implementation of software systems. Such concepts are not sufficiently expressive of business concepts: rules, constraints, objectives, responsibilities. Therefore they are difficult to validate by business users.

Over the years there has been a major change in research and application areas of decision tables. Current decision table applications mainly benefit from the representational and validation advantages of the technique. The name may not be new, but the key concepts fit very well with recent developments in the areas of business rules. Based on numerous experiences with decision tables, a refined decision table concept and a number of standards have been proposed, which support this vision.

## **7. Acknowledgement**

Ideas and viewpoints expressed in this paper should not be attributed exclusively to the author. They reflect years of working with business knowledge modeling and decision tables at K.U.Leuven and proper credit should therefore be given to current and former research participants. The invaluable contribution

of M. Verhelst is highly appreciated. Any errors or shortcomings, of course, remain the entire responsibility of the author.

## 8. References

1. Baesens, B., Setiono, R., Mues, C., Viaene, S., Vanthienen, J., *Building credit-risk evaluation expert systems using neural network rule extraction and decision tables*, International Conference on Information Systems (ICIS 2001).
2. Business Rules Group, *Defining Business Rules: What are they Really?*, 3<sup>rd</sup> ed., 2002, www.BusinessRulesGroup.org.
3. CODASYL, A Modern Appraisal of Decision Tables, *Report of the Decision Table Task Group*, ACM, New York, 1982.
4. Colomb, R., Chung, C., Strategies for Building Propositional Expert Systems, *International Journal of Intelligent Systems*, 10, 1995, 295-328.
5. Cragun, B., Steudel, H., A Decision-Table Based Processor for Checking Completeness and Consistency in Rule-Based Expert Systems, *International Journal of Man-Machine Studies*, 1987, 633-648.
6. Date, C., WHAT Not HOW, *The Business Rule Approach to Application Development*, Addison-Wesley, 2000, 131 pp.
7. Maes, R., Van Dijk, J. E. M., On the Role of Ambiguity and Incompleteness in the Design of Decision Tables and Rule-Based Systems, *The Computer Journal*, 31(6), 1988, 481-489.
8. Moreno Garcia A., Verhelle M., Vanthienen J., *An Overview of decision table literature 1982-2000*, Research Report 0044, K.U.Leuven, T.E.W., 69 pp., 2000.
9. Morgan, T., *Business Rules and Information Systems – Aligning IT with Business Goals*, Addison-Wesley, 2002.
10. Odell, J., Business Rules, *Object Magazine*, Jan 1995, pp. 53-56.
11. Preece A. & Shinghal R., Foundation and Application of Knowledge Base Verification, *Int. Journal of Intelligent Systems*, 9, 683-701, 1994.
12. Ross, R., *Principles of the Business Rule Approach*, Addison-Wesley, 2002.
13. Vanthienen, J., Dries, E., Illustration of a Decision Table Tool for Specifying and Implementing Knowledge Based Systems, *International Journal on Artificial Intelligence Tools*, 3(2), 1994, 267-288.
14. Vanthienen, J., Mues, C., Wets, G. & Delaere, K., A tool-supported approach to inter-tabular verification, *Expert Systems with Applications*, 15, pp. 277-285, 1998.
15. Vanthienen, J., Wets, G., From Decision Tables to Expert System Shells, *Data & Knowledge Engineering*, 13, 1994, 265-282.
16. Vanthienen J., Wets G., *Decision table based verification of a data mining experiment in a mail-order company*, AAAI 96, workshop on Verification and Validation of Knowledge Based Systems, Portland (Oregon), 4-8 august 1996, 91-95.
17. Verhelst, M., *De Praktijk van Beslissingstabellen*, Kluwer, Deventer/Antwerpen, 1980, 175 pp.
18. Vessey, I., Weber, R., Structured Tools and Conditional Logic: An Empirical Investigation, *Communications of the ACM*, 29(1), 1986, 48-57.
19. Von Halle, B., *Business Rules Applied: Building Better Systems Using the Business Rule Approach*, Wiley Computer Publishing, 2002.
20. Wets, G., *Decision tables in knowledge-based systems: adding knowledge discovery and fuzzy concepts to the decision table formalism*, PhD dissertation, Technische Universiteit Eindhoven, 295 pp., 1998.