A Step out of the Ivory Tower

Experiences with Adapting a Test Case Generation Idea to Business Rules

Rainer Knauf
School of Computer Science & Automation
Ilmenau Technical University
PO Box 100565, 98684 Ilmenau
rainer.knauf@tu-ilmenau.de
Germany

Silvie Spreeuwenberg
Rik Gerrits
LibRT B.V.
Silodam 264, 1013 AW Amsterdam
The Netherlands

Martin Jendreck
Bahnhofstraße 62 A
96515 Sonneberg
Germany

Abstract

One author developed a validation technology for rule bases that aims at several validity statements and a refined rule base. Two other authors are experienced developers of rule bases for commercial and administrative applications and engaged in the Business Rule (BR) research community. To ensure the requested performance of BR, they developed a verification tool for BR. To reach this objective completely, one gap needs to be bridged: The application of validation technologies to BR.

The application of the validation technology's first step, the test case generation, revealed basic insights about the different viewpoints and terminologies of foundation– and logic oriented AI research and application oriented knowledge– and software engineering.

The experiences gained during the realization of the test case generation for a BR language are reported. In particular, (1) the trade–offs between logic approaches, commercial needs, and the desired involvement of other (non–AI) software technologies as well as (2) the derived refinements of the theoretical approach are one subject of the present paper.

Introduction

Business Rules (BR) are a common knowledge representation in commercial environments. They describe and automate the business function in a declarative manner (Vanthienen 2001). Their purpose is the support of decision making in commercial and administrative environments (Rosca et al. 1997). Regarding their semantic, two basic classes of BR are distinguished (Vanthienen 2001):

- Rules that describe *constraints*, i.e. (a) stimulus–response causality, (b) operation constraints, which state pre– and post–conditions on object operations, and (c) structure constraints, which state class invariants, and
- rules that describe *derivations*, i.e. (a) computation rules, which state how something is calculated and inference rules, which fit the association to rules in Knowledge Based Systems.

Formal BR are *if–then* – statements with a conjunctive and/or disjunctive statements about an attribute's value in their condition part and an assignment of such a value to an attribute in their conclusion part.

On the one hand, they provide a "sub–language" that is simply enough to handle for domain (but not Computer Science) experts; on the other hand, they provide additionally a great variety of opportunities to express even complicated matters by including modern software engineering methods additionally. Their main purpose is to control courses of business.

The range of current application domains shows their usefulness in commercial and administrative environments: administration, legislation, regulation, advertising, banking (credit decisions), product sales, budgeting and others. In these application fields (as in the most interesting ones), gathering correct and complete knowledge is one of the greatest difficulty. Usually, the acquired rules are contradictory and/or insufficient. Their maintenance is a non–trivial challenge, because it often introduces unnoticed inconsistencies, contradictions and other anomalies (Vanthienen 2001). Therefore, BR need to be verified and validated after their specification for a certain application environment.

The verification issue is well developed. Based on long term experience of developing verification techniques and tools, the authors of (Spreeuwenberg and Gerrits 2002) list up requirements derived from practice. Interestingly, this issue causes a change in the interaction between developers and analysts or experts a soon as a verification tool is involved in the development process (Gerrits 2003). In fact, the verification contributes to the support of quality management for BR(Spreeuwenberg 2003).

For the validation of rules in Knowledge Based Systems, promising methodologies and techniques are developed (Knauf 2000; Knauf, Gonzalez, and Abel 2002) and currently refined (Kurbad 2003), but unfortunately not applied to BR so far. To gap this bridge, the authors started the adaption of the technology introduced in (Knauf 2000) to BR environments by implementing its first step, the *test case generation*. Interestingly, this attempt revealed that approaches developed in an "Ivory Tower" are often based on assumptions that do not occur in the commercial practice. Experiences and insights gained in this process are the subject of the present paper.

A Closer Look at Business Rules

The original objective of the BR concept is the description and control of business courses. According to this claim, they occur in different forms, which do not all meet what AI researchers would call a "rule" and even not meet what they would associate with the respective term:

- *Definition of Business Terminologies* It is usually described with glossaries or entities of an Entity Relationship Model (ERM). A business terminology can be more formally described with description logics which are commonly used in the Ontology community.
- *Facts* Here, they express relationships between terminologies. They are represented either as natural language statements or by some graphical model.
- *instructions and restrictions* They control the access to business data. In fact, they meet our imagination of a "rule", since they are represented by IF–THEN – constructions.
- *derivations* They define ways to conclude knowledge (a fact, e.g.) from the given knowledge base (rules and facts). From an AI viewpoint, they perform the inference calculus.

Obviously, BR provides a large variety of formal, semi–formal and informal knowledge representations and do not provide an inference technology. Consequently, the BR community developed a commonly accepted model BR (represented as an ERM in UML notation) along with a definition of the involved terms (Business Rules Group 2000).

To apply AI technologies to BR, Production Rules are derived and languages for their processing have been developed. One exemplary language that is the subject of further considerations, is *CleverPath Aion Business Rules Expert* (AIONBRE) (Computer Associates 2003).

The Representation Aspect

Rules in AIONBRE follow the association from the point of AI. Formally, they fall into the class of HORN clauses of the propositional calculus, but their expressivity is much more powerful than this, because their premises and conclusions are not limited to logical statements. Three kinds of rules are distinguished:

- *Production Rules* They infer about object within a class hierarchy. From the viewpoint of frame technology, the premises check slot contents (attribute values) for deciding, whether or not a rule fires. The conclusion consists in setting attribute values. From the AI perspective, the single expressions are first order statements about a hierarchically structured world of objects.
- *Pattern Matching Rules* These are second order statements which serve the manipulation of the class hierarchy. Before their use, so called variables needs to be bound to classes. They are excluded from processing in a Backward–Chaining manner. Their premise is checking attribute values of a class indicated by a given variable. Since this check includes the consideration of all instances, these rules are named pattern matching rules. Their conclusion part is an instruction to bound a class to one or more classes.
- *Event Rules* Fortunately, they are not relevant in practice. In fact, they are out of the inference engine's control. Their premises are checked permanently and they fire with a higher priority then any rule of the first two kinds. From an AI perspective, their effects can't be described by any (non–temporal) formal logic.

The Processing Aspect

The processing of these rules is very flexible. AIONBRE provides a Forward– and backward–Chaining rule processing with or without the control of rules' priorities. Furthermore, an implicit inference is performed by the inheritance within the class hierarchy. Finally, a kind of "experimental" preliminary and retractable inference is provided by Truth Maintenance Assignments.

The Scenario to Perform Validation

Since verification is a pre–condition for the usefulness to apply validation technologies, the implemented first step of the validation technology is embedded in an existing and successfully applied verification tool called VALENS, a commercial product of the Dutch company LIBRT (see www.librt.com and click on VALENS). It is a verification engine than can be used (1) as an integrated tool within AIONBRE, (2) as a stand alon version to verify AIONBRE rules, and (3) a component version that can be adapted to other rule base development environments that are based on the LIBRT's rule base XML schema.

VALENS provides a large variety of consistency –, redundancy –, and completeness checks as well as some elementary validity checks regarding "dead ends" of inference, i.e. unused conclusions and unreachable goals. The functionality of VALENS is highly motivated and influenced by long term practical experiences with the application of BR.

Since VALENS' analysis is based on the logical structure, it can not decide, up to which grade a given rule base really models reality. This motivated the supplementation of VALENS by a validation technology.

The first step of the validation technology to be integrated is the *test case generation* (Knauf, Gonzalez, and Abel 2002). In a very first sub–step the rules are analyzed with the objective to compute a so called Quasi Exhaustive Set of Test cases ($QuEST$). Consequently, the tool to perform the computation of $QuEST$ has been named QUESTER.

QUESTER generates a test case set that meets the following requirements:

- For each of the system's possible final output there is at least one test data $t_j \in QuEST$.
- The test data are able to reflect the boundary conditions between different system outputs, i.e. they "frame" each boundary with so–called "scanning distance" to it.
- The cardinality of $QuEST$ is as small as possible.

The process is based upon the following ideas:

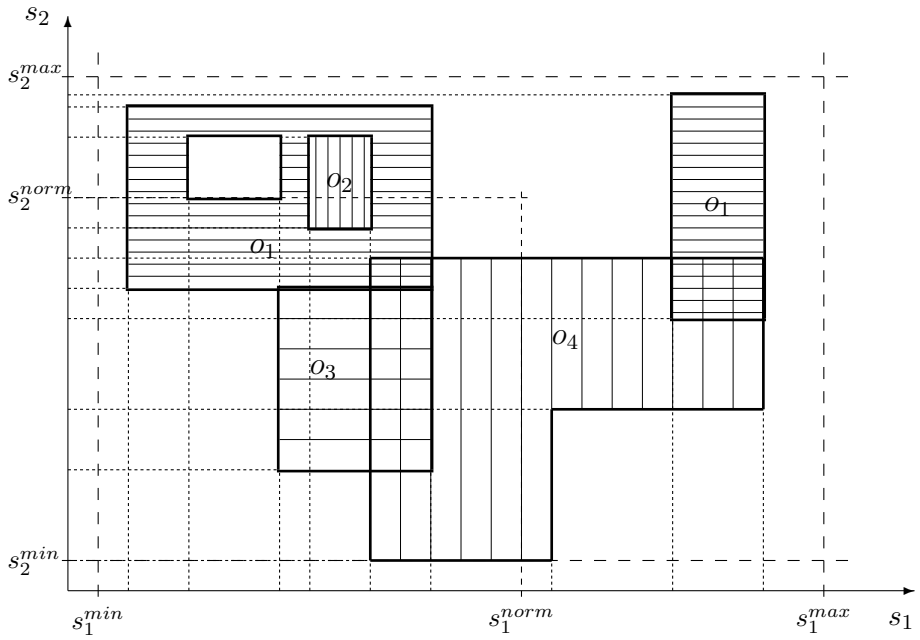

Figure 1: Regions of Influence for a 2 Input Problem

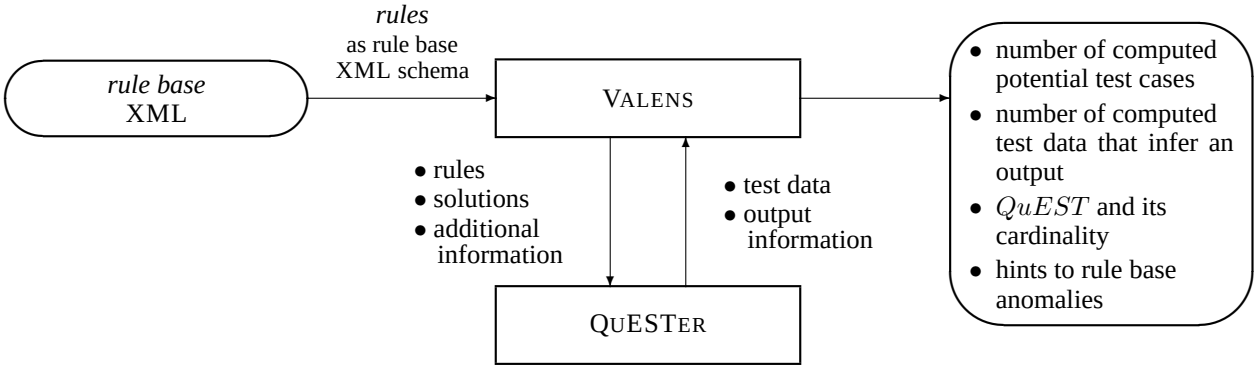

Figure 2: The Architecture of incorporating QUESTER into VALENS

1. Break down the range of an input into subranges where its values are considered to be equivalent in terms of its effects on the outputs.

   All values within such a range are within the same "region of influence" formed by the intersection of the projection of the values of the inputs that have a direct effect on a particular output. Thus, all cases within such a region are mapped to the same output by the rule base. See figure 1 for illustration.

2. Compute an initial set of potential test data P based upon combinations of values within these subranges.

3. Sort these data t_j into several sets P_i of data according to the output o_i that the system infers for the input T_j.

4. Filter all P_i by eliminating those that are subsumed by others.

   Via a well–defined interface QUESTER accesses the rule base (represented as an XML schema) and delivers the computed test data along with their expected test case solutions as illustrated in figure 2 by following the approach in (Knauf 2000) as sketched in figure 3.

Ivory Tower Ideas vs. Practice

Dependencies of Outputs

By analyzing the rule base $R = \{r_1, \ldots, r_n\}$ the dependency of each particular system output $o_i \in O$ from the input variables in $S = \{s_1, \ldots, s_m\}$ and the rules R is determined.

During the very first step, the computation of the outputs' dependencies on inputs and applicable rules revealed a first general misunderstanding about the conclusion's inherent logic: The developer of the validation approach assumed that an output is just a propositional statement that can be $true$ or $false$.

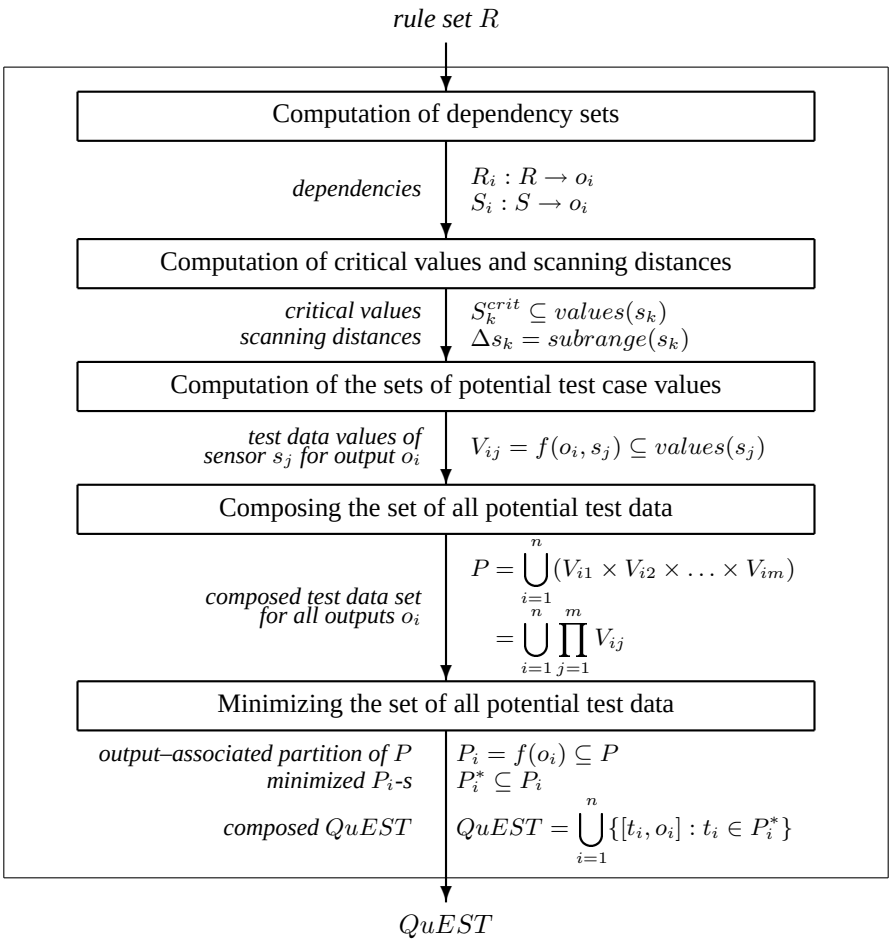

Figure 3: Computation of $QuEST$

The practice of AIONBRE and VALENS taught, that a *written attribute* is considered an output. By associating a particular value, it becomes a logic statement and thus, an output in the eye of the validation approach developer.

This didn't rise a difficulty in the computation of the above mentioned dependencies. Therefore, this issue was not solved at this point, but kept in mind.

Later we came to the insight, that this issue dramatically influences the number of computed test data and thus, the costs of executing the entire validation approach. Since a written attribute can have many values, an output in the sense of VALENS conforms many outputs in the sense of the validation technology. Since the number of values occurring at rules' conclusion parts is usually a very tiny subset of all possible attribute values, the number of test data is very much limited by underlying the validation approach assumption of an output's nature.

Another big issue is the assumption that a rule is a completely logic item. This sounds quite naturally, but is far away from practice. Difficulties occurred with elements like function– or method calls in the rules. This could be solved by "normalizing" the rule base to a one that is free of such calls.

A similar solution could be found for rules which don't follow the pure HORN–Logic because of having multiple (conjunctive) conclusions, i.e. rules like

$$\bigwedge_{i=1}^{n} B_i \leftarrow \bigwedge_{i=1}^{m} A_i$$

These rules where simple decomposed into several rules

$$B_1 \leftarrow A \quad \ldots \quad B_n \leftarrow A$$

Since inferring a hypothesis H from a set of such (facts and) rules means considering the conjunction of the elements of the rule base this decomposition leads to a rule base that is logically equivalent:

$$(\bigwedge_{i=1}^{n} B_i) \leftarrow A \equiv \bigwedge_{i=1}^{n} (B_i \leftarrow A)$$

Critical Values and their Scanning Distance

In this step,

- the values, which mark a difference in the effect of an input's value on one of its dependent final outputs (critical values) and

- the scanning distances, at which test data should be located from these critical values

are determined.

Particular (rule) languages support particular data types for the inputs and outputs of the rule base. The adaption and extension of the data structure that has been considered in the validation approach to the data structure supported in AIONBRE could be performed quite easily.

One issue that needed to be solved is the handling of type conflicts when data is relayed through the inference chain. In case the types are not compatible (an integer and a string that doesn't represent a number, e.g.), this indicates an error in the rules' semantic. In case it is compatible, a policy to translate the data has been developed. This policy consists of rules like this:

> *Upper borders of a real number's rage have to be rounded down when translated to an integer value*

If two input values (of different, but compatible data types) are compared, it happened, that the computed *critical values* (see figure 3 and (Knauf 2000)) are not allowed for the associated data type, for example that they are out of its defined value range. To handle these cases, we developed a policy of replacing such values by some "nearest valid values".

The approach to compute the scanning distances needed also to be adapted to the circumstances of practice. Originally, it did not consider the case that an input has several valid ranges with no intersection with each other. In this case, the smallest of the computed scanning distances for each range is adopted to the entire input attribute.

Furthermore, a "smallest possible step" for integer and real values needed to be introduced. For integers, it is quite clear (=1), but for real values, we had to made a very arbitrary decision with respect to AIONBRE's handling of fractional numbers and to the further use of this step width.

Some additional difficulties revealed with the handling of function calls and arithmetic expressions. In case they use attributes and their values need an analysis of intermediate conclusions, additional rules (besides the ones in the o_i–associated rule dependency set R_i) need to be included in the rule analysis. For such cases, the rule base needs to be normalized before the execution of QUESTER.

Finally, we had to state another limitation of applicability: It is limited to rules that infer a static output, i.e. an output that doesn't refer to input values. The conversion of AIONBRE – output attributes to the output definition underlaying the validation approach before the execution of QUESTER – which is desirable anyway: see above – will solve this problem, but we didn't do this yet.[1]

Computing Potential Test Case Values

The approach to compute test case values is based on an analysis of expressions about inputs in the rules' condition parts. Here the outputs are considered separately and a set of potential test case values V_ij for a considered input s_j in the test case (sub–) set for an output o_i are computed.

Originally, three cases have been distinguished: (1) the input does't influence the output o_i, (2) the input is compared with a fixed value in a rule's condition part, and (3) the input is compared with another input in a rule's condition part. Since every input attribute has to have any value (even if this value doesn't matter), the approach simply stated a "normal value".

While practicing this idea, we came to the insight, that such an arbitrary value should be chosen in a manner that supports minimality. This could be performed by not using a "normal value", but any value of this input that has been computed as a potential test case value for another output. This way we could arrange that some test data could serve for several inputs. In fact, the cardinality of $QuEST$ (i.e. the number of computed test data) could be decreased significantly by this measure.

Additionally, we had to perform some other slight adaption of the approach with respect to the introduce data types of AIONBRE that are of a larger number than the ones assumed in the approach.

Composing Complete Test Data

Since this step is just the computation of set products and their union, it did not cause any problem.

Minimizing the Set of Test Data

This was the point to solve the problem, which was put off in the very first step: The definition of the term "output" as either an output attribute or a statement about its value, because the first step of the minimizing procedure associates the test data with their outputs and forms a respective subset for each output. The minimization procedure is applied to every subset of $QuEST$ individually.

For complexity reasons (cf. first subsection) we preferred the definition in the approach. The minimization idea is based on revealing test data that are subsumed by others because of being in the same "region of influence" (Knauf, Gonzalez, and Abel 2002). In fact, such a chance occurs only by considering non–discrete input attributes.

Applying the minimization procedure to test cases of practical relevance was not as complex as we where afraid of. The iterative technology as introduced in (Knauf 2000; Knauf, Gonzalez, and Abel 2002) could be replaced by a "quasi concurrent" technology for each possible partitioning with respect to an input that is non discrete. We could show, that this way no chance of minimization is missed.

At this point, the developer of the minimization procedure came to a new theoretical insight while explaining his technology to developers of the tool which performs it in practice.

Providing other Useful Information

The procedure to compute $QuEST$ reveals also information that is not its objective, but useful for (1) practicing upcoming steps to implement the validation technology, (2) improving the verification procedure of VALENS, and (3) the rule base development in general.

[1] The procedure to do so is quite similar to the procedure which converts a MEALY automaton into a MOORE automaton.

Refining the Minimization Technology

Based on the experiments with QUESTER another insight lead to a refinement of the minimization technology of (Knauf 2000). The refinement idea of (Jendreck 2003) is based on the strategy to prefer test data that can be mapped to as many as possible outputs to those which are mapped to less outputs.

Summary and Outlook

The paper focuses insights while performing a first step to bridge an important gap in the development of BR bases: the involvement of validation approaches.

The implementation of a test case generation method revealed basic insights about the different viewpoints and terminologies of foundation– and logic oriented AI research and application oriented knowledge– and software engineering.

The experiences gained during the realization of the test case generation for a BR language have been illustrated in the paper. In particular, the trade–offs between logic approaches, commercial needs, and the desired involvement of other (non–AI) software technologies are one subject of the present paper.

Furthermore, the test case generation technology itself has been refined based on the experience with real life application environments. In particular, the complexity of a minimization methods could be dramatically decreased by using a more efficient algorithm. Additionally, the outcome of this step, the test case set could be further decreased by refining the minimization approach.

Since there is still a long way until BR applications can enjoy the benefits of validation approaches, the research in both communities BR knowledge engineering and $V\&V$ of intelligent systems need to adapt their terminologies and approaches to each other's requirements. In fact, the authors continue following this this way towards high performance BR bases.

References

Hay, D.; Healy, K.A. 2000. Defining Business Rules – What Are They Really? Final Report od the Business Rules Group (formerly: GUIDE Business Rules project), revision 1.3, July 2000.

Computer Associates, 2003. see dowload page for system documentation:
http://support.ca.com/aionmanuals/aion_manuals.html

Gerrits, R. 2003. Benefits and Requirements of Rule Verification. J. Hall& M. Schacher (eds.): Proceedings of the Second European Business Rules Conference, Zürich, Switzerland
www.eurobizrules.org/speakers/gerrits.htm

Jendreck, M. 2003. *Anpassung eines testfallbesierten Validationsverfahrens an eine Entwicklungsumgebung für Business Rules.* Diploma Thesis, Ilmenau Technical University, School of Business Economics, Ilmenau, Germany.

Knauf, R. 2000. *Validating Rule–Based Systems – A Complete Methodology.* Habilitation Thesis, Ilmenau Technical University, Faculty of Computer Science and Automation, ISBN 3-8265-8293-4, Aachen: Shaker.

Knauf, R.; Gonzalez, A.J.; Abel, T. 2002. A Framework for Validation of Rule–Based Systems. *IEEE Transactions on Systems, Man and Cybernetics, Part B: Cybernetics*, vol. 32, # 3, pp. 281–295.

Kurbad, T. 2003. A Concept to Apply a Turing Test Technology for System Validation that Utilizes External Validation Knowledge. Diploma Thesis, Ilmenau Technical University, Faculty of Computer Science and Automation, Inv.-Nr. 2003-09-03/053/IN95/2238

Rosca, D.; Feblowitz, M.; Wild, C. 1997. Decision making Methodology in Support of the Business Rules Lifecycle. Proc. of the 3^{rd} IEEE Internat. Symposium on Requirements Engineering (RE'97), Annapolis, MD, USA, pp. 236–246.

Spreeuwenberg, S. 2003. Quality Management of Business Rules. J. Hall& M. Schacher (eds.): Proceedings of the Second European Business Rules Conference, Zürich, Switzerland
www.eurobizrules.org/speakers/Spreeuvenberg.htm

Spreeuwenberg, S.; Gerrits, R. 2002. Requirements of Successful Verification in Practice. Kohlen (ed.): *Proc. of the 15^{th} Internat. Florida Artificial Intelligence Research Society Conference 2002 (FLAIRS-02)*, Pensacola Beach, FL, USA, pp. 221–225, Menlo Park, CA:AAAI Press.

Vanthienen, J. 2001. Ruling the Business: About Business Rules and Decision tables. Vandenbulcke, J. & Snoeck, M. (eds.): *New Directions in Software Engineering*, Leuven University Press, Leuven, pp. 103–120.