
PERGAMON

Expert Systems with Applications 23 (2002) 385–394

Expert Systems with Applications

www.elsevier.com/locate/eswa

Short Survey

Towards a flexible deployment of business rules

Daniela Rosca[a,*], Chris Wild[b]

[a]*Department of Software Engineering, Monmouth University, West Long Branch, NJ 07764, USA*
[b]*Department of Computer Science, Old Dominion University, Norfolk, VA 23529-0162, USA*

Abstract

Business rules give rise to an important set of requirements on any system being developed or procured for an enterprise. While most of the work done in this area focuses on identifying and documenting business rules, we have proposed a methodology that addresses several aspects of the business rules lifecycle: acquisition, deployment and evolution. The methodology assumes that business rules are expressed in terms of business concepts and corporate knowledge that are captured in a high level architecture. The architecture proposed consists of three interconnected components: the enterprise model, the business rules model and the decision support model. This approach permits a greater variety of rules to be specified while providing an opportunity to automate the production of deployable business rules. The ability to deal with the inconsistent and ambiguous rules is crucial in capturing the conflicting requirements placed on the operation of any large scale enterprise. This paper presents a flexible deployment of business rules, which not only supports decision making in the face of conflicting requirements, but also the evolution of those requirements in the face of changing regulatory environments, competitive markets and corporate goals.

Keywords: Business rules; Decision support systems; Enterprise modeling; Requirements engineering; Inconsistency; Ambiguity; Decision tree learning

1. Introduction

Business rules give rise to an important set of requirements on any system being developed or procured for an enterprise. Others have recognized the need to identify and model the business rules (GUIDE International Co, 1997; Kilov & Simmonds, 1997), or the importance of having traceability links between the business rules and the system components (Kilov & Simmonds, 1997). Also, it has been shown the necessity to storage business rules in a central repository, and strive for their clarity, consistency, and completeness (Feuerlicht & Blair, 1990). Recently, their role in e-commerce applications has been demonstrated by Grosof and Labrou (1999).

The strategic importance of the business rules require that they be expressed in terms that can be understood by the business organizations, and that they can be easily changed as the need to improve the performance of the business arises due to changes in laws, market forces or enterprise goals.

The types of business rules that can be expressed and reasoned about depend on the richness of the enterprise model in terms of which the rules are stated. In much of the current literature (Chen, Segarra, & Chong, 1992; von Halle, 1993), the expressiveness of business rules is limited to whatever can be expressed in Entity–Relationship style (E–R) models. Others, including Bubenko and Wangler (1993), Champion and Moores (1996), Herbst (1995) and Loucopoulos and Katsouli (1992), express business rules in terms of business concepts and corporate knowledge that are captured in a more general conceptual modeling architecture. This approach is adopted in our BRADES (Business Rules Acquisition Deployment and Evolution System) methodology (Rosca, 1997; Rosca, Greenspan, Feblowitz, & Wild, 1997). The architecture proposed consists of three components: the enterprise model, the business model, and the decision support model. An overview of this architecture is presented in Section 2. More information about it can be found in Rosca (1997) and Rosca et al. (1997). The relationships between these models support a wide range of requirements analyses.

While most approaches have covered only the identification and documentation of business rules, BRADES looks at aspects from their entire lifecycle. The motivation for this topic stems from the natural evolution of understanding which starts when defining the business rules and continues throughout their lifetime. The methodology covers aspects of the acquisition, deployment and evolution of business rules. An overview of the methodology is presented in

* Corresponding author.
E-mail addresses: drosca@monmouth.edu (D. Rosca), wild@cs.odu.edu (C. Wild).

0957-4174/02/$ - see front matter
PII: S0957-4174(02)00074-X

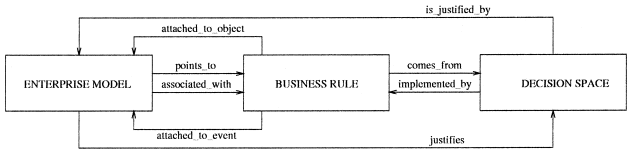

Fig. 1. The business rules environment.

Section 3. The acquisition phase has been broadly covered in Rosca et al. (1997), the deployment phase will be discussed in more detail in Section 5 of this paper, while the evolution phase is the subject of a paper in preparation.

The deployment of business rules is an important part of the methodology, which includes the resolution of issues such as non-determinism, inconsistency and ambiguity which cannot be solved until the business rule is instantiated in a particular situation. In addition, the methodology covers the evaluation of how well the operational system satisfies the requirements represented in the business rules. All these situations involve tradeoffs and exceptions, characteristics of a much needed flexible deployment of business rules in this world of increasingly competitive markets.

There is a significant body of literature that deals with inconsistency and incompleteness, two major issues in software requirements (Balzer, 1991; Boehm, Bose, Horowitz, & Lee, 1995; Easterbrook & Nuseibeh, 1995; Ghezzi & Nuseibeh, 1998; Grosof & Labrou, 1999). Dealing with inconsistency as it arises, not necessarily at analysis time, has been advocated also in Pohl et al. (1997), where 'choice contexts' are used for flexibility at run time. In Fickas and Feather (1995), there is a discussion of requirements monitoring to instrument the running system to determine whether, and to what degree, requirements are being met by the system.

Most of the above related work represents conflicting requirements in various types of logic, for an easier detection and resolution of conflicts. For example, in the viewpoints framework, the views have specific consistency rules associated with them, which allow consistency checking by evaluating the corresponding rules on pairs of viewpoints. We propose an approach based on a decision support system (DSS) and an algorithm for the rapid generation of business rules. Instead of automatically resolving conflicts, which might not be always possible, BRADES guides the users in solving a conflict, or an ambiguous situation. The algorithm described in Section 4 is based on the induction of rules from decision structures data and domain assumptions. Our approach allows the deployment of business rules to be customized, based on the data available at operation time.

2. A high level architecture for the business rules environment

The high level architecture consists of three models: the *Enterprise Model*, the *Business Rules Model*, and the *Decision Space Model*. The Enterprise Model represents the world to which the business rules apply. It defines the domain concepts about which the rules are expressed. The Business Rules Model represents the business rules themselves. The Decision Space Model offers information about the enterprise objectives that comprise the origin of business rules and captures the reasoning leading to the selection and ultimate generation of the business rules (see Fig. 1).

2.1. Enterprise model

For the representation of the Enterprise Model component the paradigm of the LiveModel modeling environment (Intellicorp, 1995) was chosen. In LiveModel an enterprise is represented in terms of 'objects' and 'processes' (see Martin and Odell (1995)). Objects are represented by Object Diagrams that are essentially Entity–Relationship diagrams.

We represent business processes by a set of LiveModel Event Diagrams, which define the sequence of operations and events for process execution. For example, Fig. 2 represents a process in a corporate travel office. When an employee presents documents for travel reimbursement, there might be missing documents, such as a missing meal receipt, or company rules violations, such as not staying over the weekend to lower the airfare. The clerk at the travel office needs to solve the discrepancies found in the documentation and decide what action to take towards the employee. The Event Diagrams are executable specifications of a process as soon as: (1) input and output variables to operations are specified; (2) trigger rules are created to define branching and control conditions; (3) procedures to define operations are written. LiveModel allows the attachment of rules to Event Diagrams, with a particular operational semantics based on those of the Object and Event Diagrams.

2.2. Business rules model

Most of the literature assumes business rules to be stated in natural language. This representation is less amenable for the application of formal methods or tools that would permit some form of automated reasoning. Nevertheless, this might be the most appropriate representation for the business rules that represent policies of an enterprise, because they need to be formulated and analyzed by business-oriented people. For example, Leite and Leonardi (1998) have included business rules expressed in natural language in the client oriented requirements baseline, for helping in the identification of software requirements. They propose scenarios as possible ways for enacting business rules viewed as policies.

Others (Bubenko & Wangler, 1993; Champion & Moores, 1996; Herbst, 1995), propose the form event–condition–action (ECA) for representing business rules. This form is more appropriate for representing and

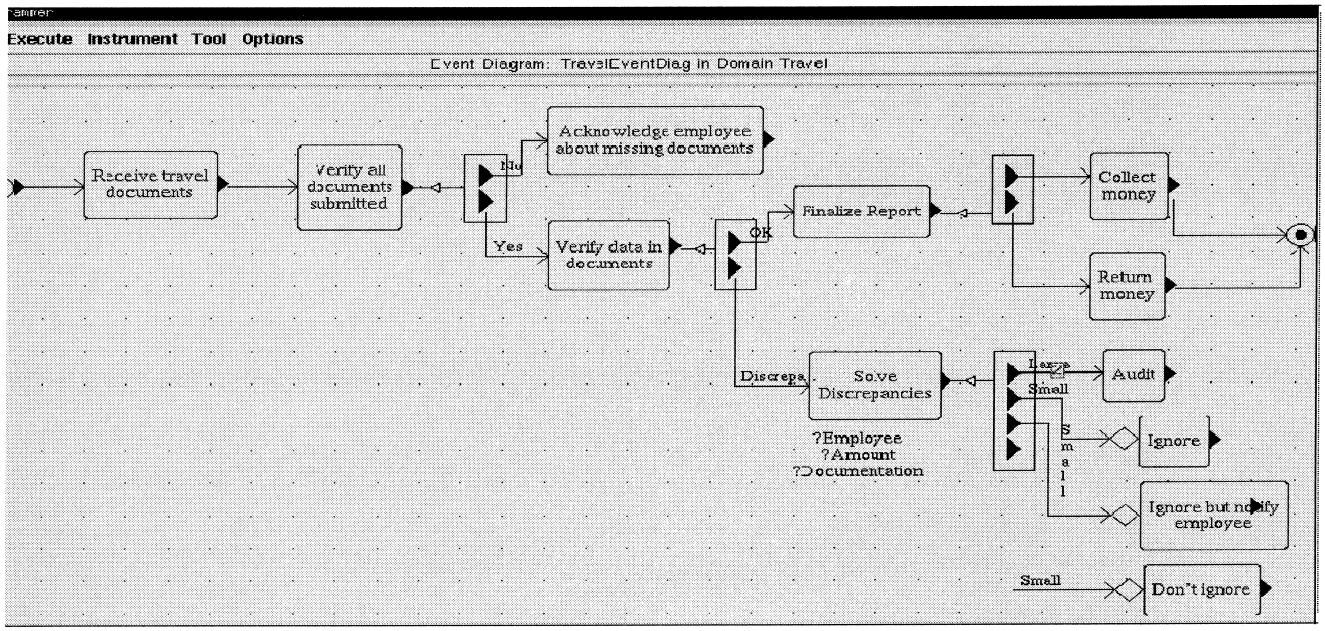

Fig. 2. An Event diagram example.

reasoning about operational business rules (rules that support the daily operation of the business). It provides a convenient way of representing most types of business rules, as long as events, conditions and actions can be identified, as shown in the BRADES process/object enterprise model, or other enterprise models such as Greenspan and Feblowitz (1993) or Yu (1994).

For the simplest form of a BRADES ECA rule

WHEN *event* IF *condition* DO *action*

when the event occurs, if at that time the condition is found to hold, then the action is initiated.

The events, conditions, and actions are formulated as expressions on the entities in the Enterprise Model.

2.3. Decision space model

The Decision Space (DS) captures pre-traceability information which consists of the business rationale that supports the elicitation of business rules. It offers information about the origin of business rules, tracing back to the enterprise objectives the reasoning leading to the selection and generation of the business rules. DS is the repository of information about *issues*, *alternatives*, *criteria*, *arguments*, *assumptions* that provides a domain specific knowledge source for assisting not only in the elicitation of business rules, but also in their deployment and evolution. The interpretation of these primitives is as follows. The refinement of high-level business rules generates *issues* that need to be solved. In the context of an information system that is introduced to support some activities of an enterprise, requirements elicited from stakeholders can also raise new issues that need to be solved, which would ultimately lead to the creation of new rules or the modification of existing ones. These issues are refined during the deliberation process. In order to solve an issue, different alternative solutions are considered for evaluation. The alternatives are evaluated against a set of *criteria* in order to decide which gives the best solution. A decision involves assessing the degree to which each alternative meets the entire set of criteria and choosing that alternate which best satisfies this set. *Arguments* and counterarguments based on various *assumptions* are recorded to document the evaluation of the alternatives, or the creation of new issues that may follow after making a decision. The best alternative solution is reflected in the resulting artifact, which in our case is represented by DSS *Business Rules*, a set of business rules in DSS format.

The decision process produces the business rule and its related *decision matrix* and *grounding guide*. The decision matrix synthesizes the criteria for making the decision regarding a business rule and the alternatives considered together with their underlying arguments and assumptions (see Fig. 3). This information is displayed in a staged process, upon user request.

The grounding guide is a structure created at operation time for the business rules that contain ungrounded terms. It describes information related to ungrounded terms and applicable definitional business rules extracted from the links between the models of the high-level architecture. This information can be used for choosing one of the existing values of an ungrounded term based on the shown definitional business rules, or for introducing new values for grounding a term. An example of a grounding guide is shown in Fig. 5.

More explanations about how it is used are given in Section 5.

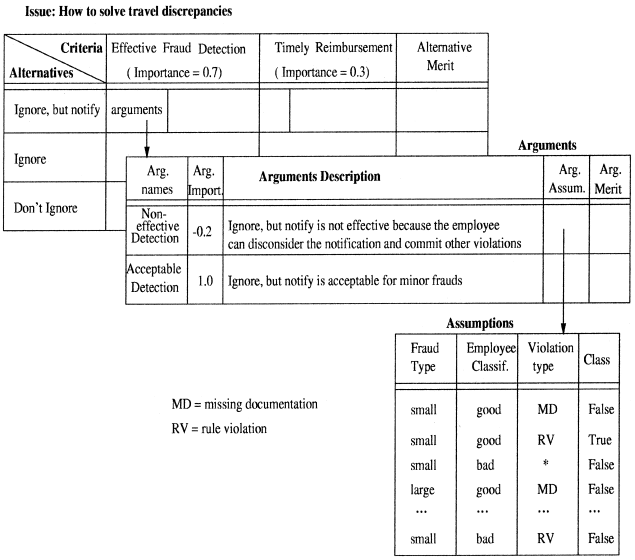

Fig. 3. Decision matrix associated with the issue *How to solve discrepancies* related to the operation *Solving Discrepancies* in Fig. 2.

Among the three components of the architecture that supports the business rules lifecycle we can identify traceability links (Rosca et al., 1997) that allow a higher degree of flexibility for enterprises that need to quickly react to various market or internal changes. The post-traceability information derived from the described links can be a very valuable asset for an impact/sensitivity analysis when one component of the architecture is changing. More details about this model and related work on decision support structures is given in Rosca et al. (1995).

3. Methodology for the business rules lifecycle

Although there has been a growing interest in capturing business rules in the last decade, there has been no attempt at addressing their lifecycle. The BRADES methodology (Rosca et al., 1995, 1997) spans all phases of the business rules lifecycle: acquisition, deployment and evolution. Next, an overview of the methodology is presented.

During the acquisition phase, enterprise objectives, goal-oriented rules and constraints are discussed, raising issues that need to be solved in order to generate operational business rules. This phase includes not only the acquisition of business rules and the enterprise model in terms of which they are stated, but also captures the deliberation process that lead to those rules, starting from high level enterprise goals. Based on the decision structures populated during the deliberation process, we developed an algorithm for the automatic acquisition of certain types of business rules that will be presented in Section 4.

During the deployment phase the methodology makes a distinction between deterministic and non-deterministic rules. The deterministic rules uniquely characterize the situations where they can be applied without further decision making. Therefore, when encoded in the enterprise's information system, they are automatically applied by the system, or by humans.

In the case of non-deterministic rules, human interaction is required due to either conflicting or ambiguous business rules. We call these interactions *operational decisions* because they are taken at operation time instead of analysis time. BRADES assists the decision maker by providing the deliberation information associated with the conflicting rules, and in the case of ambiguous rules, by providing the definitional rules that would help in grounding the terms with values available at operation time.

Conditional decisions are decisions which have been chosen by the developers to be monitored during the operational usage. The reason for monitoring a decision vary depending on the confidence the original decision makers had in the assumptions underlying a decision, and/or the importance of the deployed business rule to the success of the organization. In the case of conditional decisions, where there is a question about the adequacy of the decision assumptions, we propose to monitor the consequences of the decision by identifying observable information which can be captured during operation. This information will measure the effects of applying a business rule and will validate or deny the decision of choosing that particular rule, after reflecting on the monitoring data.

In the evolutionary phase, based on the information provided by the monitored data or by the operational decisions, one could determine that some business rules are wrong or incomplete. In correcting and improving the set of business rules, the decision model can be used to assess what is missing or whether any changes have occurred in the deliberation information. For example, one could determine that there are some missing criteria, arguments, alternatives or assumptions underlying a business rule. Also, changes in internal or external sources of an enterprise might induce the necessity of changing one or more business rules. In all these cases BRADES assists the process of evolution of business rules by providing the information existing in the decision space associated with the rules that need to be changed, and by providing an algorithm for the rapid modification of the rules, based on the new information.

The methodology described here supports CMM (Paulk, Curtis, & Chrisis, 1993) level 4 practices by prescribing a systematic collection of monitoring data and their analysis for a better understanding of the business processes. Also, by allowing the issues that appear during the operational decisions to be analyzed and contribute to the evolution of business rules, the methodology also incorporates CMM level 5 practices, which fuel a continuous experience based process improvement.

Next we present an overview of the algorithm for the automatic generation of certain types of business rules, algorithm that plays an important role in the flexibility of their deployment. Further information about this can be found in Rosca (1997).

4. Automatic generation of certain types of business rules

The method for the automatic generation of business rules uses the knowledge structure provided by the DSS through decision matrices. Also it uses statistical data which records how well domain assumptions have supported various arguments in the past.

4.1. Abstract business rules

The Business rules extracted from decision structures and statistical data follow the general pattern

WHEN Context IF Cond_1 $[w1]\langle\text{op}\rangle\text{Cond}_2$ $[w_2]$ $\langle\text{op}\rangle\cdots$THEN Action

where context, conditions and actions are represented in terms of decision type variables. The context represents the environment where the *condition* and *action* part of the rule are evaluated. It is represented by a variable from the hierarchy of decision structures. For example, when the action part of the rule is instantiated with the alternative Alt_i, the context of alternative Alt_i is the *issue* for which it represents a possible solution. The *conditions* are predicates over decision type variables. The *action* is an assignment of truth value to a decision type variable. The weight w_i is the static weight factor of $condition_i$.

Rules are interpreted using a production system inference engine. The inference engine computes a dynamic merit for each action according to the general aggregation formula:

$$\text{Merit}_{\text{Action}} = f(\text{Merit}(\text{Cond}_1) \times w_1, \text{Merit}(\text{Cond}_2) \times w_2, \ldots)$$

For example, the aggregation formula f is a weighted sum if the rule condition is a disjunction.

We are introducing three types of abstract business rules that correspond to different levels of detail in the argumentation process. They are the rules obtained at the criteria level, at the arguments level and at the assumptions level of a decision matrix. These abstract business rules correspond to different levels of decision making in the hierarchy of an enterprise. Their combination leads to the generation of operational business rules. The computation of the dynamic merit associated with these abstract business rules is done at operation time, based on the operational data available at that time.

The criteria level rules reason about the degree of satisfaction of the proposed decision criteria by the existing alternatives. They correspond to high level decision making and express objectives in very broad terms. The general format of this type of rule is:

WHEN *Iss* IF $\text{Crit}_1[w_1] \vee \text{Crit}_2[w_2] \vee \cdots \vee \text{Crit}_n[w_n]$ THEN Alt_i

where w_i represents the weight or importance of criterion Crit_j in the process of evaluation of alternative Alt_i (see Fig. 3). Their values should sum up to 1. The w_i values are given by the decision makers at elicitation time.

Whenever this type of rule is applied at operation time the system computes the merit of the alternative given in the action part of the rule (Alt_i), based on the weight of each criterion (w_{Crit_j}) and the merit $\text{Merit}_{\text{Crit}_j}$ of the alternative Alt_i in satisfying each criterion Crit_j:

$$\text{Merit}_{\text{Alt}_i} = \frac{\sum_{j=1}^{n_{\text{crit}}} w_{\text{Crit}_j} \times \text{Merit}_{\text{Crit}_j}}{\sum_{d=1}^{n_{\text{crit}}} w_{\text{Crit}_d}} \quad (1)$$

$\text{Merit}_{\text{Crit}_j}$ will be defined next, while n_{crit} represents the number of criteria defined for evaluating the alternatives of an issue.

The merit of each proposed alternative for solving an issue is calculated at operation time and the alternative with the largest merit is proposed as the solution to the issue. The rule corresponding to this alternative is chosen for deployment.

An example of a criteria level rule from the travel office case study is

WHEN *Solve travel discrepancies* IF *Effective Fraud Detection* = True [0.7] ∨ *Timely Reimbursement* = True [0.3] THEN *Ignore*, *but notify* = *True*

which shows the criteria used for evaluating the alternative *Ignore*, *but notify* the employee, for elucidating the issue Solve travel discrepancies.

The argument level rules express heuristics used in deciding how well an alternative satisfies a criterion when several arguments are presented for, or against it. They combine the evidence about the merit of each argument Arg_k correlated with a pair (Alt_i, Crit_j), in order to compute the $\text{Merit}_{\text{Crit}_j}$ of the alternative Alt_i against criterion Crit_j. These rules express the fact that the achievement of the enterprise objectives is not always obvious, and therefore requires negotiation among stakeholders. The general format of this type of rule is:

WHEN Alt_i IF $\text{Arg}_1[w_1] \vee \text{Arg}_2[w_2] \vee \cdots \vee \text{Arg}_n[w_n]$ THEN Crit_j

where w_i represents the weight (importance) of argument Arg_i in the evaluation process. Weights can take values on a scale of $[-1.0, 1.0]$. When $w_i = -1.0$, Arg_i is a strong-counterargument, while when $w_i = 1.0$, Arg_i is a strong supporting argument.

Whenever this type of rule is applied at operation time the system computes the merit $\text{Merit}_{\text{Crit}_j}$ of alternative Alt_i in satisfying the criterion Crit_j. The computation of $\text{Merit}_{\text{Crit}_j}$ is based on the weight of each argument (w_{Arg_i}) and the predicted accuracy of the truth value of that argument

($\text{Merit}_{\text{Arg}_i}$)

$$\text{Merit}_{\text{Crit}_j} = \frac{\sum_{i=1}^{n_{\text{arg}}} w_{\text{Arg}_i} \times \text{Merit}_{\text{Arg}_i}}{\sum_{i=1}^{n_{\text{arg}}} w_{\text{arg}_i}} \quad (2)$$

where $\text{Merit}_{\text{Arg}_i}$ will be defined next and n_{arg} is the number of arguments elicitated for or against the alternative Alt_i in satisfying the criterion Crit_j. For example, the rule

WHEN *Ignore*, *but notify* IF *Non-effective Detection* = True [−0.2] ∨ *Acceptable Detection* = True [1.0] THEN *Effective Fraud Detection* = True

expresses the fact that the process of refining the meaning of the *Effective Fraud Detection* objective brings up two arguments with different weights in the context of deciding to ignore a fraud, but notify the employee. The arguments correspond to different situations perceived as plausible by various stakeholders.

The assumptions level rules reason about what is true in the application domain. This is the most detailed level of rules where business objectives find their operational meanings. These rules express the operational conditions that need to be met by an alternative Alt_i to meet a criterion Crit_j. Based on domain assumptions, they assess the validity of the arguments Arg_i associated with the pair (Alt_i, Crit_j). The antecedent part of the rules consists of conditions on the assumptions underlying an argument, which is represented in the consequent part of the rules. The conditions in the antecedent part are obtained either automatically by induction from statistical data, or when this data is not available, by interaction with the decision makers.

The condition part of this type of rule is a conjunctive normal form formula that contains various atomic assumptions of the argument under consideration. An atomic assumption has the format $\text{As} \equiv (\text{Attr}\langle\text{op}\rangle\text{value})$, where $\langle\text{op}\rangle = \{ =, , > \}$. For instance, a rule whose condition has two disjunctive terms is

IF ($\text{As}_1 \wedge \text{As}_2 \wedge \text{As}_n$) THEN Arg_i [$\text{Merit}_{\text{Arg}_i}$]

where Attr_i represent the attributes whose values v_i contribute to the truth value of the argument Arg_i. The $\text{Merit}_{\text{Arg}_i}$ describes the certainty factor about the truth value of argument Arg_i. It is a by-product of the decision tree generation algorithm (Quinlan, 1993; Rosca & Rosca, 1989) that will be discussed in Section 5.

For example, the rule

IF (Fraud Type = small ∧ Employee Classification = good) ∨ (Violation Type = rule_violation) THEN Acceptable Detection = True [35.2%]

expresses the operational conditions for the argument 'Ignore, but notify is not effective because the employee can disconsider the notification and commit other violations' to be true. More than that, it shows the perceived accuracy (35.2%) of this assessment, which will be used at operation time in calculating the merit of an alternative. The above conditions were obtained by applying an induction algorithm from statistical data, a sample of which is shown in Fig. 3. The algorithm will be presented next.

4.2. Algorithm for automatic generation of business rules

We distinguish two types of automatically created business rules. The first types are the business rules that capture heuristic knowledge from the decision matrix. These rules correspond to criteria-level and two argument-level from above. The second types of rules are extracted from decision trees induced from statistical data by applying inductive learning techniques. They correspond to assumption-level rules from above.

Decision tree learning is a machine learning technique that starts with a collection of training examples and outputs a compact classification, called a decision tree. Training examples are *n*-tuples specifying the classification of a given combination of attribute values. For example, in Fig. 3, for the attribute values combination *Fraud Type* = small, *Employee Classification* = good, *Violation Type* = rule_violation, the argument *Acceptable Detection* is true. The internal nodes in a decision tree correspond to tests on the values of particular attributes, while the leaves correspond to class values. Decision trees logically correspond to a disjunction of conjunctions. They can be further generalized into business rules at assumptions level using a technique of acquisition of rules from decision trees (Quinlan, 1993; Rosca & Rosca, 1989).

The rules at criteria and arguments levels are generated automatically from the decision structures represented in a decision matrix. A criteria level rule corresponds to a row in the decision matrix, while an argument level rule corresponds to a row in the arguments matrix.

The assumptions level rules are automatically generated by a procedure that ties together the algorithm for the induction of decision trees and the algorithm for rule induction from decision trees (Quinlan, 1993; Rosca & Rosca, 1989).

The training examples are stated in terms of tuples of domain assumptions and associated values. The domain assumptions are represented in terms of a set of attributes (A). The algorithm tolerates missing values for some of the attributes, therefore allowing for imprecise information about an assumption. Classes represent the truth values of an argument. A certainty factor is associated with each class for representing the likelihood of an argument's validity. The set of training examples (E) used for the induction of the decision tree ($\text{DT}(E)$) is extracted from a database of domain assumptions. E refers to a specific argument Arg_k.

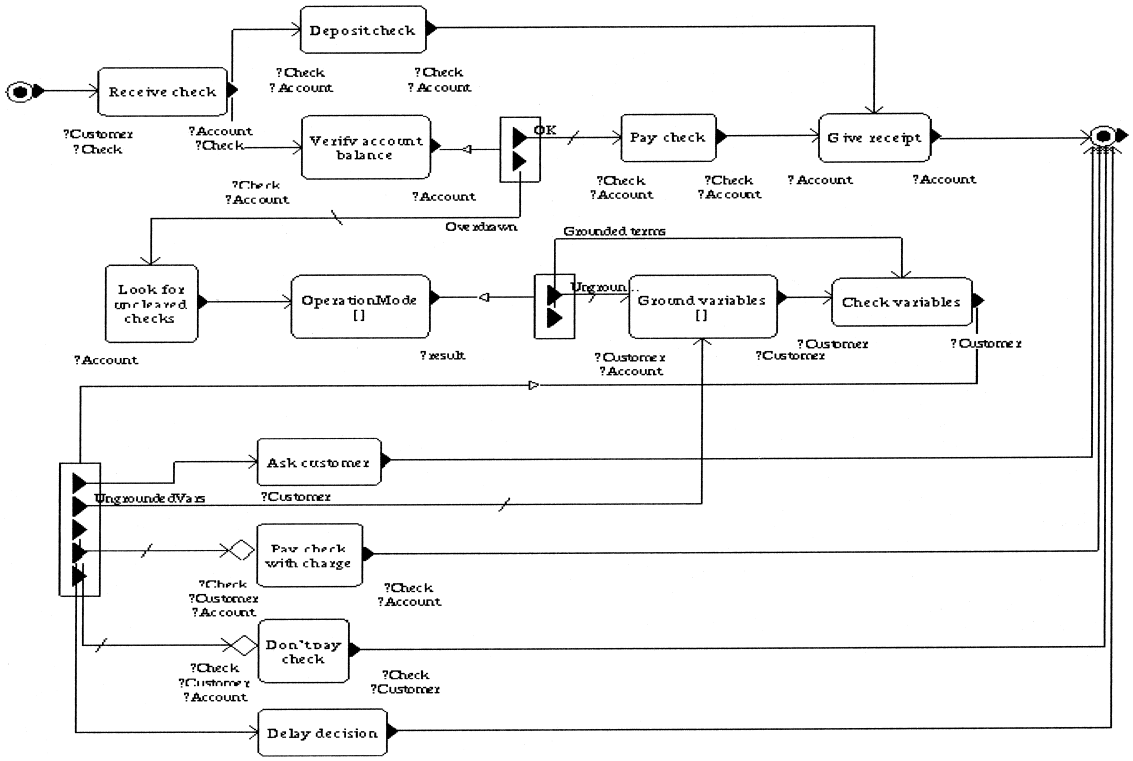

Fig. 4. A Banking process that includes ambiguous rules related to clearing checks.

From an induced decision tree we extract and optimize assumptions level rules. The optimization is applied both at the rule level and at rule sets levels.

The result of the algorithms presented above is a set of rules associated with an issue *Iss*. Each such set of rules can be transformed into an operational business rule that follows the ECA format. The transformation is achieved by applying the operation of *consequent expansion* at criteria and arguments levels. Consequent expansion replaces a condition on a variable *v* in the antecedent of a rule, with the antecedent of the rule that has *v* as a consequent. If several such rules exist they are combined into an OR logical operation.

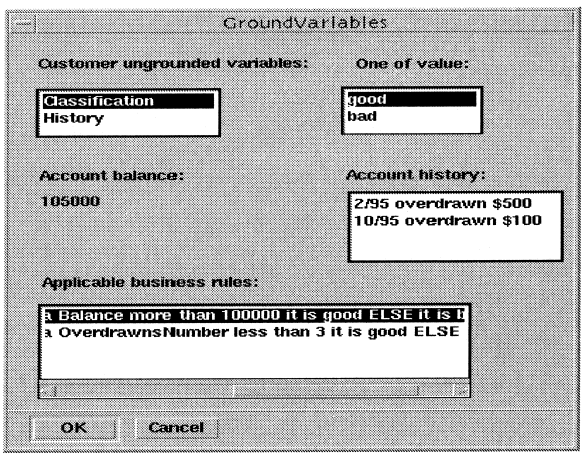

Fig. 5. Grounding guide example.

Following this automatic process, the enterprise objectives stated in criteria level rules are refined to the point where they can be translated into operational business rules that achieve the enterprise goals. The automatic generation was possible due to a deeper understanding of the business rules' origins as formalized in the decision support framework and its relationship with the enterprise model.

5. Business rules deployment

After the operational business rules are identified and integrated into the enterprise business process they are activated in one of the following contexts: *executable contexts* or *choice contexts*, similar with those specified in the NATURE process metamodel (Pohl et al., 1997). Executable contexts represent fragments of a process which can be strictly enforced, either by being automated or by being faithfully executed by people. These contexts are characterized by *deterministic business rules*, in the sense that there is only one applicable business rule and the data referenced by the rule are known with certainty (are unambiguous). Such a rule can be automatically applied either by the underlying information system or by people.

Choice contexts represent fragments of a process where human intervention is required for making a decision. The decision might be needed because there are multiple, conflicting business rules applicable in the context or, because the applicable business rules have

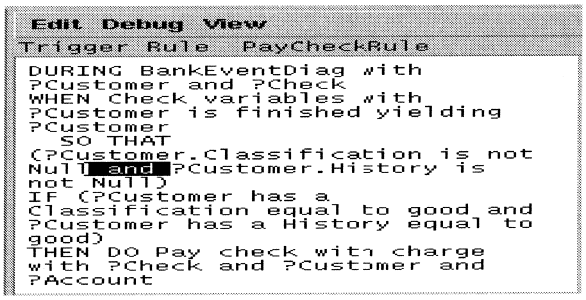

Fig. 6. Ambiguous rule due to ungrounded terms *good Classification* and *good History*.

ambiguous terms. BRADES associates these contexts with *non-deterministic business rules* that need *operational decisions* in order to be activated.

When a context is characterized by multiple, conflicting business rules, a decision support step is introduced in the process. This step involves the display of the decision matrix associated with the conflicting rules (see Fig. 3). For example, in the travel office case study presented earlier, the clerk needs to decide which one of the following two rules to apply

IF (*Fraud Type* = small ∧ *Employee Classification* = good ∧ *Violation type* = rule_violation) THEN *Ignore* = *True*

IF (*Fraud Type* = small ∧ *Employee Classification* = good ∧ *Violation type* = rule_violation) THEN *Ignore, but notify employee* = *True*

in order to solve discrepancies found in the travel documentation handed in by an employee.

The decision matrix displays *operational values* for the merit of each alternative and arguments, dynamically computed according to the formulas 1 and 2 and by the algorithm for induction of decision trees discussed in Section 4. BRADES uses operational values of the assumptions of the current case in the rule generated from the decision tree corresponding to that argument, to compute the dynamic merit of each individual argument. These rules have been obtained by applying the algorithm for induction of rules from statistical data about domain assumptions. The algorithm implicitly computes the certainty factor of the truth value of an argument, which is used in the computation of the dynamic merit of an argument. Furthermore, the dynamic merit of each argument is used for the computation of the merit of each alternative against a single criterion, according to formula 2. The overall merit of each alternative is calculated at operation time by combining the merits of an alternative against each criterion proposed initially, using the formula 1. By consulting the decision matrix, the user can choose the best rule proposed by the system, based on the merit of its associated alternative. Otherwise, the user can apply his own judgment and select another rule, or propose a new rule. This way the decision of which rule is the best has been shifted from analysis time to operation time.

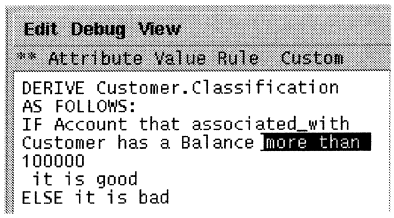

Fig. 7. Definitional rule.

When a context is characterized by ambiguous business rules, e.g. they contain ungrounded terms whose grounding could not be done with certitude at analysis time due to subjective definitions, a process of instantiation or disambiguation is needed. BRADES makes this process possible due to the links among the business rules, the Decision Space and the Enterprise Model. The definitional business rules attached to various attributes of the entities in the Enterprise Model become available for consulting. The user can choose one of the legal values of an ungrounded term, based on definitional business rules, or can disagree with those rules and choose a value according to his own judgment. After the grounding has been completed, the control in the process model resumes its normal flow. For example, modeling the services of a bank, one had to solve the issue of handling 'unauthorized overdrawn due to uncleared checks'. As shown in Fig. 4, four alternatives have been proposed for solving this issue: *Ask Customer* about the problem before applying a charge on his account, *Pay check with charge*, *Don't Pay Check*, and *Delay Decision* for allowing some more time for the problem to be fixed following the normal procedures.

If the intervention of a clerk is needed at operation time, he is able to choose among the four alternative rules, that would trigger the execution of one of the four corresponding operations. If one or more rules contain ambiguous terms (see Fig. 6), a grounding guide is displayed to help with the disambiguation of the rules. An example of a grounding guide is associated with the operation *Ground variables* in Fig. 4. Operations such as 'Ground variables', 'Check variables' and 'Operation Mode' were introduced to implement our approach in LiveModel. The grounding guide for this case, necessary for grounding the terms *Customer.Classification* and *Customer.History* is shown in Fig. 5. *Classification* and *History* are attributes of the object Customer from the Enterprise Model. An example of a definitional business rule used in this grounding guide is shown in Fig. 7.

The grounding guide indicates the parameters one should look for when making a decision and reflect the fact that, in general, a decision is not made in a vacuum of knowledge. This approach permits development to move forward even when requirements are not fully understood, or their understanding cannot be done until operational data is available.

Monitoring of the conditional decision is an important part of the methodology, but further explanation is beyond the scope of this paper.

6. Summary and conclusions

We have presented a methodology that addresses aspects that cover the entire lifecycle of business rules. The methodology is based on a high-level architecture that views the business rules as a bridge between the enterprise model and the operational system of an enterprise. Because we see business rules as decisions about how a business has decided to carry out its work to provide services to its customers, we have proposed a decision support component of the architecture that plays an important role in all phases of the business rules lifecycle.

This paper has identified and discussed solutions for one phase of the business rules lifecycle, their deployment, which includes dealing with conflicting and ambiguous situations. We have showed the possibility for automated assistance for both cases. Specifically, for dealing with conflicting rules we have proposed a customizable approach that is based on operational data captured in a DSS, and on an algorithm for the rapid generation of business rules. The algorithm is based on the induction of rules from a set of domain assumptions and data stored in decision structures, a well-known technique borrowed from the Machine Learning field. To the best of our knowledge, this is the first attempt of using this technique in the field of Software Engineering.

For dealing with the ambiguous situations, we have shown how the definitional business rules contained in a grounding guide, together with other information extracted from the Enterprise Model and Decision Space, can help in the grounding of ambiguous terms, allowing a flexible application of the business rules.

Our approach adopts the guidance strategy for handling inconsistencies, starting from the observation that restoring consistency is not always possible at analysis time (Balzer, 1991; Nuseibeh & Easterbrook, 2000). Therefore, we supply guidance and information available only at operation time, and let the users decide how to act when the inconsistency arises, or when insufficient information has been provided at analysis time. This is a less formal approach then one could find in the viewpoints or the KAOS frameworks (Finkelstein, Gabbay, Hunter, Kramer, & Nuseibeh, 1994; van Lamsweerde, Darimont, & Letier, 1998), but it might be more appropriate for the business-oriented people who are intended to express and manage business rules.

The architecture proposed has been implemented in the LiveModel modeling platform, and some case studies have been acquired. Samples of them have been shown in this paper. Also, the algorithm for the rapid generation of business rules has been implemented. Now we intend to study the scalability of this work to real-world examples, and we would like to do that in the context of the CMM certification of an enterprise.

References

Balzer, R. (1991). *Tolerating inconsistency*. *Proceedings ICSE'91*, IEEE Computer Press, pp. 158–165.

Boehm, B., Bose, P., Horowitz, E., & Lee, M. J. (1995). *Requirements negotiation and renegotiation aids: A theory-w based spiral approach*. *Proceedings ICSE'95*, IEEE Computer Society Press, pp. 23–30.

Bubenko, J., Jr, & Wangler, B. (1993). Objectives driven capture of business rules and of information systems requirements. *Proceedings of the International Conference on Systems, Man and Cybernetics*, 670–677.

Champion, R., & Moores, T. (1996). *Exploiting an enterprise model during systems' requirements capture and analysis*. *Proceedings of the International Conference on Requirements Engineering ICRE'96*, IEEE Computer Society Press.

Chen, Y., Segarra, G., & Chong, P. (1992). Visualizing business rules in corporate databases. *Industrial Management and Data Systems*, *92*(7), 3–8.

Easterbrook, S., & Nuseibeh, B. (1995). *Managing inconsistencies in an evolving specification*. *Proceedings RE'95*, IEEE Computer Press, pp. 48–55.

Feuerlicht, G., & Blair, A. (1990). An architecture for managing business rules using knowledge engineering techniques in RDBMS environment. *Proceedings of the Fourth Australian Joint Conference on Artificial Intellligence*, 384–393.

Fickas, S., & Feather, M. (1995). Requirements monitoring in dynamic environments. *Proceedings of IEEE International Symposium on Requirements Engineering*.

Finkelstein, A., Gabbay, D., Hunter, A., Kramer, J., & Nuseibeh, B. (1994). Inconsistency handling in multiple-perspective specifications. *IEEE Transactions in Software Engineering*, *20*(8), 569–578.

Ghezzi, C., & Nuseibeh, B. (1998). Special issue on managing inconsistency in software development. *IEEE Transactions on Software Engineering*, *24*(11), 906–907.

Greenspan, S., & Feblowitz, M. (1993). Requirements engineering using the SOS paradig. *Proceedings of IEEE International Symposium on Requirements Engineering, San Diego*.

Grosof, B., & Labrou, Y. (1999). *An approach to using XML and rule-based content language with an agent communication language*. Technical Reports RC 21491, IBM TJ Watson.

GUIDE International Co (1997). The GUIDE business rules project, final report.

von Halle, B. (1993). Making business rules real. *Database Programming Design*, 13–15.

Herbst, H. (1995). A meta-model for specifying business rules in system analysis. *Proceedings of CaiSE'95*, 186–199.

Intellicorp (1995). LiveModel User's Guide.

Kilov, H., & Simmonds, I. (1997). Business rules: From business specification to design. Technical Report RC 20754, IBM TJ Watson.

van Lamsweerde, A., Darimont, R., & Letier, E. (1998). Managing conflicts in goal-driven requirements engineering. *IEEE Transactions in Software Engineering*, *24*(11), 908–925.

Leite, J., & Leonardi, C. (1998). *Business rules as organizational policies*. *Proceedings of the International Workshop on Software Specifications and Design*, IEEE Computer Society Press.

Loucopoulos, P., & Katsouli, E. (1992). Modelling business rules in an office environment. *SIGOIS Bulletin*, *13*(2), 28–37.

Martin, J., & Odell, J. (1995). *Object-oriented methods: A Foundation*. Englewood Cliffs, NJ: Prentice Hall, Chapter 20.

Nuseibeh, B., Easterbrook, S., & Russo, A. (2000). Leveraging inconsistency in software development. *IEEE Computer*, 24–29.

Paulk, M., Curtis, B., & Chrisis, M. (1993). *Capability maturity model for software*, version 1.1, Technical Report, Software Engineering Institute.

Pohl, K., Domges, R., & Jarke, M. (1997). Towards method-driven trace capture. *Proceedings of CaiSE'79*.

Quinlan, J. (1993). *C4.5: programs for machine learning*. Los Altos, CA: Morgan Kaufmann.

Rosca, D. (1997). *A decision making methodology in support of the business rules lifecycle*. PhD Thesis. Computer Science Department, Old Dominion University.

Rosca, J., & Rosca, D. (1989). Knowledge acquisition facilities within an expert system toolkit. *Proceedings of the Seventh International Symposium on Computer Science, Jassy, Romania*.

Rosca, D., Greenspan, S., Feblowitz, M., & Wild, C. (1997). A decision making methodology in support of the business rules lifecycle. *Proceedings of the International Symposium on Requirements Engineering RE97 Conference*, 236–246.

Rosca, D., Greenspan, S., Wild, C., Reubenstein, H., Maly, K., & Feblowitz, M. (1995). Application of a decision support mechanism to the business rules lifecycle. *Proceedings of the Knowledge Based Software Engineering KBSE'95 Conference*, 114–122.

Yu, E. (1994). *Modelling strategic relationships for process reengineering*. PhD Thesis. University of Toronto, December.