A Decision Making Methodology in Support of the Business Rules Lifecycle

Daniela Rosca* Sol Greenspan† Mark Feblowitz† Chris Wild*

Abstract

The business rules that underlie an enterprise emerge as a new category of system requirements that represent decisions about how to run the business, and which are characterized by their business-orientation and their propensity for change. In this paper, we introduce a decision making methodology which addresses several aspects of the business rules lifecycle: acquisition, deployment and evolution. We describe a meta-model for representing business rules in terms of an enterprise model, and also a decision support submodel for reasoning about and deriving the rules. A technique for automatically extracting business rules from the decision structure is described and illustrated using business rules examples inspired by the London Ambulance Service case study. A system based on the metamodel has been implemented, including the extraction algorithm.

1 Introduction

Many requirements fall into a category known as *business rules*, which express computational requirements that determine or affect how a business is run. Business rules address how customers are to be treated, how resources are to be managed, or how special situations are to be handled in carrying out business processes. They represent decisions about how a business has decided to carry out its work to provide services to its customers. As such, they comprise an important set of requirements on any system being developed or procured for the enterprise. We prefer to view the combination of (legacy, COTS, and new) systems being employed to run the enterprise as a single operational system that is governed by business rules.

Business rules can be viewed as expressing functional and nonfunctional requirements which are, in principle, no different than other kinds of requirements. However, business rules are characterized by their strategic importance to the business and consequently deserve special consideration. They emphasize certain characteristics that imply how they should be dealt with.

For one thing, they are intended to be expressed and managed by business-level people and therefore need to be expressed in terms of an enterprise-level model [9, 1], not in terms of programs and databases. For another, the appropriate notion of "meeting the requirements" is, for business rules, a rich and complex one, involving satisficing, tradeoffs, exceptions, or other coping strategies. The "implementation" (we prefer the term "deployment") of a business rule is not so straightforward, certainly not just the usual meaning of a system meeting its specifications.

The most demanding characteristic of business rules is that the business is likely to want to change them. Business rules are formulated to have planned consequences that contribute to the success of the business, and therefore it is expected that the rules will be continually re-evaluated and subject to change to improve the performance of the business. Business rules can also be generated by or in response to external sources, such as regulatory bodies, the law, market forces, or physical realities. Since the business does not have control over these external forces, the enterprise must be prepared to change the business rules that govern its operation.

When business rules change, the operational system needs to change, which can require major investment for maintenance and evolution. This is a source of the infamous "legacy" problem, which presents high risk not only due to costs but also due to difficulty of meeting time deadlines. Of course, this problem of adapting systems to changing requirements is a classic and pervasive one, but for business rules the problem is hopefully more focussed and more tractable than the general case.

The paper presents aspects of a methodology that was developed to cope with these issues by prescribing an approach for managing business rules over their lifecycle (i.e., over the lifecycle of the enterprise and its operational system).

The methodology relies on a conceptual modeling framework or metamodel, presented in Section 3, which prescribes representations for the enterprise model, the business rules, and a decision space. The needed support for requirements analysis is then defined in terms of relationships between the submodels.

The methodology consists of three overall activities, presented in Section 4, which are motivated by the issues raised above. These are:

- **Business rules acquisition** Facilitating expression and management of business rules at an enterprise level.
- **Business rules deployment** Getting the system to follow the rules.
- **Business rules evolution** Facilitating change without major re-development effort.

Acquisition includes not only acquiring the business

*Department of Computer Science, Old Dominion University, Norfolk, VA 23529, {rosca,wild}@cs.odu.edu

†GTE Laboratories Incorporated, 40 Sylvan Road, Waltham, MA 02254, {greenspan,feblowitz}@gte.com

rules and the enterprise model in terms of which they are stated, but also capturing the deliberation process that arrived at the rules. Deployment includes the resolution of issues such as determinism, conflicts, and ambiguity due to ungrounded terms, and also an evaluation of how well the business rules are expected to support enterprise goals.

Acquisition includes, in addition to populating the business rules environment, the automatic extraction of deployable business rules from the decision space. The *business rules extraction* method, in Section 6, is based on an induction algorithm that uses knowledge from the decision support system and statistical data about domain assumptions. Section 5 introduces the example (the London Ambulance Service) used to illustrate the method.

A system has been implemented for acquisition of models according to the framework described in this paper. While the particulars of this system are not the subject of this paper, it should be noted that this system has been populated with the models and data presented in this paper, and the extraction algorithm has been implemented. The results presented below were produced by running the system on the data.

More background on business rules is given in Section 2. There the reader will find more explanation of business rules, additional motivation for the approach taken by the methodology, and insights gained from studying a sampling of a few hundred business rules associated with three years of business process re-engineering projects.

2 Business Rules Perspective

2.1 Requirements Analysis for Business Rules

Business rules are requirements that arise from the business objectives of an enterprise. A bank requires a supervisor's signature before cashing a check over $5000 to minimize loss in case of fraud. A customer service organization, trying to resolve customer complaints by telephone, tries to avoid transferring the customer's phone call more than once and hopefully not at all (this is called the "two-touch" rule, and is intended to minimize customer inconvenience). A travel agent has been contracted to provide a client corporation with "the lowest possible airfares" which helps the client corporation achieve financial goals. A company requires all departments to remain within 10% of budget, provided the total company budget is met within 1%, also to meet financial goals. In general, business rules are statements about the enterprise's way of doing business. They reflect policies, procedures or other constraints on ways to satisfy customers, make good use of resources, conform to laws or business conventions, and the like.

Business rules are presumed to be of strategic/tactical importance to the success of the business. This means, among other things, that they are to be formulated and analyzed by business-oriented people, not by information technologists or software engineers. Decisions about cost, quality, responsibility, and good service are not supposed to be delegated to system designers or programmers, but are to be implemented/deployed after they are decided upon by the appropriate parties. In reality, it is frequently the case that the only way to determine an organization's business rules is to look at the operational system; how does the Customer Service Rep schedule a repair visit? How does the software calculate the insurance reimbursement? You know there is a problem if you have to "look at the code," i.e., when there is no other explicit statement of what the business rules are. This has motivated discussion of "mining" business rules [11] through a kind of reverse engineering activity on the software. However, this can be from difficult to impossible, since the manifestation of the rules might be scattered all over the code, and since the belief that there were conscious business rules in the first place might be flawed.

Figure 1: Setting Business Rules in context

As shown in Figure 1, *business rules* are positioned between *enterprise objectives* and the *operational system* of the enterprise. The operational system is the combination of the hardware, software and humans that carry out the work of the enterprise. Enterprise objectives are the general aspirations of the enterprise: "make (more) money," "make customers happy," and "keep shareholders satisfied with their investment," etc. They are the ultimate source of all requirements on the operational system. The enterprise objectives can be decomposed and refined, eventually to the point where they can be translated into *business rules* that can achieve the goals. These business rules are intended to be carried out by the operational system, with the intent that the "deployed" or "operationalized" rules will realize the enterprise objectives.

Because of the ambiguities and conflicts inherent in the enterprise objectives, business rules are not simple refinements of the objectives; rather, business rules represent decisions that are made about how to achieve the enterprise objectives. Business rules become requirements that govern the operational system of the enterprise and determine how the business is actually run. To answer the central question, "does the operational system satisfy the business objectives (and to what degree)?", one must simultaneously address two subquestions, which are subjects for requirements analysis involving business rules:

- Decision-making: Do the business rules embody the right (or best) decisions about how to achieve the objectives of the business?
- Operationalization: Does the operational system satisfy the requirements represented in the business rules?

Decision-making requires one to choose among alternative business rules for achieving objectives, considering the expected influences or consequences - both in support of and in denial of the objectives, based on various assumptions. As described earlier in [14], we propose a decision support system/structure (DSS) for

capturing the reasoning associated with business rules. (The DSS of [14] has since then been refined and implemented.) In addition to supporting the reasoning for choosing business rules, the DSS supports *business rules extraction*, presented below, which produces a business rule from the information in the DSS.

Operationalization of business rules requires a strategy (built into the enterprise infrastructure) and tactics (methods such as transformations) to deploy the business rules in the operational system. For each type of business rule in the taxonomy, an approach is needed for deployment, preferably with minimal time and expense. Some classes of business rule are directly deployable, while others require varying degrees of transformation or implementation; still others have no direct path into operation, and must be approximated in other ways. Deployment of business rules is an important part of the methodology, but further explanation is beyond the scope of this paper.

If the analysis is correct, i.e., if the business rules decisions and operationalizations are both done without error and with total insight, then the operational system should satisfy the enterprise objectives. This seems to be the presumption usually made in requirements engineering (RE), i.e. that the requirements are valid, and that if they are valid and the implementation is correct, then the overall objectives have been met. However, this picture is too simplistic for the real world, and we need to step outside of this picture to see what else is needed. In the first place, it cannot be assumed that enterprise objectives are devoid of conflicts; a set of rules that supports conflicting goals cannot be implemented without a hitch. The implementation will have to consider tradeoffs, not all of which have been identified by the time the system is in operation. The usual RE presumption is that all of the goal conflicts have been worked out as a prerequisite to system design. We posit that this will never be the case, at least for service-providing enterprises with which we are concerned.

Furthermore, the reasoning upon which the business rules are decided is always based on assumptions about what will influence what, and what will be the consequences. These assumptions could be wrong. Even if all of the assumptions were correct and all of the relevant business rules were followed (i.e., correctly deployed), there might be further reasons why the operational system does not meet some of the business objectives; there could be missing business rules or something missing from the reasoning process. And all of this reasoning and decision making rarely occurs in the context of a fully stable environment; objectives change, as do business conditions, and strategic and tactical decisions.

It should be recognized that, just as the system can never really completely meet its requirements, the state of the reasoning about the system will also never be complete, totally correct or consistent. There should be a constant attempt to revise the business rules and their operationalizations, based on observations of the operational system and its achievement of the objectives. To this end, the methodology prescribes instrumenting the system with monitoring to check achievement of enterprise objectives, evaluating the validity of the assumptions underlying the business rules.

The saying "Rules were made to be broken" applies here. Rules will indeed be broken, either because the rules conflict with other rules, or the implementation is not complete, or the data needed is not available in some situation.

2.2 A Study of Business Rules and Other Related Work

The motivation for our work on business rules came from three years of requirements efforts associated with business process re-engineering projects. On these projects, information referred to as business rules was considered important to gather and document, but there was no particular method for doing so. There were requirements modeling methods and tools being used, but business rules were not part of the modeling framework. They had to be added as annotations to business process models or just captured separately in notebooks.

We studied a sampling, from one project, of some 250 statements referred to as business rules to try to understand better what kinds of information were being captured. An attempt to classify them produced the taxonomy shown in Figure 2. This taxonomy is not claimed to have any scientific validity but only indicates what kinds of business rules were present in this particular application.

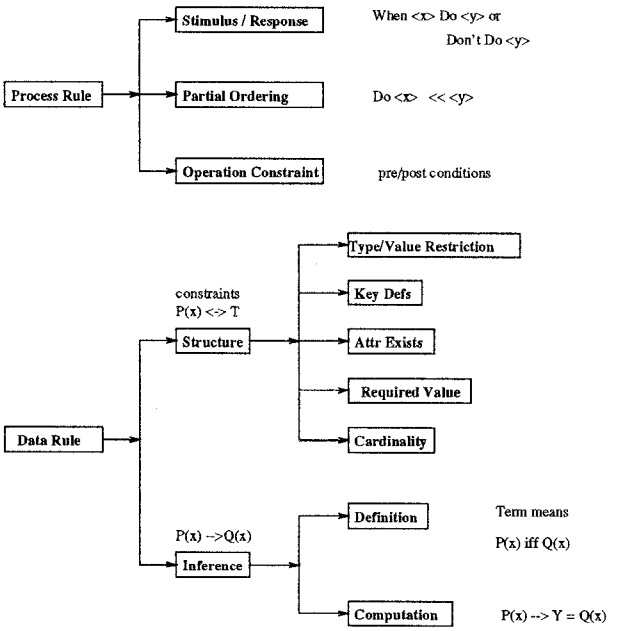

Figure 2: A Taxonomy of Business Rules

The taxonomy is divided into rules for processes and rules for data. A more detailed explanation and analysis of the taxonomy is beyond the scope of this paper, and we assume that the rule categories shown are sufficiently self-explanatory to the reader. The classification is quite close to the classification made

in [10] and to that used in our implementation. In this sense, there seems to be some agreement on what types of business rules are to be encountered in everyday life.

However, a high percentage of the rules were extremely simple (attribute value constraints and cardinality constraints). These rules seemed too low-level to have an important impact on the business. We suspect that these types of rules dominated because of a reliance on an Entity-Relationship style of thinking. Process-oriented rules were less popular, but among them the stimulus response rules were the most numerous. These are also fairly simple rules, with a clear operational semantics. We suspect that the other kinds of rules are just as important but are more difficult for business-oriented people to specify and reason about. There are also other kinds of rules, not present in the taxonomy, that would be useful to have.

In our survey of the literature and trade press, we have found descriptions of business rules that were similarly narrow, tied to what can be achieved with a specific model construct or implementation mechanism. For example, just as in our study, cardinality constraints in Entity-Relationship models are commonly referred to as business rules, despite their extremely limited expressiveness. Other discussions of business rules talk about integrity constraints and trigger rules, which are expressible and enforceable with current database systems. These are good examples of common rule types with direct deployment strategies; the same needs to be done for the other rule types.

Business rules have received a lot of attention in the trade press and other literature as holding the answers to many information technology problems. Some of the literature simply advocates the idea that it is important to spend the effort to give conscious consideration to business rules (as opposed to not doing it at all). Even just gather a list of English language statements [16] is considered a good first step. Other proposals are more structured, relying on the syntax of E-R diagrams, or in the case of [13], a much more elaborate diagrammatic syntax. Still others propose a conceptual model approach for describing an enterprise, with a notation for specifying rules that further specify the requirements on the enterprise [9, 1]. This is closest to our approach, although our current enterprise model is limited to process and data/object rules, as in [10].

Some of the elements of our methodology have been discussed to some extent in the Requirements Engineering literature. For example, in [2] nonfunctional requirements are treated as goals that have to be met through a decision-making process in which change is expected. In [8], a goal-directed requirements elaboration methodology attempts to cope with the "deidealization of unachievable goals" and also models assumptions attached to goals. In [4], 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. These are examples of the work we are trying to bring to bear on business rules.

3 A Metamodel for the Business Rules Environment

The methodology supporting the business rules lifecycle consists of

- a modeling framework (metamodel),
- a prescription of activities for populating the models, and
- techniques for using the information for requirements analysis, including continuous lifecycle support.

This section describes the metamodel, in preparation for describing the methodology activities in the next section. The metamodel described here has been implemented in our experimental requirements modeling/analysis environment for supporting the methodology. The environment consists of three submodels: the *Enterprise Model*, the *Business Rules*, and the *Decision Space*. 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* submodel represents the business rules themselves. The *Decision Space* submodel 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 Figure 3).

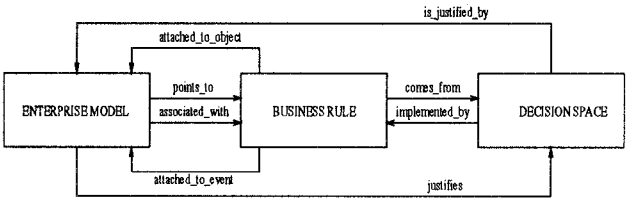

Figure 3: The Business Rules Environment

3.1 Enterprise Model

For the representation of the *Enterprise Model* we have chosen the paradigm of the LiveModel modeling environment [7]. In LiveModel, an enterprise is represented in terms of "objects" and "processes" (see [10]). Objects are represented by a set of Object Diagrams that are essentially Entity-Relationship diagrams (the Object diagram for a fragment of the LAS example can be seen in Figure 4). The business processes are represented by a set of Event Diagrams which define the sequence of operations for process execution (the corresponding Event Diagram for a fragment of the LAS example can be seen in Figure 5.) These Event Diagrams model a hierarchy of business processes, decomposing each operation in a diagram, if necessary, into a more detailed diagram. 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 Process diagrams.

3.2 Decision Space Submodel

The Decision Space submodel is shown in figure 6. It represents the primitives of an issue-based decision

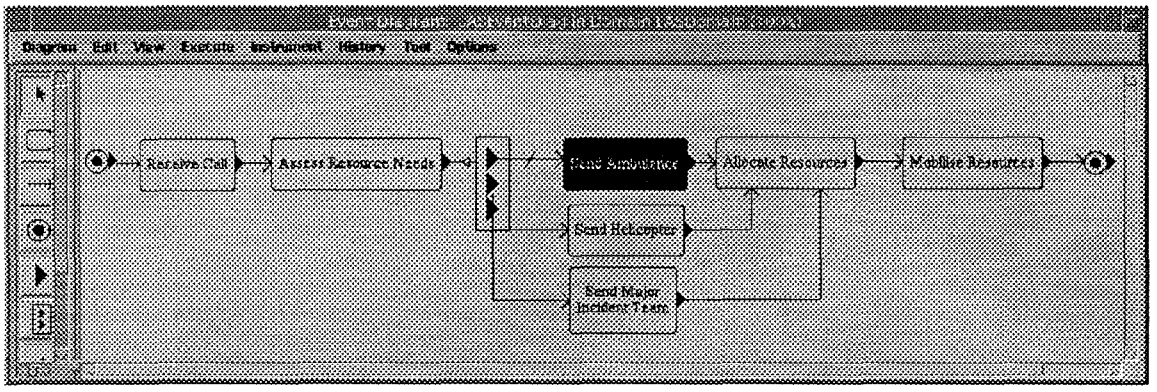

Figure 5: An Event Diagram for a fragment of the LAS example

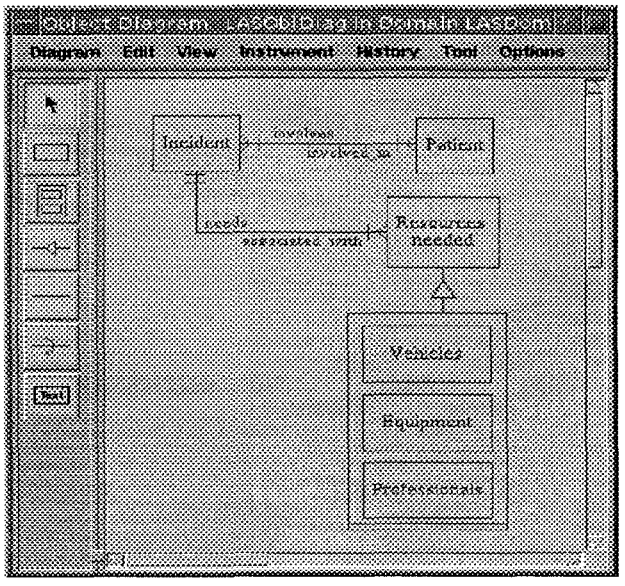

Figure 4: An Object Diagram for a fragment of the LAS example

support model. We can interpret our decision support model as follows. Both functional and non-functional requirements generate *issues* that need to be solved. 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 counter-arguments 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 decision support system (DSS) format. All of the information content of the above primitives can be retrieved from the *decision matrix* associated with a specific issue. A more detailed description of this model and related work on decision support structures is given in [14]. Here we show an augmented model with the links to other submodels of the business rules environment: the *Enterprise Model* and the *Business Rules*.

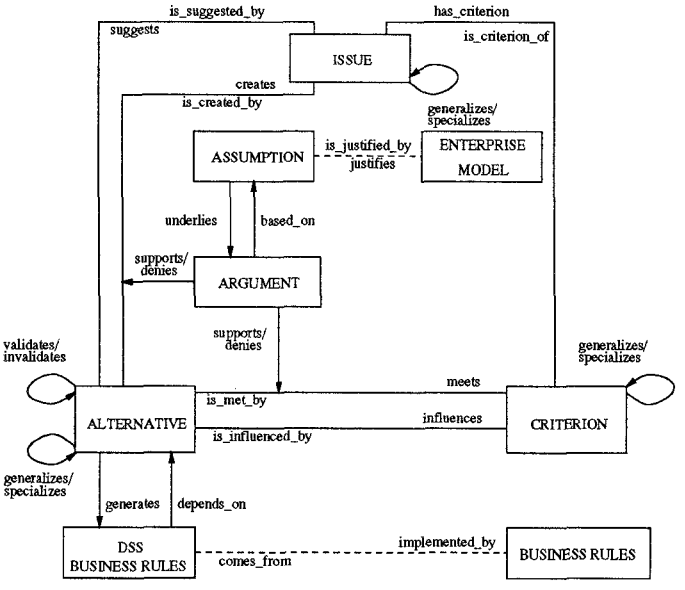

Figure 6: Decision Space Submodel

3.3 Business Rules Submodel

Business rules take the form of event-condition-action (ECA) rules, which we adopted from [6]. The ontology of the Enterprise Model, when examined at a more detailed level, contains events, conditions and actions. Whether a process/object enterprise model is used, as in this paper, or the extended SOS model described in [5] is used, ECA rules provide a convenient "assembly language" into which most kinds of rules can be translated. Since the ECA rules have a well-defined operational semantics, it has been straightforward to build an interpreter for them.

For the simplest form of an 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 objects in the Enterprise Model.

ECA rules are more generally applicable than they might first appear. As discussed in [1], where similar rules are used, judicious interpretations of special cases (such as default meanings for omitting one of the components of the rule) allow ECA rules to express several of the business rules types in the taxonomy. Additionally, an ECA rule can be used to express non-operational semantics, such as the situated enterprise objectives expressed at the *criteria level* in section 6.2.

3.4 Intermodel Relationships

Requirements analysis can be done by analyzing interrelationships between the submodels. Based on the links between these submodels the following types of analysis can be performed:

Business Rules $\longrightarrow$ Enterprise Model:

Which process component(s) does a business rule define/constrain/govern? Which (event/action) operations operationalize a business rule? What object types are referred by a business rule? This information can be used for an impact/sensitivity analysis when a rule changes.

Enterprise Model $\longrightarrow$ Business Rules:

In which business rules does a specific object type participate? This information can be used for impact/sensitivity analysis when the status of an object changes.

What business rules define/constrain/govern a specific process component? This information can be used for business processes improvement.

Business Rules $\longrightarrow$ Decision Space:

Where does the rule come from? This links a business rule to the issue that has generated it. Thus one can have a comprehensive picture of the business rule rationale by looking at the alternatives, criteria, arguments and assumptions that have been stated during the deliberation of that business rule.

Decision Space $\longrightarrow$ Business Rules:

What business rules address a specific issue? This information allows an impact/sensitivity analysis when factors like Government regulations, company policies, etc. change. It is also a useful source of information for a reuse process.

Decision Space $\longrightarrow$ Enterprise Model:

What object types/attributes are addressed by a decision/issue? This can be useful for an impact/sensitivity analysis when a decision is changed.

Enterprise Model $\longrightarrow$ Decision Space:

What decisions/issues involve this object or attribute? What decisions are affected when an object changes?

4 Methodology

The methodology we propose spans all phases of the business rules lifecycle: acquisition, deployment, change in response to changes in internal or external influences and change based on evaluation of the degree of requirements satisfaction.

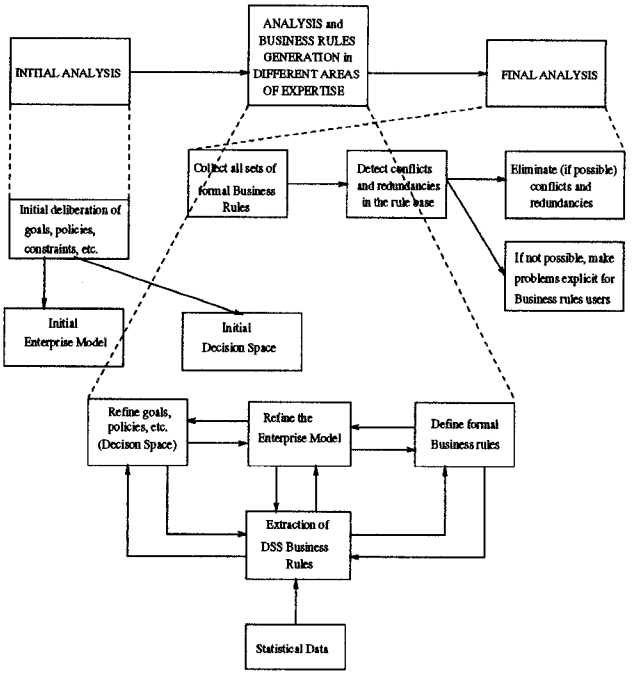

Figure 7: Business Rules Acquisition

4.1 Business Rules Acquisition

We see three major steps in the acquisition of business rules: the initial analysis, the analysis and generation of business rules in different areas of expertise, and final analysis (see Figure 7). During the initial analysis, brainstorming sessions take place for deliberating which are the goals, policies and constraints of the business that need to be modeled. As a result of these deliberations, initial versions of the enterprise model and decision space are sketched and also a first set of business rules that specify how the business should be run is defined. Because they define the goals of the enterprise, these are *strategic business rules* (in the next section, we will find these rules at the *criteria level* rules) that express very high level decisions. These rules need to be refined in order to become operational.

The first step in the refinement of business rules is the analysis and rule generation in different areas of expertise. In this phase business analysis is carried out by separate groups of people, with different areas of expertise, for refining the understanding of business entities, processes and business rules. These activities imply more detailed discussions on the ways of achieving the goals, policies and constraints of the business. They can be complemented with interviews with domain experts and/or reading existing documentation and information related to the subject of analysis. As the understanding of the enterprise objectives becomes clearer the Enterprise Model and the Decision Space are updated.

Based on the entities and processes stated in the Enterprise Model, on the decision structures captured in the Decision Space, and on statistical data from the enterprise's way of doing business, business rules

(decision support system level rules, or DSS rules) can be automatically extracted following an algorithm described in detail in the next section. These rules (called *arguments*, respective *assumptions level rules*) are more concrete than the strategic rules. They underly the structure of operational rules in ECA format that are expressed in a formal rule language, like the one used in the Livemodel tool. At the end of this step, a formal business rule will be defined for each alternative solution in the Decision Space and will be available for deployment. For process simulations, these rules can be attached to operation triggers in process diagrams, like the ones defined in Livemodel. See the slash mark on the Figure 5 for an example of rules attachment to an event diagram and Figure 8 for an example of stimulus/response (trigger) rule implemented in Livemodel (the example has been oversimplified for presentation purposes).

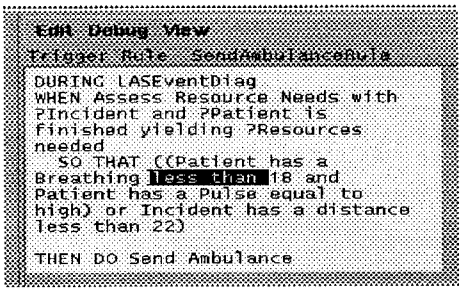

Figure 8: Example of stimulus/response (trigger) rule in the LAS case study

There can be multiple iterations on each operation of this step until a stable set of business rules, as well as a clear and comprehensive Enterprise Model for each specific area of expertise, are obtained.

During the final analysis of the business rules acquisition all of the existing sets of business rules, Decision Spaces, and parts of the Enterprise Model are put together, leading to the detection of redundancies and conflicts. The detection is facilitated by the types of analyses discussed in section 3. These redundancies and conflicts are either eliminated, or if not possible, made explicit to the designers, developers and users of the information system that will underlie the business and that will incorporate these business rules, or to the users of people oriented business rules.

4.2 Business Rules Deployment

After the business rules are defined and integrated into the enterprise process model they become operational. Therefore, whenever a new case is run through a process model inside the enterprise, there are a couple of situations that can arise in the application of business rules (see Figure 9):

1. The situation is deterministic, e.g. characterized by a single business rule and the data referenced by the rule are known with certainty. Therefore that rule can be automatically applied by the underlying information system, either by people or by machine.

2. The situation is characterized by multiple, conflicting business rules. In these cases we can show the decision matrix associated with those rules and let the

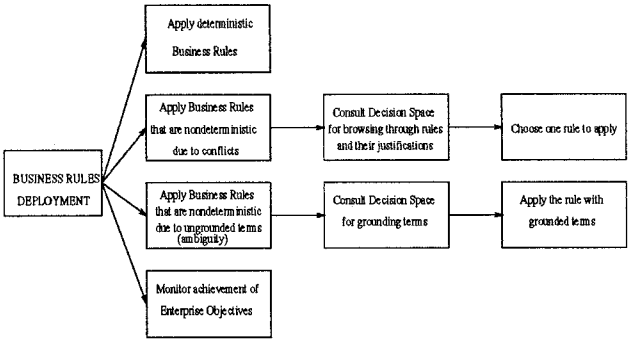

Figure 9: Business Rules Deployment

user browse through it, analyze the information contained in the decision structures and assess the merit of each alternative associated with each rule. The user can choose one of the proposed rules or apply his own judgment and select another rule. This way the decision of which rule is the best has been shifted from the analysis time to operation time, when concrete data about a case is available.

3. If the applicable rule(s) is(are) ambiguous, e.g. they contain ungrounded terms whose grounding couldn't be done with certitude at analysis time we can show the interpretation of these terms using the links among the business rules, the Decision Space and the Enterprise Model. This way the definitional business rules attached to various attributes of the entities in the business can be made available for consulting. The user can choose one of the legal values of an ungrounded term based on the definitional business rules or can disagree with those rules and choose a value according to his own judgment. This approach permits development to move forward even when requirements are not fully understood.

4. For evaluating how well the enterprise objectives are achieved we propose instrumenting the system with monitoring to check whether the assumptions underlying the business rules are valid. This information is fed back to the system for updating the business rules, enterprise model and decision space.

4.3 Business Rules Evolution

There are several possibilities for improving the business rules based on the information captured in the methodology framework. Data obtained through monitoring of the operational system can be used to study the validity of assumptions recorded in the Decision Space, leading to changed rules. New sources of information, both inside and outside of the enterprise, may arise. New solutions may be chosen by users for resolving conflicting or ambiguous situations. For example, by studying the Decision Space one can detect that some criteria, alternatives or arguments could be added/eliminated, or that their current weights were wrong. Or, by tracing back the rules applied, we can detect that some attributes are missing or should be added for more accurate business rules.

Therefore the Enterprise Model and the Decision Space is continuously updated for keeping the pace with the constant changes that occur both inside the

enterprise and in the outside world.

Depending on the nature of change observed, business rules are changed by either choosing other existing rules, modifying existing rules or creating new rules if none of the existing ones meet the new context coordinates.

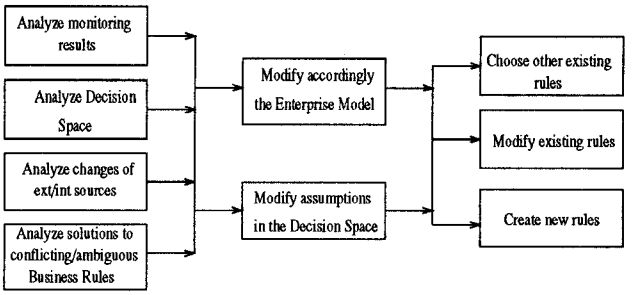

Figure 10: Business Rules Evolution

5 Example

Here we introduce the example used in the next section for the extraction of business rules algorithm. Our example is inspired from the London Ambulance Service (LAS) case study proposed as a common example at the 1996 International Workshop on Software Specification and Design [3].

Issue: Assess Resource Needs

Criteria / Alternatives	Quick Response Time	Importance	Effective Resources usage	Importance	Alternative Merit
Send Ambulance	0.351	0.6	0.376	0.4	0.18
Send Helicopter					
Send Major Incident Team					

Arg. Merit	Arg. Import.	Arg. names	Arguments Description
18.8%	-1.0	Slow Response	This alternative is too slow in lifethreatening cases.
81.2%	0.3	Acceptable Response	This alternative is acceptable in lifethreatening cases.
99.9%	1.0	Quick Response	This alternative is quick in non-lifethreatening cases.

Assumptions

Systolic Blood Pressure	Diastolic Blood Pressure	Pulse	Breathing	Temp.	Distance	Class
very high	very high	high	30	high	15	False
high	very high	normal	21	normal	65	False
very low	very low	low	8	normal	16	False
high	high	normal	21	high	65	True
high	very high	normal	22	high	5	True
...	...	...	...	...	...	...
high	normal	high	19	normal	18	True

Figure 11: Decision matrix and data base of assumptions for the Assess Resource Needs issue. The assumptions table contains examples referring to the *Send Ambulance is Quick in non-life-threatening cases* argument.

LAS is in charge of dispatching resources (ambulances, helicopters, etc.) to incident scenes. Some of the key objectives of the system are: to get the most appropriate resource(s) to the scene of an incident (e.g., a heart attack, a traffic accident, a terrorist attack, etc.) and to get it there as quickly as possible (i.e., according to standards that describe acceptable response times). In the format of the manual LAS system that existed before introducing the Computer Aided Dispatch system, a call taker writes down the details on a form and identifies the location of the caller when an incident is reported. All the incident forms go to a central collection point where a staff member reviews the details on each of them and decides what type of resources are needed for each incident and which resource allocator should deal with it. The resource allocator decides which resource should be mobilized and gives this information to a dispatcher who will pass the mobilization information to the appropriate resources (vehicles, crews, hospitals, etc.).

In this process we are interested in analyzing the situations where human judgment is involved, in particular, what business rules are applied and what is the motivation behind them. One such situation is the assessment of the type of resources needed to solve a particular incident. The information necessary to make this decision is represented in the decision matrix and the set of data illustrated in Figure 11. The content of the matrix was populated with information gathered from the decision makers. The numbers showed in the matrix are computed by the rule extraction algorithm (see the next section). In this step of the LAS process a staff member has to decide whether to send an ambulance, a helicopter or a major incident team based on the details mentioned in the incident form. In order to make this decision the staff member takes into account criteria such as quick response time since the call was received and an effective use of the resources available. The figure shows the pro (0.3, 1.0) and counter (-1.0) arguments taken into account for assessing the degree of satisfaction (0.954) of the *Quick Response Time* criterion by the *Send Ambulance* alternative. Also, we show a fragment of statistical data recording how the various assumption values support the argument "This alternative is quick in non-life-threatening cases." These assumptions are attributes that characterize the non-life-threatening cases (we have considered just a few of them for this example) and the distance of the incident location from the nearest hospital.

6 Extraction of Business Rules from Decision Structures and Examples

This section discusses how we support the automatic extraction of business rules. Our objective is to generate a set of business rules that preserve the information in the decision matrix and to reflect knowledge from statistical data about domain assumptions.

6.1 Decision Structure Knowledge

Next we will formalize the knowledge contained in the decision structures represented by a decision matrix. This knowledge is used in the rules extraction algorithm, and corresponds to the primitives described in the decision support submodel in Section 3. These primitives are naturally expressed in terms of variables

and constraints on their values. Thus we distinguish the following variables:

- The *issue* (*Iss*) that needs to be solved: *Assess Resource Needs* in our example.
- A number of *alternatives* (Alt_i) that are proposed as solutions to the issue *Iss*. For example, Alt_1 = *Send Ambulance*.
- A set of *criteria* ($Crit_j$) against which all the alternative solutions are evaluated in order to decide upon the best alternative. For example, $Crit_1$ = *Quick Response Time*.
- A number of pro and counter *arguments* that correspond to each pair (alternative, criterion) ($Arg_k(Alt_i, Crit_j)$). For example, $Arg_1(Alt_1, Crit_1)$ = *Slow Response*, related to the more intuitive description "This alternative is too slow in life-threatening cases".
- A set of *assumptions* that support each argument ($Attr_l(Arg_k)$). Assumptions represent groundable attributes of one or more objects in the Enterprise Model that are relevant to the issue under consideration. For example, $Attr_1(Arg_3(Alt_1, Crit_1))$ = *Systolic Blood Pressure*. Some attributes may be common for various arguments.

6.2 Format of the Business Rules Extracted from Decision Structures and Statistical Data

We are proposing three types of business rules that correspond to different levels of detail. They are the rules obtained at the criteria level, at the arguments level and at the assumptions level of a decision matrix. These different business rule types correspond to different levels of decision making in the hierarchy of an enterprise.

1. Criteria level:
The rules obtained at this level are the most general type of rules. They correspond to high level decision making and express enterprise objectives in very general terms. Therefore these objectives will need to be refined to the point where they can be translated into operational business rules. These objectives will serve as criteria for evaluating the alternatives of various solutions proposed for solving various problems.

For example, in the rule

WHEN *Assess Resource Needs*
IF *Quick Response Time* [0.6] ∨ *Effective Resource Usage* [0.4]
THEN *Send Ambulance*

that expresses what solution (alternative) to choose (*Send Ambulance*) for solving the problem *Assess Resource Needs* for an incident, the criteria established by the enterprise for solving this problem are *Quick Response Time* of the resources sent to an incident site and *Effective Resource Usage*. As we can see, there is no precise definition yet of what the non-functional requirements *Quick Response Time* and *Effective Resource Usage* really mean. The only thing we know at this stage is the importance of each criterion (0.6 and 0.4) in evaluating the alternatives.

The general format of this type of rule is:

WHEN *Iss*
IF $Crit_1[w_1] \vee Crit_2[w_2] \vee \cdots \vee Crit_n[w_n]$
THEN Alt_i

where w_i represent the weights or *Importance* (see Figure 11) of criterion $Crit_j$ in the process of alternative Alt_i evaluation. The w_i values are given by the decision makers.

Whenever this type of rule is applied we compute 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 $Merit_{Crit_j}$ of the alternative Alt_i in satisfying each criterion $Crit_j$:

$$Merit_{Alt_i} = \frac{\sum_{j=1}^{n_{crit}} w_{Crit_j} * Merit_{Crit_j}}{n_{crit}} \quad (1)$$

2. The arguments level:
The rules at this level express the heuristics used in deciding how well an alternative satisfies a criterion when several arguments are presented for, or against a solution (alternative). 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 $Merit_{Crit_j}$ of the alternative Alt_i against criterion $Crit_j$. These rules express the fact that the meaning of the enterprise objectives is not always obvious, and therefore requires negotiation among stakeholders.

For example, in the rule

WHEN *Send Ambulance*
IF *Slow Response* = True [-1.0] ∨ *Acceptable Response* = True [0.3] ∨ *Quick Response* = True [1.0]
THEN *Quick Response Time*

we express the fact that in the process of refining the meaning of the *Quick Response Time* objective there have been brought up three arguments with different weights in the context of sending an ambulance to an incident. These arguments correspond to different situations perceived by various stakeholders as being plausible: 1) sending an ambulance may be too slow in life-threatening cases where the location of the incident is far away from a hospital (and therefore, this is a counterargument for sending an ambulance in these cases); 2) sending an ambulance may be acceptable in life-threatening cases if they are close to a hospital; 3) sending an ambulance is a quick solution in non-life-threatening situations, regardless of the distance from the hospital (and therefore gives a stronger support to the alternative than argument 2)). All these arguments (and others corresponding to other criteria in the level 1 rule) will be taken into consideration when making the decision about whether sending an ambulance.

The general format of this type of rule is:

WHEN Alt_i
IF $Arg_1[w_1] \vee Arg_2[w_2] \vee \cdots \vee Arg_n[w_n]$
THEN $Crit_j$

w_i represent the weight of each argument in the evaluation process. It can take values on a scale [-1.0, 1.0] meaning: when $w_i = -1.0$, Arg_i is a counterargument, while when $w_i = 1.0$, Arg_i is a strong supporting argument.

The computation of $Merit_{Crit_j}$ is based on the weight of each argument (w_{Arg_i}) and the predicted accuracy of the truth value of that argument ($Merit_{Arg_i}$):

$$Merit_{Crit_j} = \frac{\sum_{i=1}^{n_{arg}} w_{Arg_i} * Merit_{Arg_i}}{n_{arg}} \quad (2)$$

3. The assumptions level:
This is the most detailed level of rules where the business objectives find their operational meanings. Even though they correspond to operational (low level) decision making, there might still be situations that require grounding of some terms inside the rules.

These rules express the operational conditions that need to be met in order for the alternative Alt_i to meet the criteria $Crit_j$ (an enterprise goal), e.g. they assess the truth value of the arguments Arg_i associated with the pair (Alt_i,$Crit_j$) based on various domain assumptions. The subconditions in the antecedent part of the rules are obtained either automatically by induction from statistical data, or, when this data is not available, by asking the decision maker. Even though these rules might look like the rules of an expert system, they are in fact business rules that achieve a goal.

For example, the rule

IF (*Diastolic Blood Pressure* = High $\wedge$ *Systolic Blood Pressure* = High) $\vee$ (*Distance from Hospital* < 28)
THEN *Quick Response* = True [99.9%]

expresses the operational conditions for the argument *Send ambulance is a quick solution in non-life-threatening situations* to be true. More than that, it shows the perceived accuracy (99.9%) of this assessment.

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

IF $(As_1 \wedge As_2 \wedge As_3) \vee (As_4 \wedge As_5)$
THEN Arg_i $[Merit_{Arg_i}]$

where $Attr_i$ represent the attributes whose values v_i need to be checked in order to assess the truth value of the argument Arg_i. The $Merit_{Arg_i}$ describes the certainty factor about the truth value of argument Arg_i.

6.3 Automatic Generation of Business Rules

The method for the automatic generation of business rules uses the knowledge structure provided by the decision support system through decision matrices. Also it uses statistical data recording how domain assumptions support various arguments (see Fig. 11).

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

Decision tree learning is a supervised machine learning technique that uses a collection of training examples and outputs a compact decision structure called a decision tree. 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 of level three using a technique of extraction of rules from decision trees (see [12, 15]).

The rules at levels two and one can be automatically generated from the decision structures represented in a decision matrix. The merit of each alternative in the decision matrix can be computed according to the formulae 1 and 2.

For obtaining the rules at level three we apply an algorithm for the induction of decision trees and rules from statistical data [12, 15]. The attributes used for classification are the domain assumptions that underlie an argument. The algorithm tolerates missing values for some of the attributes, therefore allowing for imprecise information about a case. The classes resulted after the application of the algorithm over a data base of assumptions values are the truth values for an argument. Each class has associated the certainty factor of that classification, which represents the likelihood that an argument value is true or false. From an induced decision tree we extract and optimize rules (both at the rule level and rule sets level) that will represent the level three type of rules.

The result of the antecedent phase is a set of rules (DSS rules) associated with every issue ($DSSR(Iss)$). Each such set of rules can be transformed into an operational business rule that follows the ECA format. An operational rule associated with an issue Iss is obtained from $DSSR(Iss)$ by applying the operation of consequent expansion at levels one and two. Consequent expansion means replacing 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 in an OR logical operation. The weights of the new conditions generated through consequent expansion depend on their initial weights, certainty factors of the rules expanded, and the number of rules expanded.

This way the enterprise objectives (stated in level one DSS rules) are refined to the point where they can be translated into operational business rules that achieve the enterprise goals.

7 Summary and Conclusions

A methodology has been presented that provides a fair degree of guidance for the activities of acquiring, deploying and evolving business rules of an enterprise. A requirements modeling framework has been introduced for representing the information needed to conduct the methodology. The meaning of requirements analysis in this context has been articulated. We have emphasized the importance of several aspects of the methodology and its associated framework.

First, we studied a project where business rules were recorded to understand the types of business rules and the factors that determined them. We were motivated by the desire to include business rules in the requirements modeling framework.

Second, we have emphasized the importance of an appropriate enterprise model, in terms of which the business rules can be expressed. The Process/Object

enterprise model suffices for the purposes of this paper. However, ultimately, a richer enterprise model will be needed in order to facilitate the expression of additional types of business rules. The Service-Oriented Systems model [5] will be extended to include the types of reasoning proposed in [17], in terms of goals, tasks, and resource dependencies. Also, a model of services is needed to express business rules about how services are provided between the participants in a service (customers and providers), the roles of the processes, and the positions of the organization that bear capabilities and responsibilities.

Third, we have highlighted the special nature of business rules as *decisions* whose consequences are speculative and subject to change. The sense in which business rules are "satisfied" is also defined flexibly to correspond to how they must be treated and tolerated in practice. Although we have focussed on business rules in this paper, the ideas in this paper are applicable, in principle, to requirements in general.

Fourth, we described the use of a decision support framework for capturing deliberations that can be used to continually evaluate and regenerate the business rules as the enterprise and its operational systems evolve. It should be clear that the information used in deliberations of this sort will be needed through cycles in which the business rules are re-examined and revised. This is in contrast to the usual requirements engineering scenario where the requirements documentation may not contain the right kinds of information that are needed later in the lifecycle or are difficult to keep up to date.

Fifth, the possibility for lifecycle automated assistance has been demonstrated in terms of the automatic extraction of business rules using the decision support framework. The extraction algorithm applies a technique that is well-known in the field of Machine Learning to a new field, Requirements Engineering, by using it in the business rules context.

The contribution of this paper is in the synthesis of the elements of the methodology. The methodology has not been used in practice, so there is no experience to report on its overall effectiveness. The metamodel has been implemented and some example models have been acquired (enterprise models, decision spaces, business rules). The business rules extraction algorithm has been implemented and some of the results produced have been shown in this paper.

Acknowledgments

We acknowledge Howard Reubenstein for his work on the taxonomy of business rules.

References

[1] J. Bubenko Jr. and B. Wangler. Objectives driven capture of business rules and of information systems requirements. In *Proceedings of the International Conference on Systems, Man and Cybernetics*, pages 670–677, 1993.

[2] L. Chung, B. Nixon, and E. Yu. Using non-functional requirements to systematically support change. In *Second IEEE International Symposium on Requirements Engineering*. 1995.

[3] M. Feblowitz, S. Greenspan, H. Reubenstein, and R. Walford. ACME/PRIME: Requirements acquisition for process-driven systems. In *International Workshop on Software Specifications and Design*, pages 36–45. IEEE Computer Society Press, 1996.

[4] S. Fickas and M. Feather. Requirements monitoring in dynamic environments. In *Proceedings of IEEE International Symposium on Requirements Engineering*, 1995.

[5] S. Greenspan and M. Feblowitz. Requirements engineering using the SOS paradigm. In *Proceedings of IEEE International Symposium on Requirements Engineering, San Diego*, 1993.

[6] H. Herbst. A meta-model for specifying business rules in system analysis. In *Proceedings of CAiSE'95*, pages 186–199, 1995.

[7] Intellicorp. *Livemodel User's Guide, Betaversion*, 1995.

[8] A. v. Lamsveerde, R. Darimont, and P. Massonet. Goal-directed elaboration of requirements for a meeting scheduler:problems and lessons learnt. In *Proceedings of RE'95*. IEEE Computer Society Press, 1995.

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

[10] J. Martin and J. Odell. *Object-Oriented Methods: A Foundation*, chapter 20. Prentice Hall, 1995.

[11] L. Paul. Hidden assets. *PC Week*, February 20 1995.

[12] J. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993.

[13] R. Roland. *The Business Rules Handbook*. Database Research Group, Inc., Boston, MA, 1994.

[14] D. Rosca, S. Greenspan, C. Wild, H. Reubenstein, K. Maly, and M. Feblowitz. Application of a decision support mechanism to the business rules lifecycle. In *Proceedings of the KBSE95 Conference*, pages 114–122, 1995.

[15] J. Rosca and D. Rosca. Knowledge acquissition facilities within an expert system toolkit. In *Proceedings of the Seventh International Symposium on Computer Science*, Jassy,Romania, 1989.

[16] A. Sandifer and B. V. Halle. *Business Rules: Capturing the most elusive information asset*. Auerbach Publications, 1993.

[17] E. Yu. *Modelling Strategic Relationships for Process Reenginering*. PhD thesis, University of Toronto, December 1994.