Agent-Oriented Enterprise Modeling Based on Business Rules*

Kuldar Taveter[1] and Gerd Wagner[2]

[1] VTT Information Technology (Technical Research Centre of Finland),
P.O.Box 1201, FIN-02044 VTT, Finland,
kuldar.taveter@vtt.fi
[2] Eindhoven University of Technology, Faculty of Technology Management,
P.O. Box 513, 5600 MB Eindhoven, The Netherlands,
G.Wagner@tm.tue.nl
WWW home page: http://tmitwww.tm.tue.nl/staff/gwagner

Abstract. Business rules are statements that express (certain parts of) a business policy, defining business terms and defining or constraining the operations of an enterprise, in a declarative manner. Since these rules define and constrain the interaction among business agents in the course of business processes, they have to refer to the components of their mental state, such as the knowledge/information and the commitments of an organization. We propose an agent-oriented approach to business rules and show how to represent and visualize business rules and business processes in Agent-Object-Relationship modeling.

1 Introduction

Agent-Orientation is emerging as a new paradigm in software and information systems engineering. It offers a range of high-level abstractions that facilitate the conceptual and technical integration of communication and interaction with established information system technology. Agent-Orientation is highly significant for business information systems since business processes are driven by and directed towards agents (or *actors*), and hence have to comply with the physical and social dynamics of interacting individuals and institutions.

While today's enterprise information system technology is largely based on the metaphors of *data management* and *data flow*, and is under pressure to adopt concepts and techniques from the highly successful object-oriented programming paradigm, Agent-Orientation emphasizes the fundamental role of actors/agents[1] and their mental state, and of communication and interaction, for analyzing and designing organizations and organizational information systems. This turns out to be crucial for a proper understanding of business rules. Since these rules define and constrain the interactions among business agents, they have to refer to the

* To appear in Proc. of 20th Int. Conf. on Conceptual Modeling (ER2001), November 2001, Yokohama, Japan, Springer-Verlag, LNCS, 2001.

[1] We use the terms 'actor' and 'agent' as synonyms.

components of their mental state, such as the knowledge/information and the commitments of an organization.

We attempt to show that our agent-oriented approach, that is based on the Agent-Object-Relationship (AOR) metamodel proposed in [Wag01a,Wag01b], allows to capture more of the dynamic and deontic semantics of enterprise modeling than object-oriented modeling approaches, such as the UML, do. Taking into account that the main motivation for object-oriented modeling stems from software engineering and not from enterprise modeling, or cognitive modeling, this should not be surprising.

The rest of the paper is organized as follows. In Section 2, we review the relevant literature on business rules, and present our own definitions of business rules and business processes. In Section 3, we review the Agent-Object-Relationship (AOR) metamodel which we use as the basis of our agent-oriented business rule modeling. Finally, in Section 4, we discuss the formalization and visualization of business rules on the basis of the AOR metamodel.

2 Business Rules and Business Processes

According to Martin and Odell [MO98], business rules allow user experts to specify policies in *small, stand-alone units* using explicit statements. The term *business rule* can be understood both at the level of a business domain and at the operational level of an information system. The more fundamental concept are business rules at the level of a business domain. In certain cases, they can be automated by implementing them in an information system, preferably in the form of an executable specification. It should be the goal of advanced information system technology to provide more support for business rules in the form of high-level machine-executable declarative specifications, similar to the SQL concepts of *assertions* and *triggers*.

2.1 Business Rules at the Business Level

At the business level, a business rule is defined as

- a statement about how the business is done, i.e., about guidelines and restrictions with respect to states and processes in an organization [Her97];
- a law or custom that guides the behaviour or actions of the actors connected to the organization [Ass88];
- a declaration of policy or condition that must be satisfied [OMG, 1992].

Business rules can be enforced on the business from the outside environment by regulations or laws, or they can be defined within the business to achieve the goals of the business. A business rule is based on a *business policy*. An example of a business policy in a car rental company is "only cars in legal, roadworthy condition can be rented to customers" [HH00]. Business rules are *declarative* statements: they describe *what* has to be done or *what* has to hold, but not *how*.

Our definition of business rules is based on [HH00], [Ass88], [KK92], and [BBS]: ***Business rules are statements that express*** (certain parts of) ***a business policy, such as defining business terms, defining deontic assignments (of powers, rights and duties), and defining or constraining the operations of an enterprise, in a declarative manner*** (not describing/prescribing every detail of their implementation).

According to [HH00] and [MDC99], business rules can be divided into 'structural assertions' (or 'term rules' and 'fact rules'), 'action rules', and 'derivation rules'.[2] Similarly, Bubenko et al [BBS] categorize business rules into 'constraint rules', 'event-action rules', and 'derivation rules', while Martin and Odell [MO98] group rules into two broad classes, 'constraint rules' and 'derivation rules' (remarkably, they subsume 'stimulus response rules' – which we call *reaction rules* – under 'constraint rules'). [Her97] distinguishes between 'integrity rules' (that are further divided into static and dynamic integrity constraints) and 'automation rules'.

In [HH00], a further class of business rules, *authorizations*, is proposed. Authorizations represent a particular type of *deontic assignments*. Synonyms for authorizations are *rights* and *permissions*. They define the privileges of an agent (type) with respect to certain (types of) actions. Complementary to rights, we also consider *duties*.

In summary, three basic types of business rules have been identified in the literature: *integrity constraints* (also called 'constraint rules' or 'integrity rules'), *derivation rules*, and *reaction rules* (also called 'stimulus response rules', 'action rules', 'event-action rules', or 'automation rules'). A fourth type, *deontic assignments*, has only been partially identified (in the proposal of considering 'authorizations' as business rules).

An ***integrity constraint*** is a an assertion that must be satisfied in all evolving states and state transition histories of an enterprise viewed as a discrete dynamic system. There are state constraints and process constraints. *State constraints* must hold at any point in time. An example of a state constraint is: "a customer of the car rental company EU-Rent must be at least 25 years old". *Process constraints* refer to the dynamic integrity of a system; they restrict the admissible transitions from one state of the system to another. A process constraint may, for example, declare that the admissible state changes of a `RentalOrder` object are defined by the following transition path: *reserved* → *allocated* → *effective* → *dropped-off*.

A ***derivation rule*** is a statement of knowledge that is derived from other knowledge by an inference or a mathematical calculation. Derivation rules capture terminological and heuristic domain knowledge that need not to be stored explicitly because it can be derived from existing or other derived information

[2] 'Structural assertions' introduce the definitions of business entities and describe the connections between them. Since they can be captured by a conceptual model of the problem domain, e.g. by an Entity-Relationship (ER) or a UML class model, we do not consider them as business *rules* but rather as forming the business *vocabulary* (or *ontology*).

on demand. An example of a derivation rules is: "the rental rate of a rental is inferred from the rental rate of the group of the car assigned to the rental".

Reaction rules are concerned with the invocation of actions in response to events. They state the conditions under which actions must be taken; this includes triggering event conditions, pre-conditions, and post-conditions (effects). An example of a reaction rule from the domain of car rental is: "when receiving from a customer the request to reserve a car of some specified car group, the branch checks with the headquarter to make sure that the customer is not blacklisted".

Deontic assignments of powers, rights and duties to (types of) internal agents define the deontic structure of an organization, guiding and constraining the actions of internal agents. An example of a deontic assignment statement is: "only the branch manager has the right to grant special discounts to customers".

The triggering event conditions in the definitions of reaction rules in [HH00], [Her97], [BBS], and [MDC99] are either explicitly or implicitly bound to update events in databases. Depending on some condition on the database state, they may lead to an update action and to system-specific procedure calls. In contrast to this, we choose the more general concept of a reaction rule as proposed in [Wag98]. Reaction rules define the behaviour of an agent in response to environment events (perceived by the agent), and to communication events (created by communication acts of other agents).

2.2 Business Rules at the Level of an Information System

In certain cases, business rules expressed at the business level can be automated by mapping them to executable code at the information system level as shown in Table 1.

Concept	*Implementation*
Constraints	if-then statements in programming languages; `DOMAIN`, `CHECK` and `CONSTRAINT` clauses in SQL table definitions; `CREATE ASSERTION` statements in SQL database schema definitions
Derivation Rules	deductive database (or Prolog) rules; SQL `CREATE VIEW` statements
Reaction Rules	if-then statements in programming languages; `CREATE TRIGGER` statements in SQL; production rules in 'expert systems';

Table 1. Mapping of business rules from the business level to the information system level using currently available technology.

This mapping is, however, not one-to-one, since programming languages and database management systems offer only limited support for it. While general purpose programming languages do not support any of the three types of expressions (with the exception of the object-oriented language Eiffel that supports integrity constraints in the form of 'invariants' for object classes), SQL has some

built-in support for constraints, derivation rules (views), and limited forms of reaction rules (triggers).

2.3 Business Processes

Business rules define and control *business processes*. A widely accepted definition of a business process is [Dav92]: "A business process can be defined as a collection of activities that takes one or more kinds of input, and creates an output that is of value to the customer". In [HC93] this definition is paraphrased by stating: "A [business] process is simply a structured set of activities designed to produce a specified output for a particular customer or market". A business process describes from start to finish the sequence of events required to produce the product or service [YWT+96]. A business process is assumed to consume input in terms of information and/or material and produce output of information and/or material [BBS]. Business processes typically involve several different functional organization units. Often business processes also cross organizational boundaries.

We prefer to adopt a more general perspective and consider a business process as a special kind of a *social interaction process*. Unlike physical or chemical processes, social interaction processes are based on communication acts that may create commitments and are governed by norms. We distinguish between an interaction process type and a concrete interaction process (instance), while in the literature the term 'business process' is ambiguously used both at the type and the instance level.

We thus refine and extend the definitions of [YWT+96], [HC93], and [Dav92]: ***A business process is a social interaction process for the purpose of doing business.*** According to [Wag01b], a ***social interaction process*** is a temporally ordered, coherent set of events and actions, involving one or more communication acts, perceived and performed by agents, and following a set of rules, or protocol, that is governed by norms, and that specifies the type of the interaction process. Notice that we did not choose *activities* as the basic elements of a process. While an *action* happens at a time point (i.e., it is immediate), an *activity* is being performed during a time interval (i.e., it has duration), and consists of a set of actions.

We propose to model both business rules and business processes in the framework of the *Agent-Object-Relationship* metamodel reviewed in Section 3.

3 Principles of Agent-Object-Relationship Modeling

Agent-Object-Relationship (AOR) diagrams were proposed in [Wag01a,Wag01b] as an agent-oriented extension of Entity-Relationship diagrams, or UML-style class diagrams. In order to capture more semantics of the dynamic and deontic aspects of organizations and organizational information systems, such as the events and actions related to the ongoing business processes of an enterprise, it is proposed to make an ontological distinction between active and passive entities, that is, between *agents* and *ordinary objects*. AOR modeling suggests

that the semantics of business transactions can be more adequately captured if the specific ***business agents*** associated with the involved events and actions are explicitly represented in organizational information systems in addition to passive ***business objects***.

In AOR modeling, an entity is either an *agent*, an *event*, an *action*, a *claim*, a *commitment*, or an *ordinary object*. An organization is viewed as a complex ***institutional agent*** defining the rights and duties of its internal agents that act on behalf of it, and being involved in a number of interactions with external agents. ***Internal agents*** may be humans, artificial agents (such as software agents, agentified information systems, robots or agentified embedded systems), or institutional agents (such as organizational units).

As usual, entity types are visually represented by rectangles while relationship types are represented by connection lines (possibly with crows feet endings in order to indicate multiplicity). While an *object type* is visualized as an ordinary rectangle, an *agent type* is graphically rendered as a rectangle with rounded corners. An internal agent type is visualized by such a rectangle with a dashed line drawn within the institutional agent rectangle it belongs to (like `Branch` in Fig. 1). An instance of an agent type is distinguished from an agent type by underlining its name (like the `EU-Rent` in Fig. 1).

3.1 Actions and Events

In a business domain, there are various types of actions performed by agents, and there are various types of state changes, including the progression of time, that occur in the environment of the agents. For an external observer, both actions and environmental state changes constitute events. In the internal perspective of an agent that acts in the business domain, only the actions of other agents count as events.

Actions create events, but not all events are created by actions. Those events that are created by actions, such as delivering a product to a customer, are called ***action events***. Examples of business events that are not created by actions are the fall of a particular stock value below a certain threshold, the sinking of a ship in a storm, or a timeout in an auction.

We make a distinction between *communicative* and *non-communicative* actions and events. Many typical business events, such as receiving a purchase order or a sales quotation, are communication events. Business communication may be viewed as asynchronous point-to-point message passing. The expressions *receiving a message* and *sending a message* may be considered to be synonyms of *perceiving a communication event* and *performing a communication act*.

As opposed to the low-level (and rather technical) concept of messages in object-oriented programming, AOR modeling assumes the high-level semantics of speech-act-based *Agent Communication Language (ACL)* messages (see [KQM,FIP]).

3.2 Commitments and Claims

Commitments are fundamental components of business interaction processes. This is acknowledged by the ebXML standardization initiative in the statement "The business semantics of each commercial transaction are defined in terms of the Business Objects affected, and the *commitment(s)* formed or agreed."[3]

Representing and processing commitments and claims in information systems explicitly helps to achieve coherent behavior in interaction processes. In [Sin99], the social dimension of coherent behavior is emphasized, and commitments are treated as ternary relationships between two agents and a 'context group' they both belong to. For simplicity, we treat commitments as binary relationships between two agents.

Commitments to perform certain actions, or to see to it that certain conditions hold, typically arise from certain communication acts. For instance, sending a sales quotation to a customer commits the vendor to reserve adequate stocks of the quoted item for some time. Likewise, acknowledging a sales order implies the creation of a commitment to deliver the ordered items on or before the specified delivery date.

Some of these modeling concepts are *indexical*, that is, they depend on the perspective chosen: in the perspective of a particular agent, ***actions*** of other agents are viewed as ***events***, and ***commitments*** of other agents are viewed as ***claims*** against them.

In the internal perspective of an agent, a commitment refers to a specific action to be performed in due time, while a claim refers to a specific event that is created by an action of another agent, and has to occur in due time.

3.3 External AOR Models

In an external AOR model, we adopt the view of an external observer who is observing the (prototypical) agents and their interactions in the problem domain under consideration. Typically, an external AOR model will have a *focus*, that is an agent, or a group of agents, for which we would like to develop a state and behavior model. We do not consider internal AOR models (taking the internal/subjective perspective of a particular agent/system to be modeled) in this paper.

An ***Agent Diagram*** depicts the focus agent (or agents) and the agent types it is (or they are) interacting with. If another agent (type) is to be represented by a focus agent (type) with 'proprietary' attributes (that have only meaning for the representer), such as when `Customer` is to be represented by `Headquarter` with the proprietary Boolean attribute `isBlacklisted`, then a corresponding agent rectangle with a dot-dashed line is drawn as a representation of the 'real' agent (type) within the focus agent (type), as in Fig. 1. Such an explicit representational duplication of an entity type is only necessary if proprietary attributes are to be included in the agent diagram. Otherwise, it is tacitly assumed that the

[3] From the *ebXML Technical Architecture Specification* v0.9.

focus agent has a representation of all agents it deals with in terms of 'standard' attributes.

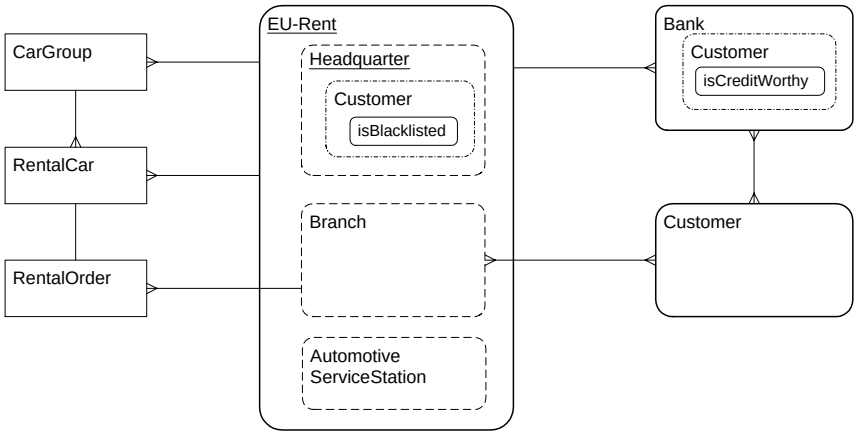

Fig. 1. An *Agent Diagram*: The car rental company `EU-Rent` is the focus agent. It consists of an internal agent `Headquarter` and (instances of) the internal agent types `Branch` and `AutomotiveServiceStation`. The headquarter classifies customers by means of the proprietary attribute `isBlacklisted`. Similarly, banks classify customers by means of the proprietary attribute `isCreditWorthy`. In both cases, a rectangle with a dot-dashed line is used to graphically render the internal representation entity type `Customer`.

In the view of an external observer, actions are also events, and commitments are also claims, exactly like two sides of the same coin. Therefore, an external AOR model contains, besides the agent and object types of interest, the action event types and commitment/claim types that are needed to describe the interactions between the focus agent(s) and the other types of agents. They are visualized in an ***Interaction Frame Diagram***.

In an external AOR model, a *commitment* of agent a_1 towards agent a_2 to perform an action of a certain type (such as a commitment to return a car) can also be viewed as a *claim* of a_2 against a_1 that an action of that kind will be performed. Commitments/claims are conceptually coupled with the type of action event they refer to (such as *returnCar* action events). This is graphically rendered by an arrow rectangle with a dotted line on top of the action event rectangle it refers to, as depicted in Fig. 2.

Action event types, and commitment/claim types are graphically rendered like in Fig. 2 which depicts the *interaction frames* between `Customer` and `Branch`, and between `Branch` and `Headquarter`. Notice that not for all action event types there is a corresponding commitment/claim type. For instance, there are no commitments of (or claims against) customers to pick up a car, whereas there are commitments and claims to return a car. An interaction frame between two agent

types consists of those action event types and commitment/claim types that form the basis of the interaction processes in which these two agent types are involved. Unlike a UML sequence diagram, it does not model any sequential process but provides a static picture of the possible interactions including commitment/claim types.

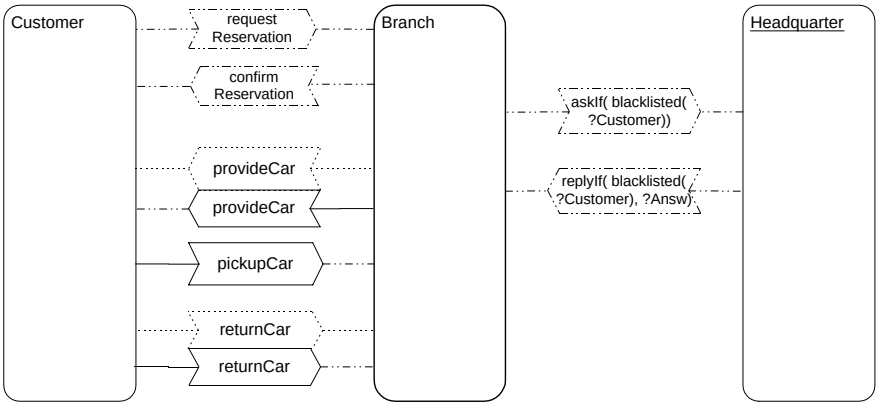

Fig. 2. An *Interaction Frame Diagram*: The interaction frame between the agent types `Customer` and `Branch` consists of the communicative action event (or ACL message) types `requestReservation` and `confirmReservation`, the action event type `pickupCar`, and the commitment/claim types `provideCar` and `returnCar` coupled with the corresponding action event types. The interaction frame between the agent type `Branch` and the agent `Headquarter` consists of the ACL message types `askIf( blacklisted(?Customer))` and `replyIf( blacklisted(?Customer), ?Answer))`.

An external AOR model should not include any software artifacts. It should rather represent a conceptual analysis view of the problem domain, similarly to the function of a UML *use case* model.

4 Business Rules as Reaction Rules

We propose to formalize business rules as integrity constraints, as derivation rules, as reaction rules, or as deontic assignments in the semantic framework of *knowledge-perception-memory-commitment (KPMC)* agents. The concept of KPMC agents is an extension of the *knowledge- and perception-based (KP)* agent model proposed in [Wag96,Wag98]. We can only sketch this logical framework here.

A KPMC agent consists of five components: a knowledge base KB, an event queue EQ (representing the perception state), a memory base MB (recording past events and actions), a commitment/claim base CB, and a set of reaction rules RR (encoding the behavior of the agent). The schema of a KPMC agent is

composed by a knowledge system (in the sense of [Wag98]), an agent communication language (ACL), an action language, and an environment event language. Integrity constraints and derivation rules are expressible on the basis of a knowledge system (and the query and input language defined by it). For expressing reaction rules one needs, in addition to the query and input language of a knowledge system, languages for expressing events and actions.

In this paper, for space limitations, we restrict our considerations to *reaction rules* that are visualized in ***Interaction Pattern Diagrams***. In [TW01], we discuss *deontic assignments*.

Business rules that define the interactive behavior of business agents are best formalized as reaction rules. Business interactions are influenced by commitments and claims. So, reaction rules are those business rules where an agent-oriented approach is most promising. At the same time, they seem to be the most important type of business rules.

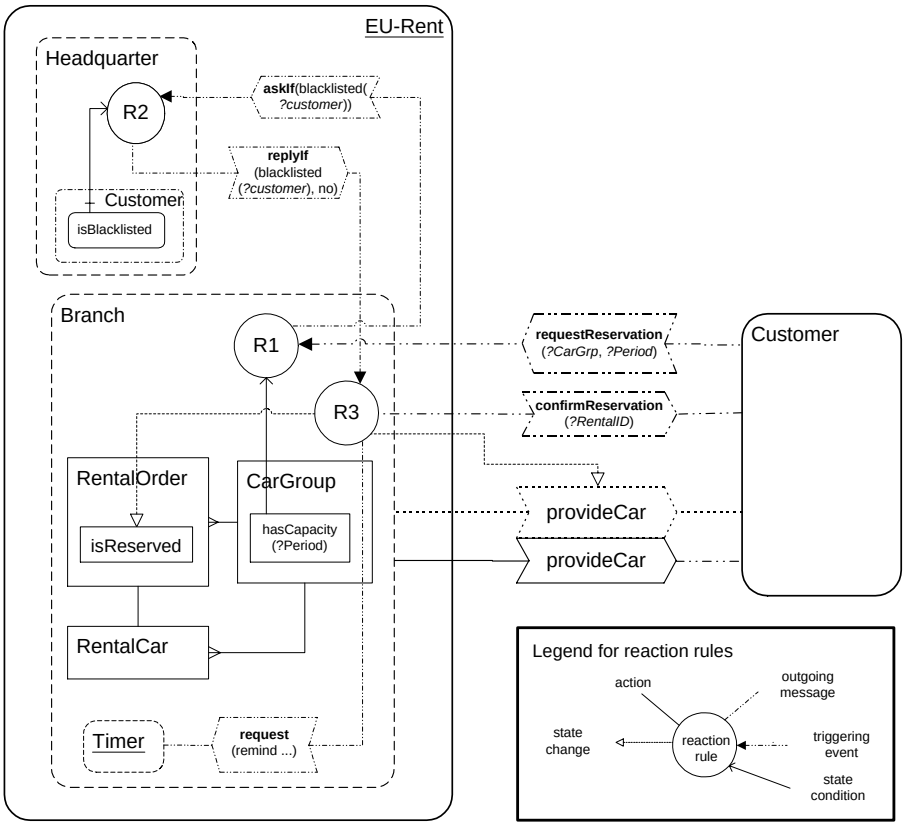

Fig. 3. An *Interaction Pattern Diagram*: Visualizing a process fragment defined by the reaction rules R1, R2 and R3.

In Fig. 3, the business process type of rental reservation is modeled on the basis of three reaction rules, R1, R2 and R3. Variables in the parameter list of a message type or predicate are prefixed with a question mark.

R1: Upon receiving from a `Customer` the request to reserve a car of some `?CarGroup` for a certain rental `?Period`, if that car group has sufficient capacity during the period requested – determined by evaluating the intensional predicate `hasCapacity(?Period)` of the corresponding instance of `CarGroup` – the `Branch` sends a query to the `Headquarter` to make sure that the customer is not blacklisted.

R2: Upon receiving from a `Branch` a query if some customer is blacklisted, the headquarter checks if the concerned customer is blacklisted, and if he is not, replies with 'no'.

R3: Upon receiving from the `Headquarter` a reply telling that the `Customer` is not blacklisted, the `Branch` creates the corresponding rental reservation (i.e. an instance of `RentalOrder` with the status `isReserved`), commits towards the customer to provide a car, sends a request to the `Timer` software agent to remind about the allocation time of a car for the given rental order (a car is allocated for the rental reservation 12 hours before the pickup-time), and sends a confirmation to the customer.

A reaction rule is visualized as a circle with incoming and outgoing arrows. Each reaction rule has exactly one incoming arrow that is solid: it represents the triggering event condition which is also responsible for instantiating the reaction rule. In addition, there may be ordinary incoming arrows representing state conditions (referring to corresponding instances of other entity types). There are two kinds of outgoing arrows. An outgoing arrow of the form $\multimap$ denotes a mental effect referring to a change of beliefs and/or commitments. An outgoing connector to an action type denotes the performance of an action of that type.

Reaction rules may also be represented in textual form. For instance, R1 could be expressed as

```
ON RECEIVE requestReservation(?CarGrp, ?Period) FROM ?Customer
IF ?CarGrp.hasCapacity(?Period)
THEN
    SEND askIf( blacklisted(?Customer)) TO headquarter
```

and R3 could be expressed as

```
ON RECEIVE replyIf( blacklisted(?Customer), no) FROM headquarter
THEN
    COMPUTE ?RentalNo = getNewRentalNo();
    CREATE BELIEF  RentalOrder(?RentalNo, ?Customer, isReserved, ...)
    CREATE COMMITMENT TOWARDS ?Customer TO provideCar(...) BY ...
    SEND request( remind(?RentalNo, ...)) TO timer
    SEND confirmReservation(?RentalNo,...) TO ?Customer
```

5 Related Work

We restrict our discussion of related work to those approaches in enterprise modeling where business rules play an essential role.

In object-oriented approaches, rules are frequently implemented within the methods of a business object class. In many cases, however, this binding of a business rule to a specific object class is not adequate. Typically, a rule refers to more than one type of business object. Therefore, business rules should be defined on top of the business object definitions (classes) in a separate module.

In [EP99], Eriksson and Penker propose an approach to business modeling with UML based on four primary concepts: resources, processes, goals, and rules. In this proposal, there is no specific treatment of agents. They are subsumed, together with "material, information, and products" under the concept of *resources*. This unfortunate subsumption of human agents under the traditional 'resource' metaphor prevents a proper treatment of many agent-related concepts such as commitments, deontic assignments, and communication/interaction.

Ross [Ros97] has proposed one of the most comprehensive methodologies for modeling business rules. The Ross Notation is, however, largely a database-oriented methodology, and does therefore not allow to model events and actions. Neither does it support to model business processes. E.g., [Hur98] remarks that the primary deficiency of the Ross Notation is its inability to model process aspects, due to its fundamental restriction of only considering persistent data as a basis for business rules.

The Enterprise Knowledge Development (EKD) approach described in [BBS] also addresses the modeling of business rules, business processes, and actors. The EKD approach does not, however, bring these notions straightforwardly to the operational level like we do in our approach. Also, visualization of business rules and processes in EKD is quite simplistic (by boxes).

Recently, in [OvDPB00], an agent-oriented extension of UML, called *AUML*, mainly concerning the expressivity of sequence diagrams and activity diagrams, has been proposed. However, AUML does not distinguish between agents and objects. In fact, UML class diagrams are not modified at all in AUML. Neither does it provide any support for (business) rules.

6 Conclusion

Business rules have traditionally been modeled and implemented in the narrow context of (active) databases. We have adopted a broader view, and a more cognitive stance, by proposing to model and implement business rules as the "rules of behaviour" of business agents. We have also shown how to visualize business rules in Agent-Object-Relationship models. We did not say anything about a suitable modeling process/method associated with the AOR modeling language. This is a topic for further research.

In addition to the case study of a car rental company, the methodology described in this paper has been used in designing an information system in support

of inter-enterprise business processes of electronic advertising in newspapers, and in designing the scheduling information system for a ceramic factory, see [Tav01].

We are aware that, by introducing new concepts for enterprise modeling, we have also created new problems and research challenges. Some of the new questions that arise from our approach are:

- How can commitments/claims be used in real systems? What is their operational semantics?
- How can we relate our formalization of business rules with goals and goal-oriented behavior based on planning and plan execution?
- How can we handle exceptions to standard processes (for instance, when a customer does not appear to pick up a car as agreed, or when the automotive service station fails to return a car on time)? Possibly as violations of commitments?

These and many more questions will guide our future work.

Acknowledgements We are grateful to the anonymous referees for their valuable hints and suggestions.

References

[Ass88] F. Van Assche. Information systems development: a rule-based approach. *Knowledge-Based Systems*, 1(4):227–234, 1988.

[BBS] J. A. Bubenko, D. Brash, and J. Stirna. EKD user guide. Technical report, Kista, Dept. of Computer and Systems Science, Royal Institute of Technology (KTH) and Stockholm University, Stockholm, Sweden. `http://www.dsv.su.se/~js/ekd_user_guide.html`.

[Dav92] T. H. Davenport. *Process Innovation: Reengineering Work through Information Technology*. Harvard Business School Press, 1992.

[EP99] H.E. Eriksson and M. Penker. *Business Modeling with UML: Business Patterns at Work*. John Wiley & Sons, 1999.

[FIP] Foundation for intelligent physical agents (FIPA). http://www.fipa.org.

[GLC99] B.N. Grosof, Y. Labrou, and Hoi Y. Chan. A declarative approach to business rules in contracts: Courteous logic programs in XML. In *Proc. 1st ACM Conference on Electronic Commerce (EC99)*, Denver, Colorado, USA, November 1999.

[HC93] M. Hammer and J. Champy. *Reengineering the Corporation*. Harper Collins, New York, 1993.

[Her97] H. Herbst. *Business Rule-Oriented Conceptual Modeling*. Contributions to Management Science. Springer-Verlag, 1997.

[HH00] D. Hay and K. A. Healy. Defining business rules - what are they really? Technical Report 1.3, The Business Rules Group, July 2000. `http://businessrulesgroup.org/first_paper/br01c0.htm`.

[Hur98] Russ Hurlbut. *Managing Domain Architecture Evolution Through Adaptive Use Case and Business Rule Models*. PhD thesis, Illinois Institute of Technology, 1998.

[KK92] A. Kieser and H. Kubicek. *Organisation*. De Gruyter, Berling/New York, 3rd edition edition, 1992.

[KQM] Knowledge query and manipulation language (KQML). http://www.cs.umbc.edu/kqml/.

[MDC99] Meta data coalition open information model, business engineering model, business rules. review draft, Kista, Dept. of Computer and Systems Science, Royal Institute of Technology (KTH) and Stockholm University, July 1999. http://www.mdcinfo.com/OIM/models/BRM.html.

[MO98] James Martin and James Odell. *Object-Oriented Methods: A Foundation (UML Edition)*. Prentice-Hall, 1998.

[OvDPB00] J. Odell, H. van Dyke Parunak, and B. Bauer. Extending UML for agents. In G. Wagner, Y. Lesperance, and E. Yu, editors, *Proc. of the 2nd Int. Workshop on Agent-Oriented Information Systems*, Berlin, 2000. iCue Publishing.

[Ros97] R. G. Ross. *The Business Rule Book: Classifying, Defining and Modeling Rules*. Database Research Group, Inc., Boston (MA), 2nd edition edition, 1997.

[Sin99] M.P. Singh. An ontology for commitments in multiagent systems. *Artificial Intelligence and Law*, 7:97–113, 1999.

[Tav01] Kuldar Taveter. *Agent-Oriented Business Modelling and Simulation*. PhD thesis, Tallinn Technical University, 2001.

[TW01] K. Taveter and G. Wagner. Agent-oriented business rules: Deontic assignments. In *Proc. of Int. Workshop on Open Enterprise Solutions: Systems, Experiences, and Organizations (OES-SEO2001)*, Rome, Italy, September 2001.

[Wag96] G. Wagner. A logical and operational model of scalable knowledge- and perception-based agents. In W. Van de Velde and J.W. Perram, editors, *Agents Breaking Away*, volume 1038 of *Lecture Notes in Artificial Intelligence*, pages 26–41. Springer-Verlag, 1996.

[Wag98] G. Wagner. *Foundations of Knowledge Systems – with Applications to Databases and Agents*, volume 13 of *Advances in Database Systems*. Kluwer Academic Publishers, 1998. See http://www.inf.fu-berlin.de/∼wagnerg/ks.html.

[Wag01a] G. Wagner. Agent-oriented analysis and design of organizational information systems. In J. Barzdins and A. Caplinskas, editors, *Databases and Information Systems. Fourth International Baltic Workshop, Vilnius, Lithuania, May 2000*, Vilnius, Lithuania, 2001. Kluwer Academic Publishers.

[Wag01b] Gerd Wagner. The Agent-Object-Relationship meta-model: Towards a unified conceptual view of state and dynamics. Technical report, Eindhoven Univ. of Technology, Fac. of Technology Management, `http://tmitwww.tm.tue.nl/staff/gwagner/AOR.pdf`, May 2001. Submitted.

[YWT+96] E. Yourdon, K. Whitehead, J. Thomann, K. Oppel, and P. Nevermann. *Mainstream Objects: An Analysis and Design Approach for Business*. Yourdon Press, 1996.