

Specifying Process-Aware Access Control Rules in SBVR

Stijn Goedertier¹, Christophe Mues², and Jan Vanthienen¹

¹ Department of Decision Sciences and Information Management,
Katholieke Universiteit Leuven, Belgium

{stijn.g Goedertier, jan.vanthienen}@econ.kuleuven.be

² School of Management, University of Southampton, United Kingdom
c.mues@soton.ac.uk

Abstract. Access control is an important aspect of regulatory compliance. Therefore, access control specifications must be process-aware in that they can refer to an underlying business process context, but do not specify when and how they must be enforced. Such access control specifications are often expressed in terms of general rules and exceptions, akin to defeasible logic. In this paper we demonstrate how a role-based, process-aware access control policy can be specified in the SBVR. In particular, we define an SBVR vocabulary that allows for a process-aware specification of defeasible access control rules. Because SBVR does not support defeasible rules, we show how a set of defeasible access control rules can be transformed into ordinary SBVR access control rules using decision tables as a transformation mechanism.

Keywords: access control, defeasible logic, RBAC, SBVR, BPM.

1 Introduction

Access control is the ability to permit or deny access to physical or informational resources. It is an organization's first line of defence against unlawful or unwanted acts. Recently, access control has become a key aspect of regulatory compliance. The Sarbanes-Oxley Act [1], for instance, has led to far-reaching changes in access control policies of organizations world-wide. Effective access control is managed centrally and is process aware. A centralized specification and management of access control policy is more likely to prevent flaws in the access control policy and allows to incorporate user life cycle management. In addition, access control can only safeguard the integrity of an organization's business processes when it is aware of these underlying business processes.

The design and maintenance of an enterprise-wide, process-aware access control policy is however non-trivial. Consider, for instance, a credit approval process. A customer applies for credit and after a credit review, the bank can either make a credit proposal or reject the credit application. Credit approval requires the collaboration between the sales and the risk department. Suppose, for instance, that the following access control policy is formulated:

stakeholders: the customer, the bank, regulators

threat: the bank accepts credit applications with a high probability of default.

threat: credit reviews take up too much time and the customer defects to another bank.

concern: In general, each credit application must be reviewed by the bank’s risk department.

concern: For reasons of efficiency, credit applications of less or equal than 2000 euros may be reviewed by the sales department.

concern: The employee who reviews a credit application can neither be the beneficiary nor the applicant of the credit application.

concern: The same employee should not both review a credit application and make a credit proposal for credit applications larger than 2000 euros.

To date many access control specifications are either process agnostic or process driven. **Process-driven** access control specifications, hinder both design and runtime flexibility, because they are embedded within procedures, applications or process models. For instance, the left-hand side of Fig. 1 embeds the access control policy in a BPMN decision gateway [2]. Such process-driven specifications, hinders traceability and raises a myriad of problems when the policy is duplicated in multiple implementation forms. Duplication and lack of traceability of access control policies undermine the ability to have a consistent, flexible and guaranteed enterprise-wide access control policy. The opposite situation also entails problems. **Process-agnostic** access control specifications have only limited expressiveness, because they cannot relate to the state of business processes to grant or deny access rights. For instance, the above-defined policy prevents the same employee from both reviewing and making a proposal. This cannot be expressed without an awareness for an underlying process model. Expressive and flexible access control specifications are **process-aware** in that they can refer to an underlying business process context, but do not specify when and how they must be enforced. Such a specification is represented in the right-hand side of Fig. 1.

Process-aware access control policies should be on the one hand comprehensible so that they can be understood by business people and on the other hand

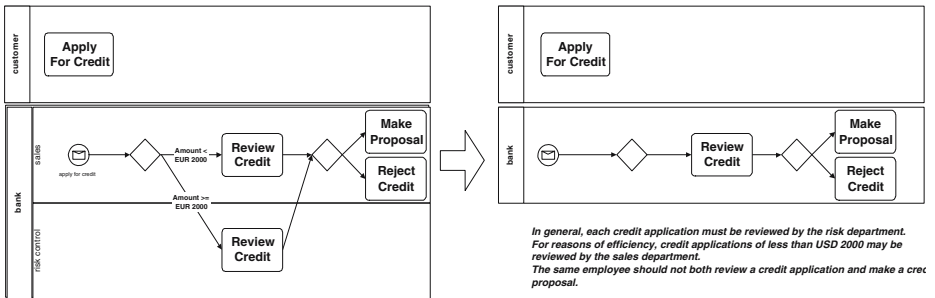


Fig. 1. Process-driven versus process-aware access control specifications

formal so that they can be enforced by information systems. The Semantics of Business Vocabulary and Business Rules (SBVR) [3,4] is a language for business modeling that has such property. In this paper we give a brief introduction to the SBVR and demonstrate how a role-based, process-aware access control policy can be specified. To this end we define a new SBVR vocabulary for verbalizing process-aware, access control rules. The vocabulary is part of a general initiative to declaratively model the business concerns that govern business processes in terms of business rules [5]. Access control specifications adhering to the role-based access control (RBAC) model are expressed in terms of general rules and exceptions and are related to defeasible logic. Because SBVR does not support defeasible rules, we show how a set of defeasible access control rules can be transformed into ordinary SBVR access control rules using decision tables as a transformation mechanism.

2 An Introduction to SBVR

The Semantics of Business Vocabulary and Business Rules (SBVR) is a new standard for business modeling that currently is under finalization within the Object Management Group (OMG). The standard provides a number of conceptual vocabularies for modeling a business domain in the form of a vocabulary and a set of rules [3,4]. In SBVR, meaning is kept separate from expression. As a consequence, the same meaning can be expressed in different ways. In real-life, meaning is more often expressed in textual form than in diagrams as statements provide more flexibility in defining vocabulary and expressing rules. For these reasons, the SBVR specification defines a structured, English vocabulary for describing vocabularies and verbalizing rules, called SBVR Structured English [4]. One of the techniques used by SBVR structured English are font styles to designate statements with formal meaning. In particular,

- the term font (normally green) is used to designate a noun concept.
- the name font (normally green) designates an individual concept.
- the *verb* font (normally blue) is used for designation for a verb concept.
- the keyword font (normally red) is used for linguistic particles that are used to construct statements.

The definitions and examples in the remainder of the text use these SBVR Structured English font styles.

In SBVR a vocabulary and a set of rules make up a so called ‘conceptual schema’. A conceptual schema with an additional set of facts that adheres to the schema is called a ‘conceptual model’. Figure 2 depicts the relationship of a conceptual schema and a conceptual model to some of the core building blocks in SBVR. These core building blocks are part of the SBVR ‘Meaning and Representation Vocabulary’. This vocabulary contains among others the following definitions [4]:

A conceptual schema is a combination of concepts and facts (with semantic formulations that define them) of what is possible, necessary, permissible, and obligatory in each possible world.

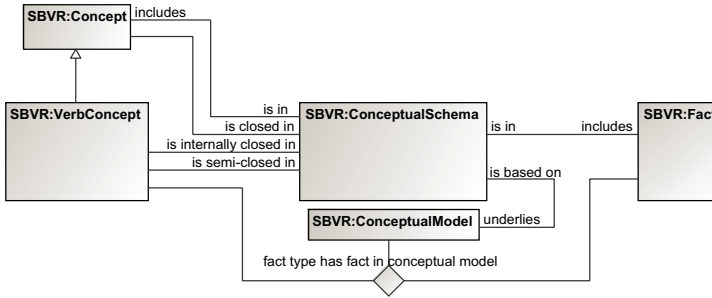


Fig. 2. A MOF/UML representation of SBVR conceptual schema and model [4]

A conceptual model or fact model *is* a combination of a conceptual schema and, for one possible world, a set of facts (defined by semantic formulations using only the concepts of the conceptual schema).

The facts in a conceptual model may cover any period of time. Changing the facts in a conceptual model creates a new and different conceptual model. In this way the SBVR gives conceptual models a monotonic semantics.

The SBVR provides a vocabulary called the ‘Logical Formulation of Semantics Vocabulary’ to describe the structure and the meaning of vocabulary and business rules in terms of formalized statements about the meaning. Such formalized statements are called ‘semantic formulations’ [6]. In addition to these fundamental vocabularies, the SBVR provides a discussion of its semantics in terms of existing, well-established formal logics such as First-Order logic, Deontic Logic and Higher-Order logic. This combination of linguistics and formal logic provides the fundamentals for developing a natural language parser that allows to express the meaning of rules that have a textual notation [7,8].

3 An SBVR Vocabulary for Process Modeling

The SBVR is a suitable base languages for defining process-aware access control rules, but to date there exists no SBVR vocabulary with process related concepts such as agents, activities and events. Consequently, it is not possible to declaratively refer to the state of a business process. In [5] we define an SBVR vocabulary for expressing process-related concepts, called the EM-BrA²CE Vocabulary. EM-BrA²CE stands for ‘Enterprise Modeling using Business Rules, Agents, Activities, Concepts and Events’. The vocabulary thinks of a business process instance as a *trajectory* in a *state space* that consists of the possible sub-activities, events and business concepts. Each activity in a process instance can undergo a number of distinct state transitions. The occurrence of a state transition is logged as an activity event. Business rules determine whether or not a particular state transition can occur. Consider, for instance the following state transitions:

- *create(AId, AT, BId, PId, CoordinatorId)*: requests the creation of a new activity *AId* of type *AT* with business identifiers *BId*, parent activity *PId* by an agent *CoordinatorId*. Activity event type: *created*.
- *assign(AId, AgentId, CoordinatorId)*: requests the assignment or revocation of the assignment of activity *AId* to an agent *AgentId* by an agent *CoordinatorId*. Activity event type: *assigned*.
- *updateFact(AId, C₁, C₂, WorkerId)*: requests the update of a business fact *C₁* by *C₂* within the context of activity *AId* by an agent *WorkerId*. Activity event type: *factUpdated*.
- *complete(AId, WorkerId)*: requests the completion of activity *AId* by an agent *WorkerId*. Activity event type: *completed*.

In [5] a total of twelve generic state transitions have been identified and a generic execution model has been defined in terms of Colored Petri Nets. Figure 3 illustrates a number of state transitions that occur to a given ‘review credit’ activity ‘a1’. Notice that each state transition results in a new set of concepts and ground facts, and thus a new state, that are partially represented in the columns of the figure. As each new activity state is considered to be a new SBVR:conceptual model, deductive reasoning can use a monotonic reasoning paradigm [4]. The current state of an activity determines which state transitions can occur. For the purpose of access control, the *assign(AId, AgentId, CoordinatorId)* is the activity state transition of interest.

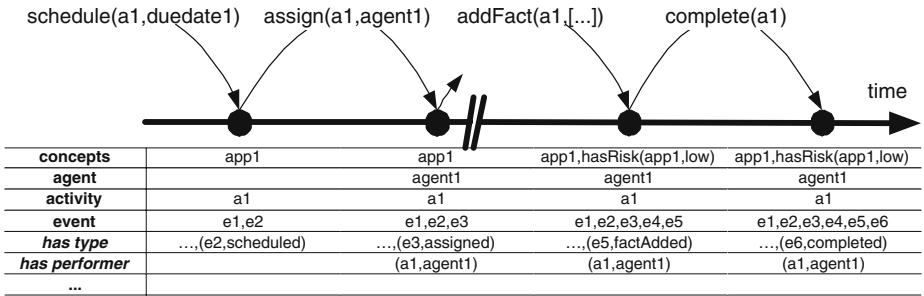


Fig. 3. An illustration of the state transitions for a ‘review credit’ activity ‘a1’

To make this text self-contained a number of concepts of the EM-BrA²CE Vocabulary are discussed in the remainder of this section. These concepts are depicted in Fig. 4. ‘Activity’ is a central concept in the vocabulary. An activity can either represent the act of performing an atomic unit of work or the act of coordinating a set of sub-activities (a business process). The former is called an atomic activity whereas the latter is called a composite activity. Each activity has an activity type.

An activity type is an SBVR:concept type that *specializes* the individual concept ‘activity’ and that *classifies* an activity.

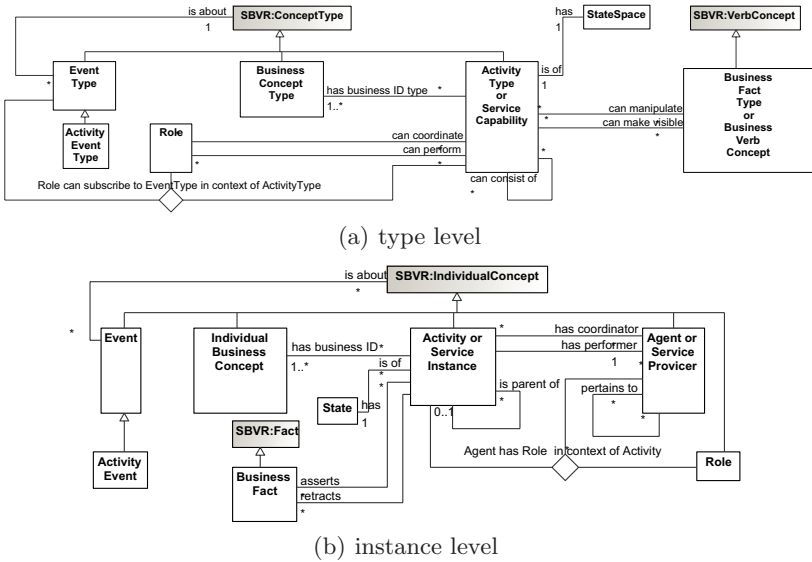


Fig. 4. A MOF/UML representation of the EM-BrA²CE Vocabulary

An activity *is* an SBVR:individual concept that represents a unit of (coordination) work to be performed by an agent.

In the context of a business process an agent can fulfill a particular role that represents an authorization to perform a number of activities. This conception of role is consistent with the Role Based Access Control (RBAC) standard [9,10,11].

A role *is* an SBVR:individual concept that represents a set of authorizations with regard to the performance of activities of given activity types. Example: the roles ‘applicant’, ‘sales representative’, ‘risk assessor’.

An agent *is* an SBVR:individual concept that represents an actor or a group of actors who can perform activities. Example: the agents ‘workerX’, ‘sales department’, ‘banking inc.’.

‘Agent can have role role’ *is* an SBVR:associative fact type that represents that an agent can assume a particular role. Example: ‘workerX can have role risk assessor’

‘Agent has role role in the context of activity’ *is* an SBVR:associative fact type that represents that an agent assumes a particular role in the context of an activity.

In the vocabulary, the state of an activity includes the history of events related to the activity or its sub-activities. Unlike many ontologies for business modeling, such as for instance the UFO [12], a distinction is made between activities and events. Activities are performed by agents and have a particular duration whereas events occur instantaneously and represent a state change in the world.

Changes to the life cycle of an activity are reflected by means of activity events. Activity events allow process modelers to distinguish between the activity state transitions that occur when, among others, creating, scheduling, assigning, starting and completing an activity.

An event *is* an SBVR:individual concept that corresponds to an instantaneous, discrete state change of a concept in the world.

An event type *is* an SBVR:concept type that *specializes* the individual concept ‘event’ and that *classifies* an event.

Agents that have a particular role in the context of a business process have the authorization to perform a particular activity. This authorization is expressed by the ‘can perform’ fact type. When performing an activity of a particular activity type, an agent can manipulate business facts of particular business fact types. This is expressed by the ‘can manipulate’ fact type. Additionally, agents can retrieve information about particular business fact types when performing activities. The business fact types that are visible are indicated by the ‘can make visible’ fact type.

‘Role can perform activity type’ represents the authorization that an agent that has a given role can perform an activity of a particular activity type. Note: These authorizations can be restricted by an activity authorization constraint. Example: sales representative can perform review credit.

‘Activity type can manipulate business fact type’ *is* an SBVR:associative fact type that represents that a business fact of type business fact type can be asserted or retracted during the performance of an activity of type activity type. Example: review credit can manipulate the business fact type ‘credit application has risk’.

‘Activity type can make visible business fact type’ *is* an SBVR:associative fact type that represents the business fact types that can be made visible in the context of activities of activity type. Note: visibility can be restricted by a visibility constraint. Example: review credit can make visible the business fact type ‘applicant has income’. Example: apply for credit can make visible the business fact type ‘credit application has reason of rejection’

The fact type ‘role can subscribe to event type in context of activity type’ expresses the visibility of events to agents in the context of an activity. Notice that it does not express how agents are notified of the event, which can generally occur using either a pull, a push or a publish-subscribe mechanism. Furthermore, it is possible that the visibility is constrained by so-called ‘event subscription constraints’.

‘role can subscribe to event type in context of activity type’ *is* an SBVR:associative fact type that expresses that an agent with a particular role can subscribe to an event of event type in the context of an activity of activity type. Example: An applicant can subscribe to completed in the context of review credit.

‘agent perceives event’ is an SBVR:associative fact type that expresses that an event is non-repudiable to a particular agent. Example: anAgentX perceives anEventY.

4 Specifying Access Constraints in SBVR

The EM-BrA²CE Vocabulary allows to specify access control policies that are able to refer to the state of a business process instance. In particular, each business process can be modeled by describing its state space and the set of business rules that constrain movements in this state space. For instance, the state space of the credit approval process is described by facts about the following concepts:

- **roles:** applicant, sales representative, risk assessor
- **atomic activity types:** apply for credit, review credit, make proposal, reject credit
- **activity event types:** created, assigned, started, completed
- **business concepts:** credit application, amount, collateral, applicant, income, risk
- **business fact types:** credit application has collateral, credit application has amount, applicant has income

Access control rules constrain the $assign(AId, AgentId, CoordinatorId)$ state transition. In the EM-BrA²CE Vocabulary [5] three kinds of kinds of access rules can be specified: activity authorization constraints, visibility constraints and event subscription constraints. An activity authorization constraint allows to constrain the agent-role assignments that can be granted to an agent. For instance, the fact ‘sales representative can perform review credit’ is constrained by the rule that credit applications larger than 2000 euros cannot be reviewed by employees of the sales department.

An activity authorization constraint is a structural business rule that dynamically constrains the activities that can be assigned to an agent based on the properties of the activity, the business facts in its state space and the properties of the agent.

Example: It is impossible that an agent that *is from department* sales department can perform a review credit activity that *has subject* credit application that *has an* amount larger than 2000 euros.

Example: It is impossible that an agent that *has been assigned to a* review credit activity that *has subject* credit application that *has an* amount larger than 2000 euros can perform a make proposal activity that *has subject* credit application

Access control does not only refer to the performance of activities, but also deals with the access to informational resources. In the context of an activity an agent may be able to perceive particular business facts. For example, the fact ‘apply for credit can make visible ‘credit application has reason of rejection’ implies that agents that apply for credit can see the reason of rejection if any.

A visibility constraint is a structural business rule that dynamically constrains the visibility of business facts to agents based on the properties of the business facts and the behavior of the agents.

Example: It is not possible that the fact type ‘credit application *has reason of rejection*’ is visible to the agent who *has role* applicant if the credit application *has type* consumer credit.

In addition to limiting access to business facts, access rules must also be able to limit the visibility of the behavior of agents to other agents. For example, the fact ‘a risk assessor *can subscribe to* started in the context of apply for credit’.

An event subscription constraint is a structural business rule that constrains the conditions under which agents who have a particular role in the context of an activity can perceive the occurrence of an activity event.

Example: It is not possible that an agent that *has role* risk assessor *perceives* a started event that *is about* an apply for credit activity that *has subject* a credit application that *has an amount of less than* 2000 euros.

When an event occurs, each agent who has a particular role in the context of the activity and whose role is subscribed to the event type and for whom no subscription constraints apply, can perceive the event. Consequently, the event is non-repudiable to external agents such that any legal obligation that results from the event can be enforced.

The proposed vocabulary and business rules allow to specify process-aware access control in four steps:

1. Define an **access control policy**. An access control policy is a non-actionable directive that identifies security threads and safety concerns and motivates access control implementation. Metamodels such as the OMG’s Business Motivation Model [13] provide the required structure to formulate access control policies.
2. Identify the access control **roles**. Roles are permissions involving the performance of activities or the involvement in activities that pertain to meaningful groups of activity types. Roles provide *stability* because role definitions do not change as often as agent-activity type assignments would. Fact types: ‘role can perform activity type’, ‘activity type can manipulate business fact type’, ‘activity type can make visible business fact type’ and ‘role can subscribe to event type in context of activity type’.
3. Make **agent-role assignments**. Agent-role assignment is the provisioning of agents with roles that represent access rights. Agent-role assignments can be defined by derivation rules, but for reasons of compliance and flexibility, agent-role assignments are often hard-coded. Fact type: ‘agent can have role role’.
4. **Specify access constraints**. Access constraints refine the role-based access control policy to take into account issues that are beyond the scope of user-role assignment. Access constraints give an access control model *precision*,

because they constrain the role-based access according to the properties of the agent, the activity and the business process event history.

5 Verbalizing Defeasible Access Control Rules

Access control specifications adhering to the role-based access control (RBAC) have a non-monotonic semantics that can be expressed in defeasible logic [14,15]. Defeasible logic is a means to formulate knowledge in terms of general rules and exceptions. To this end, defeasible logic allows for rules of which the conclusions can be defeated (defeasible rules) by contrary evidence provided by strict rules, other defeasible rules and defeaters. A superiority relation between rules indicates which potentially conflicting conclusions can be overridden. In EM-BrA²CE the ‘agent can perform activity’ fact type is defined using a rule set of two generic defeasible rules:

$$\begin{aligned} r_1 : true &\Rightarrow \neg canPerform(G, A), \\ r_2 : hasRole(G, R), hasType(A, At), canPerform(R, At) \\ &\Rightarrow canPerform(G, A) \end{aligned}$$

and a number of domain-specific activity authorization constraints, that translated to the following defeasible rules:

$$\begin{aligned} r_3 : A(3) &\Rightarrow \neg canPerform(G, A), \\ \dots \\ r_i : A(i) &\Rightarrow \neg canPerform(G, A), \\ \dots \\ r_n : A(n) &\Rightarrow \neg canPerform(G, A). \end{aligned}$$

The following priority relationship applies between the generic and domain-specific defeasible rules:

$$r_1 < r_2, r_2 < r_3, r_2 < r_4, \dots r_2 < r_n.$$

Because activity authorization constraints all potentially defeat the ‘agent can perform activity’ conclusion, they cannot contradict each other. As a consequence, no activity authorization constraint can be specified that can invalidate the outcome of another constraint. This is a valuable result, because it facilitates the incremental specification of access control policies [16]: new rules can be added without the conditions of previous rules need to be reconsidered. Ordinary rules, in contrast, require a complete, encyclopedic knowledge of all rules to be updated or decided upon. In addition to their enhanced representational capability, efficient implementations of defeasible logic have been proposed [17] together with visualization techniques to render the added complexity of default reasoning comprehensible to end users [18,19].

Although defeasible logic is naturally present in the role-based access control model, the current SBVR specification does not allow for the specification of defeasible rules. To cope with this shortcoming, defeasible access control rules must

be transformed into a set of ordinary, strict rules. Decision tables can be such a transformation mechanism. Decision tables are one of many visualizations [20] of rule sets. Figure 5 displays an example of a decision table. Graphically, a decision table consists of four quadrants. The two upper quadrants make up the condition sphere and the two lower quadrants represent the conclusion sphere. Properly built decision tables contain columns that represent exhaustive and mutually exclusive conjunctions of conditions and that have one or more conclusions. A set of input rules, possibly expressed using a defeasible logic, determines which combination of conditions in the upper right quadrant leads to which conclusion values in the lower right quadrant.

Several tools provide support in constructing decision tables [21,22,23,24]. In Prologa [22] decision tables are constructed by providing conditions with mutually exclusive condition labels, conclusions and input rules. Although Prologa fits in a propositional logic framework, Prologa proves a useful tool to visualize and transform a number of input rules, expressed using Prologa’s own default logic, into table rules. In particular, Prologa provides the user with a number of features that facilitate knowledge modeling, such as the reordering of condition labels to expand or contract the table, the syntactical verification of rules and the helpful visualizations that allow to semantically validate a rule set. The columns of the decision table in Fig. 5 represent the six ‘review credit’ access rules that are obtained by transforming the three defeasible rules of the access control policy. For instance, column 6 translate in the following SBVR rule:

It is necessary that an agent that *is not applicant of the credit application* and that *is not beneficiary of the credit application* and that *has not role risk assessor* and that *has role sales representative*, can not perform

review credit						
1. <u>agent is applicant of credit application</u>	Y	N				
2. <u>agent is beneficiary of credit application</u>	-	Y	N			
3. <u>agent has role risk assessor</u>	-	-	Y	N		
4. <u>agent has role sales representative</u>	-	-	-	Y	N	N
5. <u>credit application has amount larger than 2000 euros</u>	-	-	-	Y	N	-
1. <u>agent can perform the review credit activity</u>	-	-	x	x	-	-
2. <u>agent cannot perform the review credit activity</u>	x	x	-	-	x	x
	1	2	3	4	5	6

- R₁** In general, conclusion 2.
- R₂** In general, conclusion 1 if condition 3.
- R₃** In general, conclusion 1 if condition 4.
- R₄** In general, conclusion 2 if condition 1.
- R₅** In general, conclusion 2 if condition 2.
- R₆** In general, conclusion 2 if condition 4 and condition 5.

$$R_1 < R_2 < R_3 < R_4 < R_5 < R_6$$

Fig. 5. Transforming defeasible access rules into ordinary access rules

a review credit activity that *has subject credit application* that *has an amount more than 2000 euros*.

Six ordinary SBVR rules is the minimal number of rules required to capture the access control policy. Because SBVR rules do not allow defeasibility, the conditions of every activity authorization constraint must be repeated in almost every rule resulting in a larger number of longer rules.

6 Related Work

The Semantics of Business Vocabulary and Business Rules (SBVR) provides a number of conceptual vocabularies for modeling a business domain in the form of a vocabulary and a set of rules. The current SBVR specification [4] does not have a built-in vocabulary for expressing process-related concepts such as agent, activity, event or deontic assignment. Such vocabularies and formal semantics for expressing dynamic constraints are deferred to a later version of the SBVR standard [4]. Without such a vocabulary it is difficult to express process-aware access control rules. In [5] we have supplemented the SBVR with a vocabulary for declarative process modeling called the EM-BrA²CE Vocabulary.

The access control model of the EM-BrA²CE Vocabulary is based on the existing standard for Role-Based Access Control [9,10,11]. The existing RBAC standard allows to specify dynamic separation of duty (SoD) constraints. Such constraints impose that within a single session a user cannot assume two roles on which a dynamic SoD constraint applies. Strembeck et al. discuss an extension to the RBAC standard by allowing to express dynamic, process-related, access rules [25]. The proposed EM-BrA²CE Vocabulary goes further in that it allows to express process-aware access control policies that can declaratively relate to the current state of a business process and to a history of related business events.

The SBVR does by itself not allow for the specification of default rules or defeasible rules. Nonetheless, it is very common in natural language to use a kind of default logic to express access control policies. In this paper we have applied the long-lived practice of using decision tables as a transformation mechanism to transform default rules into a set of strict rules. On the output a (minimal set) of strict rules is produced that is equivalent to the defeasible input rules. A decision table visualisation serves a different purpose than for instance an immediate visualization of defeasible rules. Bassiliades et al., for instance, describe a visualization technique for defeasible logic rules based on directed graphs of so-called literal boxes and predicate boxes. Predicate boxes are connected with different connection types indicating the kind of rule or the priority relationship [18,19]. In correspondence to the Prologa tool, this technique is implemented at the level of the literals rather than at the level of terms, including variables and variable conditions. This graph-based technique seems an intuitive, user-friendly representation of defeasible rules. Decision tables, in contrast, serve the purpose of verifying a defeasible rule set and allow to transform the rule set into ordinary rules.

7 Conclusion

In this paper we have indicated how process-aware, role-based access control specifications can benefit from the upcoming SBVR standard. In particular, an SBVR vocabulary for process modeling was defined that allows to declaratively refer to the state of a business process when specifying access control rules. This vocabulary allows to model a business process by describing its state space and the set of business rules that constrain movements in this state space. The advantage is that access control rules are process aware, yet maintain a declarative nature by not specifying how and when an access rule must be enforced. To cope with the absence of defeasible logic in SBVR, we have shown how a set of defeasible access control rules can be transformed into SBVR access control rules using decision tables as a transformation mechanism. The SBVR offers many research opportunities to support the standard both in terms of linguistics and logics.

References

1. Securities and Exchange Commission, U.S.A.: Sarbanes Oxley Act 2002. Securities and Exchange Commission (SEC), U.S.A (2002)
2. Object Management Group: Business Process Modeling Notation (BPMN) – final adopted specification. OMG Document – dtc/06-02-01 (2006)
3. Chapin, D.: Semantics of Business Vocabulary & Business Rules (SBVR) [26]
4. Object Management Group: Semantics of Business Vocabulary and Business Rules (SBVR) – Interim Specification. OMG Document – dtc/06-03-02 (2006)
5. Goedertier, S., Vanthienen, J.: EM-BrA²CE v0.2: A Vocabulary and Execution Model for Declarative Process Models. Fetew research report, K.U.Leuven (2007), <http://www.econ.kuleuven.ac.be/public/ndbaf38/EM-BrAACE>
6. Baisley, D.E., Hall, J., Chapin, D.: Semantic Formulations in SBVR [26]
7. Unisys: Unisys rules modeler (2005) (10-11-2005), www.unisys.com
8. Digital Business Ecosystem (DBE): Sbeaver (2007), <http://sbeaver.sourceforge.net>
9. Sandhu, R.S., Coyne, E.J., Feinstein, H.L., Youman, C.E.: Role-based access control models. *IEEE Computer* 29(2), 38–47 (1996)
10. Ferraiolo, D.F., Sandhu, R.S., Gavrila, S.I., Kuhn, D.R., Chandramouli, R.: Proposed nist standard for role-based access control. *ACM Trans. Inf. Syst. Secur.* 4(3), 224–274 (2001)
11. InterNational Committee for Information Technology Standards (INCITS): Role-Based Access Control. American National Standard ANSI/INCITS 359-2004 (2004), <http://csrc.nist.gov/rbac>
12. Guizzardi, G., Wagner, G.: Ontologies and Business Systems Analysis. In: Rosemann, M., Green, P. (eds.) *Some Applications of a Unified Foundational Ontology in Business Modeling*, pp. 345–367. IDEA Publisher, USA (2005)
13. Object Management Group: Business Motivation Model (BMM) – adopted specification. OMG Document – dtc/2006-08-03 (2006)
14. Nute, D.: Defeasible Logic. In: *Handbook of Logic in Artificial Intelligence and Logic Programming*, pp. 353–395. Oxford University Press, New York (1994)

15. Antoniou, G., Billington, D., Governatori, G., Maher, M.J.: Representation results for defeasible logic. *ACM Trans. Comput. Log.* 2(2), 255–287 (2001)
16. Grosof, B.N., Labrou, Y., Chan, H.Y.: A declarative approach to business rules in contracts: courteous logic programs in XML. In: *ACM Conference on Electronic Commerce*, pp. 68–77. ACM Press, New York (1999)
17. Maher, M.J., Rock, A., Antoniou, G., Billington, D., Miller, T.: Efficient defeasible reasoning systems. *International Journal on Artificial Intelligence Tools* 10(4), 483–501 (2001)
18. Bassiliades, N., Kontopoulos, E., Antoniou, G.: A visual environment for developing defeasible rule bases for the semantic web. In: *Adi, A., Stoutenburg, S., Tabet, S. (eds.) RuleML 2005. LNCS, vol. 3791*, pp. 172–186. Springer, Heidelberg (2005)
19. Kontopoulos, E., Bassiliades, N., Antoniou, G.: Visualizing defeasible logic rules for the semantic web. In: *Mizoguchi, R., Shi, Z., Giunchiglia, F. (eds.) ASWC 2006. LNCS, vol. 4185*, pp. 278–292. Springer, Heidelberg (2006)
20. Antoniou, G., Taveter, K., Berndtsson, M., Wagner, G., Spreeuwenberg, S.: A First-Version Visual Rule Language. Report IST-2004-506779, REWERSE (2004)
21. Vanthienen, J., Robben, F.: Developing legal knowledge based systems using decision tables. In: *ICAAIL*, pp. 282–291 (1993)
22. Vanthienen, J., Mues, C., Aerts, A.: An Illustration of Verification and Validation in the Modelling Phase of KBS Development. *Data Knowl. Eng.* 27(3), 337–352 (1998)
23. Spreeuwenberg, S., Gerrits, R., Boekenoogen, M.: Valens: A knowledge based tool to validate and verify an aion knowledge base (2000)
24. Vanthienen, J., Mues, C.: Prologa 5.3 - tabular knowledge modeling (2005)
25. Strembeck, M., Neumann, G.: An integrated approach to engineer and enforce context constraints in rbac environments. *ACM Trans. Inf. Syst. Secur.* 7(3), 392–427 (2004)
26. W3C Workshop on Rule Languages for Interoperability, 27-28 April 2005, Washington, DC, USA. In: *Rule Languages for Interoperability, W3C* (2005)