Unified Patterns to transform business rules into an event coordination mechanism

Willem De Roover and Jan Vanthienen

Department of Decision Sciences & Information Management,
Katholieke Universiteit Leuven, Belgium
(willem.deroover;jan.vanthienen)@econ.kuleuven.be

Summary. Business rules define and constrain various aspects of the business, such as vocabulary, behavior and organizational issues. Enforcing the rules of the business in information systems is however not straightforward, because different mechanisms exist for the (semi-)automatic transformation of various business constraints and rules. In this paper, we examine if and how business rules, not only data rules, but also process rules, timing rules, authorization rules, etc., can be expressed in SBVR and translated using patterns into a more uniform event mechanism, such that the event handling could provide an integrated enforcement of business rules of many kinds.

Keywords: business rules, event coordination, business processes, SBVR, declarative process modeling

1 Introduction

Enforcing the various rules of the business in information systems is not straightforward, because different mechanisms exist for the transformation of business constraints, process rules, timing rules, access control rules, or other rules into model driven implementations, leading to partial solutions for process management, data constraints, audit constraints, etc.

In this paper, we examine if and how business rules can be translated into a more uniform event mechanism, such that the event handling could provide an integrated enforcement of business rules of many kinds. To this end, we provide a pattern mechanism to transform SBVR (Semantics of Business Vocabulary and Business Rules) [1] integrity constraints and derivation rules into event-driven enforcement rules. We also use an extension of SBVR to declaratively model business processes [2] and use similar patterns to transform the process rules into event driven process enactments. The result is a set of event rules, enabling an integrated enforcement of business rules of many kinds.

The paper is structured as follows. In section 2 we describe the use of SBVR for vocabulary constraints and process constraints. The different types of business rules are identified in section 3. In section 4 we examine some example transformation patterns. Finally, in section 5 we relate the approach to the relevant literature.

2 The need for a unified framework

Business rules should be on the one hand comprehensible so that they can be understood by business people and on the other hand formal so that they can be enforced by information systems. The Semantics of Business Vocabulary and Business Rules (SBVR) is a language for business modeling that has such property [1], as long as it is extended with a vocabulary for expressing process-related concepts.

2.1 SBVR for vocabulary constraints

The Semantics of Business Vocabulary and Business Rules (SBVR) is a new standard for business modeling within the Object Management Group (OMG). 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. In addition to 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. The SBVR specification defines a structured, English vocabulary for describing vocabularies and verbalizing rules, called SBVR Structured English [1]. One of the techniques used by SBVR structured English are font styles to designate statements with formal meaning. In particular,

- the term font (green) is used to designate a noun concept.
- the name font (green) designates an individual concept.
- the *verb* font (blue) is used for designation for a verb concept.
- the keyword font (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.

2.2 Procedural versus Declarative Process Modeling

A business process model is called **procedural** when it contains explicit information about how processes should proceed, but only implicitly keeps track of why these design choices have been made, the underlying business rules. Procedural process models are modeled with **procedural languages** such as the Business Process Modeling Notation (BPMN) and UML Activity Diagrams. These languages predominantly focus on the control-flow perspective of business processes. In such process languages it might be possible to **enforce business rules** using a control-flow-based modeling construct. For instance, the enforcement of a derivation or integrity constraint can be directly modeled as a calculation or input validation step, but the disadvantage of procedural process modeling is that business rules cannot be formulated independently from the process models in which they are to be enforced.

The counterpart of a procedural process model is a declarative one. Process modeling is said to have a **declarative** nature, when it explicitly takes into account the business concerns that govern business processes and leaves as much freedom as is permissible at execution time for determining a valid and suitable execution scenario. Examples of declarative languages are: the case handling paradigm [3], the constraint specification framework of Sadiq et al. [4], the ConDec language [5] and the PENELOPE language [6]. An overview is given in [7]. Declarative process modeling separates business rule modeling from business rule enforcement. In particular, it does not make use of control flow to indicate when and how business rules are to be enforced [8]. Instead, it is left to the execution semantics of the declarative process models to define an execution model in which different business rule types are automatically enforced.

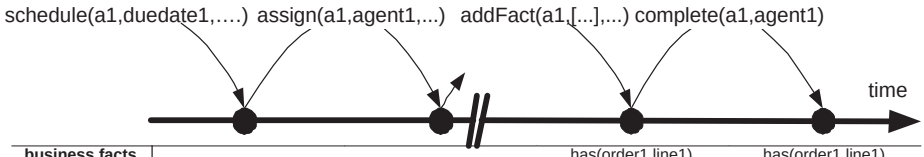
Procedural process models depict communication logic in a procedural manner, because they specify how and when business events are communicated and information is transmitted. Declarative process models are only concerned with the ability of business agents to perceive business events and business concepts. When an agent can perceive a particular event, the event becomes non-repudiable to the agent, irrespective of how the agent is notified of the event. The execution semantics of a declarative process model determines how events are communicated. In particular, events can be communicated as messages that are sent by the producer (push model), retrieved by the consumer (pull model) or via a publish-subscribe mechanism. This declarative modeling style enhances design-time flexibility, as it allows to model business processes irrespective of the used communication channels.

2.3 SBVR for process constraints

SBVR is a suitable base language for defining process-aware rules, but it does not contain a vocabulary with process related concepts such as agents, activities, process states and events. In [2, 9] we defined an SBVR vocabulary for expressing process-related concepts, called the EM-BrA^{2}CE Vocabulary. EM-BrA^{2}CE 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. 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. Each activity in a process instance can undergo a number of distinct state transitions. Business rules determine whether or not a particular state transition can occur.

The following state transitions are e.g. considered: create, assign, updatefact, complete. In [2] 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 1 illustrates a number of state transitions that occur to a given place order 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.



business facts			has(order1,line1)	has(order1,line1)
agent		agent1	agent1	agent1
activity	a1	a1	a1	a1
event	e1,e2	e1,e2,e3	e1,e2,e3,e4,e5	e1,e2,e3,e4,e5,e6
has type	...,(e2,scheduled)	...,(e3,assigned)	...,(e5,factAdded)	...,(e6,completed)
has performer		(a1,agent1)	(a1,agent1)	(a1,agent1)
...				

Fig. 1. An illustration of the state transitions for a place order activity a1

In the vocabulary, the state of an activity (or service instance) 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 [10], 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.

3 Business Rule Types

Given the SBVR vocabulary for process-related concepts, each business process can be modeled by describing its state space and the set of business rules that constrain the possible transitions in this state space. For instance, the state space of an order-to-cash process is described by the following concepts:

- **composite activity types**: coordinate sales order
- **atomic activity types**: place order, accept order, reject order, pay, ship
- **activity event types**: created, assigned, started, completed
- **business concepts**: order, order line
- **business fact types**: order *has* order line, order *is critical*,...

Business rules come in different forms (structural/definitional rules, derivation rules, behavioral rules, permissions and obligations), and refer to different aspects (data, behavior, organization). In [2] a total of sixteen business rule types are identified that can constrain specific activity state transitions, as indicated in Table 1. They refer to one of the three aspects of business process modeling that are generally considered [11]: control-flow, data and organizational aspects.

For reasons of brevity, only a number of these business rule types are included in this text.

Table 1. Business rule types

aspect	business rule type	related work
control flow	**Temporal deontic rule**	[12],[6]
	Activity precondition	[13]
	Activity postcondition	[13],[3]
	Dynamic integrity	[14]
	Activity cardinality	[5]
	Serial activity constraint	[4]
	Activity order	[4],[5]
	Activity exclusion	[4],[5]
	Activity inclusion	[4],[5]
	Reaction rule	[12]
data	**Static integrity**	[14]
	Derivation rule	[14]
organization	**Activity authorization**	[15]
	Activity allocation rule	
	Visibility constraint	[15]
	Event subscription	[15]

Control-flow Aspects. Business policy and regulations contain a lot of constraints (partial order, timing, exists, activity pre- and postconditions). In a trade community, for instance, different business protocols lay down the obligations of business partners and can be expressed in the form of temporal deontic rules [6].

Data aspects. The performer of an activity can perform particular manipulations (addition, removal or update) of business facts. These state transitions can be constrained by integrity constraints and derivation rules.

Organizational aspects. Organizational aspects relate to the visibility of business concepts and events and the authorization to perform particular activities.

4 Example patterns for transforming business rules into event rules

We examine how various business rules can be translated into more uniform event rules, such that the event handling could provide an integrated enforcement of business rules of many kinds, not only process rules, but also data rules, timing rules, authorization rules and others. To this end, we provide a pattern mechanism to transform SBVR (Semantics of Business Vocabulary and Business

Rules) integrity constraints, derivation rules and process rules into event-driven enforcement rules and notifications.

4.1 Data constraints and derivations

Example patters for integrity constraints and derivations are shown in figures 2, 3 and 4.

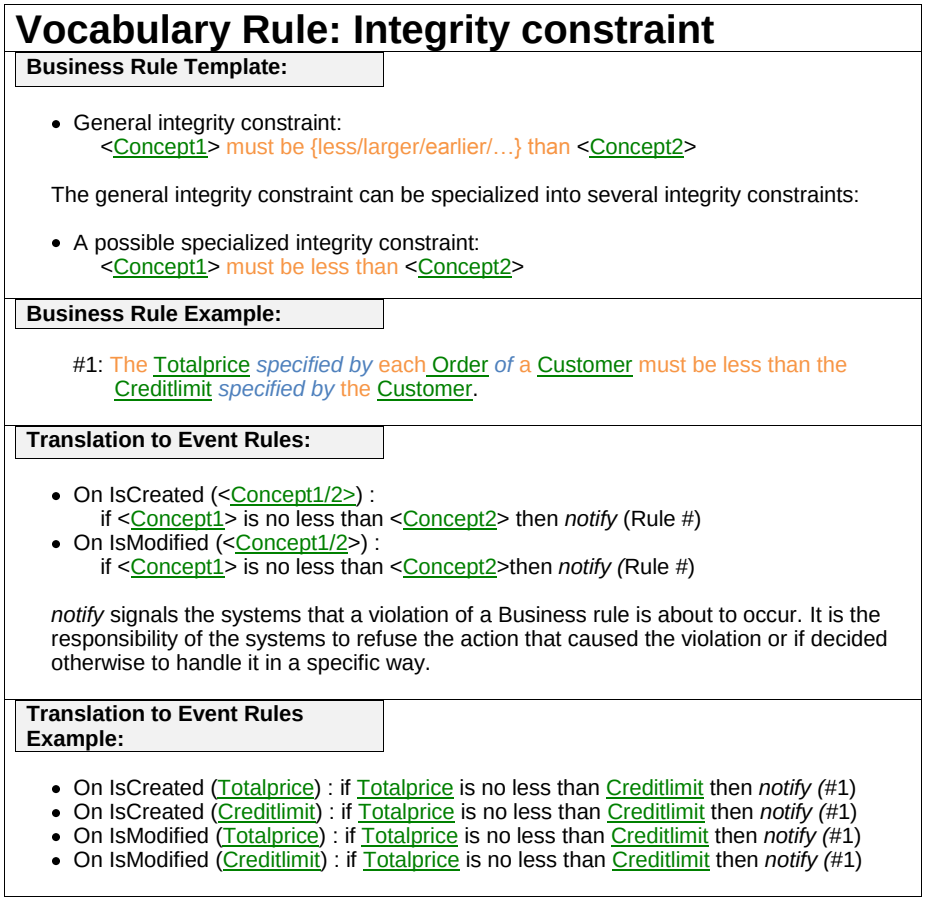


Fig. 2. Integrity Constraint

For each type of business rule we have defined a general template. The use of templates limits the ways in which rules can be formulated, but in this way it will be easy to extract the necessary information from a business rule. This information includes the type of the business rule and the concepts used in the rule. We use this information in event based rules and notifications. For each

type of business rule we have defined corresponding Event-Condition-Action (ECA) rules. The extracted concepts from the business rule are filled in into the corresponding ECA rule. The sets of ECA rules are equivalent to the business rules that they express. However ECA rules have the advantage that they make clear when they have to be checked. The condition of a ECA rule checks whether the business rule is violated and in case of a violation the system will be notified of this violation.

Some business rules will also generate events. This is the case when a business rule changes the value of some concept. Derivation rules e.g. calculate the value of a concept based on other concepts. These rules will generate an event that signals that the value of the calculated concept has changed.

4.2 An example

The following rule stated in [16] explains our case: *The total value of a customers unpaid orders must not exceed his credit limit.* This rule will have to be checked at several points in the execution of some processes as indicated in [16]:

- When a customer submits an new order
- When a customer changes an existing order (adds items,changes quantities, substitutes products)
- When a customers credit limit is changed
- When product prices are changed (unless prices are frozen at order time)
- and for any other relevant events the system recognizes.

In figure 5 three business rules are presented with their corresponding event rules and notifications. As the three rules are closely related to each other, changes that occur due to one rule can be propagated to other rules. For example, if the LinePrice of an OrderLine is recalculated due to changes in ProductPrice or Amount then this results in an event that signals that the LinePrice has changed. This event is handled by an ECA-rule generated from rule #b and leads to the recalculation of the TotalPrice. The change of TotalPrice will be signaled to the system by means of a new event. This will trigger all event rules that act on changes to TotalPrice including an ECA rule generated from rule #a. This rule will check if the new TotalPrice is no less than the specified CreditLimit. If this is the case, the system will notify this violation.

4.3 Control flow

The approach is not limited to data rules. It is possible to develop patterns for control flow and organization rules, as already indicated in [17, 18, 19, 20]. As SBVR does not provide process related concepts, we used the concepts provided by the EM-BA^{2}CE framework. For the sake of simplicity we present these concepts as simple SBVR fact types in our patterns. Figures 6 and 7 present two patterns for transforming control flow and organizational rules into event-driven enforcement rules and notifications.

Vocabulary Rule: Derivation rule

Business Rule Template:

- *General derivation rule:*
  <Concept1> must be computed as <calculation>

The general derivation rule can be specialized into several derivation rules.

- *A specialized derivation rule:*
  <Concept1> must be computed as <Concept2> {plus /minus/ times /divided by} <Concept3>

This derivation rule is used to explain the mechanism for converting derivation rules into event rules and events.

Business Rule Example:

#2: The LinePrice specified by each Orderline must be computed as the ProductPrice specified by the Product of the Orderline times the Amount specified by the Orderline

Translation to Event Rules:

- *Create the following rules:*
  - *On IsCreated (<Concept1>) : compute (<Concept1>)*
  - *On IsModified (<Concept2>) : compute (<Concept1>)*
  - *On IsModified (<Concept3>) : compute (<Concept1>)*
- *Signal the following event:*
  - *On compute (<Concept1>) : signal IsModified (<Concept1>)*

compute will calculate the value of Concept1 based on the given business rule.

Translation to Event Rules Example:

- *Create the following rules:*
  - *On IsCreated (LinePrice) : compute (LinePrice)*
  - *On IsModified (ProductPrice) : compute (LinePrice)*
  - *On IsModified (Amount) : compute (LinePrice)*
- *Signal the following event:*
  - *On compute (LinePrice) : signal IsModified (LinePrice)*

Fig. 3. Derivation Rule

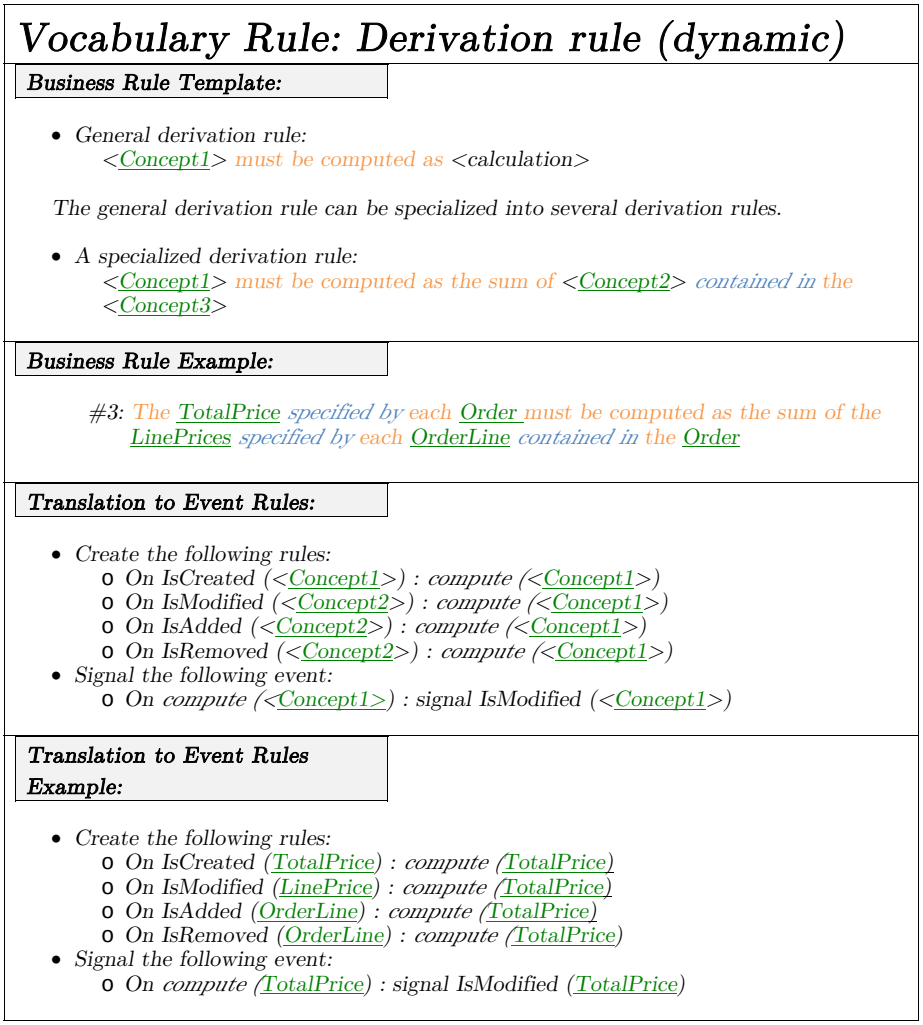


Fig. 4. Derivation Rule (dynamic)

5 Evaluation

Languages for declarative process modeling often do not cover the many real-life business concerns that exist in reality. Some only allow to express business rules about sequence and timing constraints, i.e. the control-flow perspective, others include the organizational and data model aspects, but do not provide a temporal logic to express temporal relationships between concepts such as activities or events. Moreover, these languages make use of very different knowledge representation paradigms. These heterogeneous knowledge representation paradigms

The CrediLimit example

#a: The Totalprice *specified by* each Order *of* a Customer must be less than the Creditlimit *specified by* the Customer.

#b: The TotalPrice *specified by* each Order must be computed as the sum of the LinePrices *specified by* each OrderLine *contained in* the Order

#c: The LinePrice *specified by* each Orderline must be computed as the ProductPrice *specified by* the Product *of* the Orderline times the Amount *specified by* the Orderline

- Create the following rules for #a :
  - On IsCreated (Totalprice): if Totalprice is no less than Creditlimit then *notify (*#a)
  - On IsCreated (Creditlimit): if Totalprice is no less than Creditlimit then *notify (*#a)
  - On IsModified (Totalprice): if Totalprice is no less than Creditlimit then *notify (*#a)
  - On IsModified (Creditlimit): if Totalprice is no less than Creditlimit then *notify (*#a)
- Create the following rules for #b :
  - On IsCreated (TotalPrice) : *compute* (TotalPrice)
  - On IsModified (LinePrice) : *compute* (TotalPrice)
  - On IsAdded (OrderLine) : *compute* (TotalPrice)
  - On IsRemoved (OrderLine) : *compute* (TotalPrice)
- Signal the following event for #b:
  - On *compute (*TotalPrice) : signal IsModified (TotalPrice)
- Create the following rules for #c:
  - On IsCreated (LinePrice) : *compute* (LinePrice)
  - On IsModified (ProductPrice) : *compute* (LinePrice)
  - On IsModified (Amount) : *compute* (LinePrice)
- Signal the following event for #c:
  - On *compute (*LinePrice) : signal IsModified (LinePrice)

Fig. 5. Credit Limit example

raise the question how to reason about such heterogeneously expressed knowledge.

Moreover, not all these languages have an explicit execution model or they have an execution model that explicitly assumes either human or machine-mediated service enactment. The EM-BrA^{2}CE framework with its formal execution model [2] makes abstraction of the differences between humans and machines. Coordination work such as creating, scheduling, assigning, skipping, aborting or redoing an activity can then be performed by humans, machines or both.

6 Conclusion

In this paper, we have examined if and how business rules in SBVR, not only data rules, but also process rules, timing rules, authorization rules, etc., can be translated using patterns into a more uniform event mechanism, such that the event handling could provide an integrated enforcement of business rules of many kinds. Future work consists of developing a tool that uses these templates to transform SBVR rules into ECA rules and creates an execution model that is compliant with these rules.

Behavior Rule: Timed precedence of activities

Business Rule Template:

<Activity2> may ... only <time constraint> after <Activity1>

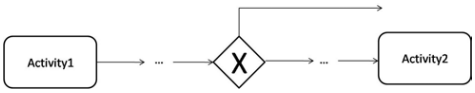
Business Rule Example:

- *Activities:*
  - *Activity1: Trainee applies for license*
  - *Activity2: Trainee takes practical car examination*
- *Business Rule:*
  #5: A trainee may take a practical car examination only within 1 year after that trainee has applied for a license

Remarks:

Activity2 can only be performed (a limited time) after Activity1 has been performed. However performing Activity1 does not imply that Activity2 will be performed.

Visual Representation:



The representation makes clear that this rule only puts a constraint on the execution of Activity2.

Translation to Event Rules:

- *On start (<Activity2>) : if not ended(<Activity1>) or (<Activity2> expired) then notify (Rule #)*
- *Add the following events to the event list:*
  - *on <time constraint> : signal that <Activity2> is expired.*
  - *on <time constraint * [notice factor]> : signal that <Activity2> will expire.*

  The event list keeps track of events that will have to happen in the future. Every event in the event list will have a timer. If the timer expires that event will be triggered.

Translation to Event Rules Example:

- *On start (trainee takes a practical car examination) : if not end(trainee applies for license) then notify (#5)*
- *Add the following events to the event list:*
  - *On 1 year : signal that trainee takes practical car examination is expired.*
  - *On 0.9 year : signal that trainee takes practical car examination will expire.*

Fig. 6. Control flow: timed precedence

Management Rule: Authorization

Business Rule Template:

<Concept1> that <verb phrase><Concept2> must be different from <Concept3> that <verb phrase><Concept4>

Business Rule Example:

#6: The Person1 that *applies for* a Loan must be different from the Person2 that *approves* the Loan

Translation to Event Rules:

- *On IsCreated (<Concept1> <verb phrase> <Concept2>):*
  if <Concept1> is equal to <Concept3> then notify (Rule #)
- *On IsCreated (<Concept3><verb phrase> <Concept4>) :*
  if <Concept1> is equal to <Concept3> then notify (Rule #)
- *On IsModified (<Concept1>) :*
  if <Concept1> is equal to <Concept3> then notify (Rule #)
- *On IsModified (<Concept3>) :*
  if <Concept1> is equal to <Concept3> then notify ((Rule #)

Translation to Event Rules Example:

- *On IsCreated (Person2 approves Loan) :*
  if Person1 is equal to Person2 then notify (#6)

Remarks:

- *There is no need to check any other event rules in the example.*
  - *Applying for a loan always happens before the loan is approved, this can be enforced by a behavioural rule, therefore it is not necessary to check the rule when a person applies for a loan.*
  - *Approving a loan happens at one point in time. In this example we only keep track of the actual approver, not any planned approver. Therefore it is not necessary to keep track of the changes before the actual approval.*

Fig. 7. Authorization rule

References

1. Object Management Group: Semantics of Business Vocabulary and Business Rules (SBVR) – Interim Specification. OMG Document – dtc/06-03-02 (2006)
2. Goedertier, S., Haesen, R., Vanthienen, J.: EM-BrA^{2}CE v0.1: A vocabulary and execution model for declarative business process modeling. FETEW Research Report KBI_0728, K.U.Leuven (2007)
3. van der Aalst, W.M.P., Weske, M., Grünbauer, D.: Case handling: a new paradigm for business process support. Data & Knowledge Engineering **53**(2) (2005) 129–162
4. Sadiq, S.W., Orlowska, M.E., Sadiq, W.: Specification and validation of process constraints for flexible workflows. Information Systems **30**(5) (2005) 349–378

5. Pesic, M., van der Aalst, W.M.P.: A declarative approach for flexible business processes management. In: Business Process Management Workshops. (2006) 169–180
6. Goedertier, S., Vanthienen, J.: Designing compliant business processes with obligations and permissions. In Eder, J., Dustdar, S., eds.: Business Process Management Workshops. Volume 4103 of Lecture Notes in Computer Science., Springer (2006) 5–14
7. Goedertier, S., Vanthienen, J.: An overview of declarative process modeling principles and languages. Communications of SWIN **6** (April 2009) 51–58
8. Morgan, T.: Business Rules and Information Systems: Aligning IT with Business Goals. Addison-Wesley Professional (2002)
9. Goedertier, S., Mues, C., Vanthienen, J.: Specifying process-aware access control rules in SBVR. In Paschke, A., Biletskiy, Y., eds.: Proceedings of the International Symposium Advances in Rule Interchange and Applications (RuleML 2007). Volume 4824 of Lecture Notes in Computer Science., Springer (2007) 39–52 (Best Paper Award).
10. Guizzardi, G., Wagner, G.: in: Ontologies and Business Systems Analysis, ed. M. Rosemann and P. Green. In: Some Applications of a Unified Foundational Ontology in Business Modeling. IDEA Publisher (2005) 345–367
11. Jablonski, S., Bussler, C.: Workflow Management. Modeling Concepts, Architecture and Implementation. International Thomson Computer Press, London (1996)
12. Paschke, A., Bichler, M., Dietrich, J.: Contractlog: An approach to rule based monitoring and execution of service level agreements. In Adi, A., Stoutenburg, S., Tabet, S., eds.: First International Symposium Advances in Rule Interchange and Applications (RuleML 2005). Volume 3791 of Lecture Notes in Computer Science., Springer (2005) 209–217
13. Roman, D., Keller, U., Lausen, H., de Bruijn, J., Lara, R., Stollberg, M., Polleres, A., Feier, C., Bussler, C., Fensel, D.: Web service modeling ontology. Applied Ontology **1**(1) (2005) 77–106
14. Wagner, G.: The agent-object-relationship metamodel: towards a unified view of state and behavior. Information Systems **28**(5) (2003) 475–504
15. Strembeck, M., Neumann, G.: An integrated approach to engineer and enforce context constraints in RBAC environments. ACM Transactions on Information System Security **7**(3) (2004) 392–427
16. Ross, R.: Business Rule Concepts, Third Edition. Business Rule Solutions, LLC (2009)
17. Pesic, M.: Constraint-based workflow management systems: Shifting control to users. PhD thesis, Eindhoven University of Technology (2008)
18. van der Aalst, W.M.P., Pesic, M.: Decserflow: Towards a truly declarative service flow language. In Leymann, F., Reisig, W., Thatte, S.R., van der Aalst, W.M.P., eds.: The Role of Business Processes in Service Oriented Architectures. Volume 06291 of Dagstuhl Seminar Proceedings., Internationales Begegnungs- und Forschungszentrum fuer Informatik (IBFI), S chloss Dagstuhl, Germany (2006)
19. Wang, M., Wang, H.: From process logic to business logic–A cognitive approach to business process management. Information & Management **43**(2) (2006) 179–193
20. Ceponiene, L., Nemuraite, L., Vedrickas, G.: Separation of event and constraint rules in uml & ocl models of service oriented information systems. Information Technology and Control **38**(1) (2009) 29–37