ISSN 1392 – 124X INFORMATION TECHNOLOGY AND CONTROL, 2009, Vol. 38, No. 1

SEPARATION OF EVENT AND CONSTRAINT RULES IN UML&OCL MODELS OF SERVICE ORIENTED INFORMATION SYSTEMS

Lina Ceponiene, Lina Nemuraite, Gediminas Vedrickas

Kaunas University of Technology, Department of Information Systems
Studentu st. 50-315a, LT-51368 Kaunas, Lithuania
Lina.Ceponiene@ktu.lt, Lina.Nemuraite@ktu.lt, g.vedrickas@erp.eu

Abstract. In this paper, possibilities of advancing Business Process Modelling when joining it with Business Rules approach are analysed. The problem currently under discussion in business rule and business process management communities is that business process (or event) rules are changing independently from structural, or constraint-oriented business rules, and coupling them together requires changing business processes when rules are changing, and vice versa. The paper addresses the problem of modelling – separating business constraint rules from event rules in UML&OCL models. The proposed principles of separation are based on UML 2 state machines applied in the context of Extended Model Driven Approach for Service Oriented Information Systems. Representation of event and constraint rules in SBVR, Template Based Language and OCL is analysed.

Keywords. Business rules, event rules, constraint rules, state machines, UML, OCL, MDE.

1. Introduction

Existing Model driven approaches as Model Driven Architecture (MDA), Model Driven Development (MDD) and Model Driven Engineering (MDE) are taking a very little care about business rules. J. D. Haan [18] has mentioned eight reasons why Model Driven approaches may fail. He argues that the main goal of MDE is to reduce the vulnerability of software artefacts regarding changes; however, currently addressed changes are mainly related with implementation platforms and not with business changes. Currently business rules management systems and business rule engines are addressing these needs by enabling the non-technical users to make changes to their software. Model Driven approaches are limited as they do not treat business rules according to the Business Rule approach.

One of the most fundamental principles of developing the software is separation of concerns. Concerns that could be captured, analyzed, developed, implemented and maintained separately should be separated. One of such concerns is business rules that usually are embedded by developers in use cases, interactions, state machines, class and other models. As graphical modelling languages are not capable to express all real life semantics that should be implemented by software, graphical model elements are supplemented with textual expressions (informal, e.g. natural language, or formal, e.g. OCL [29]). Properly separated from visual models, business rules could be updated by business people, and translated to system perspective and backwards by IT people. J. D. Haan [18] maintains that improved business and IT alignment requires a shared language and such a language should be business rules that allow for IT and business people to work together. We additionally argue that MDE CASE tools should have repositories of business rules that should be used during modelling business and generating software code – similarly as business rule engines are using business rules during execution [16, 17, 19].

Object Constraint Language (OCL) is the language designed for expressing business rules in UML models. Recent analysis has shown [28] that UML&OCL models are capable to express all types of executable business rules subsumed in their different classifications and typologies [11, 40, 41]. However, in these business rule representations OCL expressions are tangled with UML graphical elements. On the one hand, augmenting visual models with elements of business rules helps developer to anticipate modelling; on the other hand, it is purposeful to separate business rule expressions from graphical elements to facilitate changes. The problem currently under discussion in business rule and business process management communities is that business process (or event) rules are changing independently of other (structural, or constraint oriented) business rules and coupling them together requires changing business processes when rules are changing, and vice versa. So there exists a problem of modelling – how to separate constraint-oriented business rules from process-oriented rules in UML&OCL behavioural models?

The rest of the paper[1] is organized as follows. In section 2 the related work is considered. Section 3 presents the classification of business rules and introduces constraint and event rule types. Section 4

[1] The work is supported by Lithuanian State Science and Studies Foundation according to High Technology Development Program Project "VeTIS" (Reg.No. B-07042)

is devoted for representation of process rules with UML state machines. In section 5 rule representations in SBVR, Template Based Language and OCL are considered in the context of EMDA process. Section 6 draws conclusions and highlights the future work.

2. Related Work: Accelerating Model Driven Development Process with Business Rule Approach

Though OMG has issued Semantics of business Vocabulary (SBVR) standard [38] and is working on Production Rule Representation [33], Business Rule approach initiated by R. G. Ross [35, 36] and Business Rule group [11] yet has a few applications within MDE. The advantages of such a development are well understood [22], but for the meantime MDE lacks standards and methodologies for modelling executable business rules and transition from semantic business rules, i.e. Computation Independent models (CIM), to executable, Platform Independent models (PIM).

For this purpose, RuleML initiative and REWERSE group are creating interchangeable specifications for various kinds of rules, devoted for Semantic Web and Object-Oriented systems. URML [25] is an interesting approach of REWERSE group for visual modelling of derivation and production rules, implemented in UML CASE tool "Strelka". Rules are represented as first class entities in class models and have relationships (supplemented with expressions) with concepts involved. This approach differs from practice of expressing business rules in object-oriented models and may be inefficient for large sets of business rules.

There already are proposals for simple transformations of SBVR specifications to UML models [23, 34]. The Model Driven Enterprise Engineering (MDEE) methodology created by KnowGravity is one of the first efforts to apply OMG SBVR and other standards in the holistic IS development process where information technologies are managed by business needs [37]. MDEE supports the smooth going from SBVR structural and operative rules to PIM Constraints, ECA and CA (Condition-Action) rules. It uses fact diagrams for representing business vocabularies; UML class, use case diagrams and state machines for representing system models; and BPMN for modelling business processes. The final PIM is presented by executable state machines with KnowGravity expressions for business rules that are proprietary solution requiring hard manual efforts; nevertheless, MDEE is an excellent evidence of usefulness and applicability of OMG standards. The prototype, proposed by M. H. Linehan [23] for transformation of limited SBVR rules to OCL pre-conditions, also is related with business process modelling. OCL constraints are addressed in [6, 9, 42, 43].

The simple, but proven BROOD approach recently published in [24] proposes simple templates for specification of restricted typology of business rules, and simple object-oriented development process that augments UML by explicitly considering business rules as an integral part of an object-oriented development. BROOD process is supported by a tool developed on top of the Generic Modelling Environment (GME). This approach (though it is not related with OCL) has many common points with our efforts and represents modern trends in MDE [12], i.e. development of modelling environments tailored for specific domains and specific development methodologies.

Currently, new technologies complementing event manipulating in database systems [27] are arising such as Complex Event Processing, Event Driven Architecture, Event Servers and Event Rule Engines, for example, [10, 21, 26] and others. These technologies are related with the further enhancement of Web Service Architecture, Business Process Management and Business Rule approach. Separation of processes and constraints may be done on the base of events that are already addressed in Event Driven technologies; however, they are lacking modelling support. There are some proposals how to separate process rules from constraints. T. Graml et all [15] externalize decisions, data constraints and activity compositions. S. Goedertier and J. Vanthienen [14] propose a declarative process modelling language that extends SBVR business vocabulary categorising sixteen business rule types. Most of these rules are expressed in terms of activity states and events instead of states of business objects. This approach may lead to desired separation of business processes and business constraints, despite it does not consider complex events and can be more difficult to understand than the graphical process notations. F. Bry et all [4] have proposed the Xchange language for representing complex events, however, in this approach constraint and event rules are coupled. The idea of T. van Eijndhoven et all [13] is the identifying the variable and non-variable process segments and combining workflow patterns that model the behaviour of each variant by means of business rules, but it also does not solve the aforementioned problem. D. Bugaite and O. Vasilecas [5] investigate related, but rather different viewpoint: how events and rules are linking together in business system, information system and software system levels.

The mentioned problems of separating process rules from business constraints are especially important in Model Driven Development processes of service-oriented information systems. We are working in creation of enhanced development process which was called "Extended Model Driven Approach" (EMDA). We have coped with limitations of Model Driven Architecture of using PIM and PSM and therefore introduced Design-independent model (DIM) for representation of

requirements in MDD process [7]. Currently MDA also acknowledges the necessity of involving more layers (or dimensions) of models into the Model Driven Development processes (as in the more powerful model-driven methodology "Model Driven Engineering" (MDE) introduced by S. Kent [20]).

Other characteristic of our method is the precise conceptual modelling [31, 32] during requirement phase and reconciliation of conceptual data model with behavioural model. The resulting requirement model (DIM) is represented as a class diagram containing entities and abstract interfaces to the system under development [7, 8]. In our previous work, we have introduced several types of events and OCL expressions facilitating separation of constraint-oriented rules and process-oriented rules (we name them "event rules") during EMDA process. In the current paper, we develop this idea for other types of event rules and are intending to extend the State Coordinator pattern for processing complex events that are inherent for service-oriented systems.

3. Constraint rules and event rules

One of the problems of applying Business Rule approach is a proper specification of business rules separating rules related with business processes, from rules related with business constraints. Such a separation allows increasing the agility of Business Rule approach because constraint rules are changing faster and independently of process rules. OMG has issued the SBVR standard for specification of "real" business rules (i.e. rules under business jurisdiction) in implementation technology independent manner. In declarative business rule statements of SBVR standard, process rules are expressed implicitly. The overall SBVR rules are classified to structural and operative rules. The structural rules express necessities; the operative rules serve for obligations of business behaviour.

When going to information system requirement (i.e. DIM) or design (i.e. PIM) models, both types of semantic rules may be transformed to integrity rules, production rules, or variations of event-condition-action rules; additionally, derivation and transformation rules may be distinguished [40, 41]. We have added object-oriented rules to this classification (Figure 1). Rules may have pre-condition, action, post-condition, left-hand side (LHS), right-hand side (RHS) and other expressions as their components. These components are expressible using UML and OCL constructs.

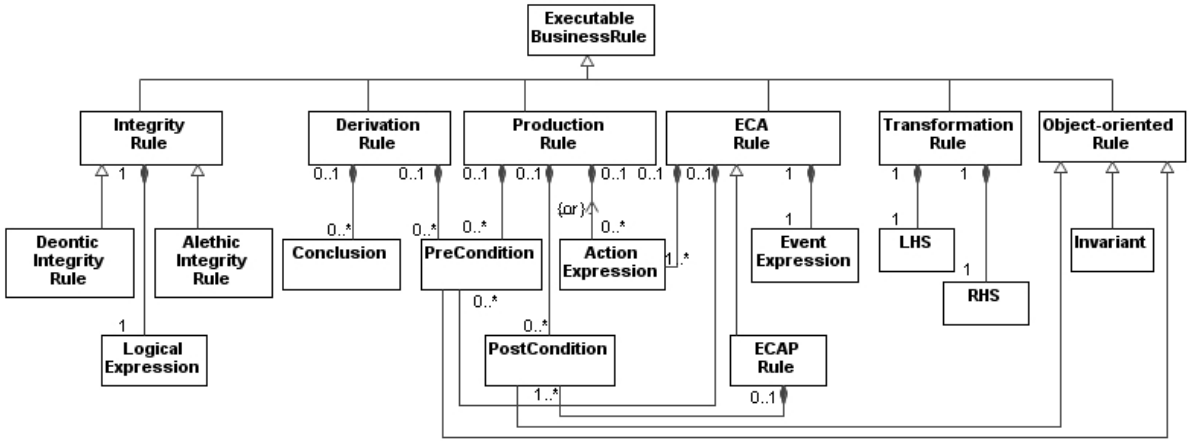


Figure 1. Classification of executable business rules (adapted from RuleML, R2ML [40, 41])

Main UML construct for business rule representation is `Constraint`. It has `context`, `constrainedElement`, and `Value Specification` [30]. Value specification may be provided as `OpaqueExpression` using any language, un-interpretable in UML. Business rules are represented using logical expressions or sentences such as disjunction, conjunction, negation (strong negation or negation as failure), implication, bi-conditional comparisons (>, <, ≤, ≥ , …), quantified sentences (existential and universal), user defined predicates (i.e. functions or relations) etc.

None of these rules directly represent process rules. For investigating the variety of process rules it is purposeful to examine workflow patterns that W. M. P. van der Aalst et all [1] have proposed for evaluating the expressive power of business process modelling languages. In these patterns event rules and constraint rules are coupled together. Event-condition-action rules allow more flexible representation of business processes. A simple ECA rule may be represented by template:

`On <event> if <condition> then <action>.`

Unlike `if-then` constructs in programming languages, ECA rules do not allow `if-then-else` statements but the action part of an ECA rule can represent operation invocation, branching or looping constructs that ensure the required flexibility. However, such a language is not capable to express all workflow patterns, for example, synchronizing merge [1]. The Xchange language [4] allows

representing complex events and may express all of workflow patterns, but it cannot separate business processes from business constraints because constraint rules also are coupled with event rules. T. Graml et all [15] propose to externalize decisions, data constraints and activity compositions as business rules but their approach also is incomplete.

Unlike the majority of approaches, we propose to use UML state machines for business process modelling. There are many reasons for this: state machines are best suited to represent behavioural semantics of object-oriented and service-oriented systems [2, 3]; they are more rigorous than intuitive activity diagrams and more expressive than Pi-calculus [44]; state machines are often clearer and more compact. UML state machines are extension of Harel statecharts that have expanded the Mealy and Moore state machines (in Mealy state machines actions are performed in transitions, in Moore, conversely, actions are performed in states). Harel allowed actions in both states and transitions, and enhanced the previously flat models with nested states and concurrent states. The UML 2.1.2 state machines have the further improvements in respect with previous UML specifications based on Harel statecharts. The UML 2.1.2 state machines [30] are very similar to activity diagrams (it is possible to consider activity diagrams as a special case of UML state machines). Similarly as ECA rule actions, effects of state machine transitions may represent any kind of behaviour – state machine, activity, interaction or opaque behaviour; in such a way, state machines are flexible enough to express any kind of behaviour. However, for separation of event and business constraint concerns, an appropriate methodology is needed.

For processing real life tasks, ECA rules should allow to represent complex events. The problem of representing complex events may be solved by event derivation rules. For modelling services, we have distinguished between atomic event types `SendRequestEvent`, `SendResponseEvent`, `ReceiveRequestEvent` and `ReceiveResponseEvent` [7] that may correspond to UML `SendOperationEvent` or `SendSignalEvent`; or to `ReceiveOperationEvent` and `ReceiveSignalEvent`. For directly representing ECA rules by UML state machine diagrams we use `OperationCallEvent` and execution events (`ExecutionStartEvent` and `ExecutionFinishEvent`). For brevity, we will unite where appropriate the `ReceiveOperationEvent` with `OperactionCallEvent` on state transitions and mark them with stereotype `<<call>>`. `ExecutionStartEvent` and `ExecutionFinishEvent` will be marked as `<<start>>` and `<<finish>>` events. They are explained on the sequence diagram in Figure 2.

Further, in EMDA we differentiate between UML state machines of entities and protocol state machines of interfaces. In state machines of entities, states are pervasive states stored in database; they cannot have events, "`do`" activities, and internal transitions. In protocol state machines of interfaces, conversely, states mean activity states. Entering to a state by default coincides with calling an operation triggered by event on transition. Execution starts when pre-conditions of that operation are satisfied. Effects on transitions of UML Protocol State Machines are replaced by post-conditions of operations called by events on the corresponding transitions.

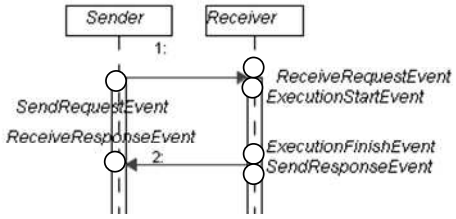


Figure 2. Atomic event types

We will distinguish between event rules where event and action parts of ECA–like rules may express only atomic or complex events without any constraints; and constraint rules where the "`on`" part of ECA rule may express only a single atomic event, the condition part may express a constraint on that event and the action part corresponds to an atomic or a complex event reacting to that simple event. Such rules will look like:

`<<eventRule>>: on <Event1> then <Event2>;`

`<<constraintRule>>: on <SimpleEvent> if <Condition> then <Event>.`

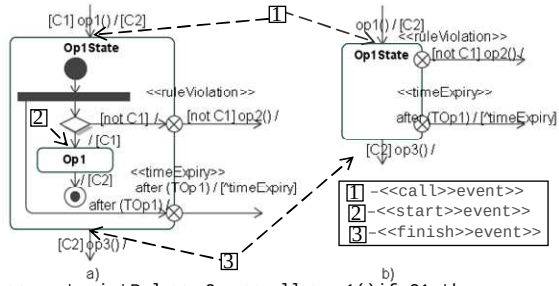
Here `<Event>` generalizes complex and atomic events. `<Event2>` can represent `<<call>>` event, if `<Event1>` is `<<finish>>` event; `<Event>` can denote `<<start>>` event, if `<SimpleEvent>` is `<<call>>` event etc. In such a way it is possible to write constraint and event rules separately.

4. Representation of business rules in UML state machines

In EMDA, complex event types are represented by UML state machine diagrams. As was mentioned, for investigating the variety of process rules we will examine workflow patterns [1]. From the main workflow patterns, we distinguish sequence, parallel split, merge, choice and multiple instances event patterns (synchronization, discriminator, multiple merge, exclusive choice, multiple choice etc may be considered as specializations of these main patterns). Besides, there are other event patterns that should be handled: transactions where sequences of messages should be sent and received between start and finish of complex events; time events; exception events, and correlation between events.

Protocol state machines will model the main successful transitions, when pre-conditions of operations are satisfied; the alternative transitions

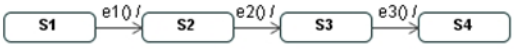
may be caused by violation of pre-conditions or different faults – time expires, exceptions, etc (Figure 3). All these transitions are specified by separate event and constraint rules.




```
<<constraintRule>> On <<call>>op1()if C1 then
     <<start>>op1()
<<constraintRule>> On <<call>>op1()if not C1 then
     <<start>>op2()
<<constraintRule>> On <<call>> op3() if C2 then
     <<start>> op3()
<<eventRule>> On <<finish>>op1 then <<call>>op3()
<<eventRule>> On<<start>>op1() and after(TOp1)then
     ^timeExpiry()
```

Figure 3. Alternative transitions in a full (a) and the abbreviated view (b); here `after(TOp1()` denotes a relative time event , `^timeExpiry()` – a message

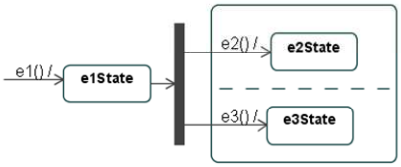
Event sequence is represented by transitions with trigger events without pre-conditions for both state machines of entities and state machines of interfaces (Figure 4).




```
<<eventRule>> On <<finish>>e1()
   then <<start>>e2()
<<eventRule>> On <<finish>>e2() then
   <<start>>e3()
```

Figure 4. Event sequence

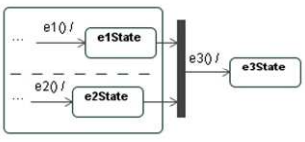
Parallel split is represented by using composite state with regions (Figure 5). Note that here `<<startEvent>>` can mean `<<sendRequest>>` or `<<sendResponse>>` events.




```
<<eventRule>> On <<finish>> e1()
  then <<start>> e1() and <<start>> e2()
```

Figure 5. Parallel split

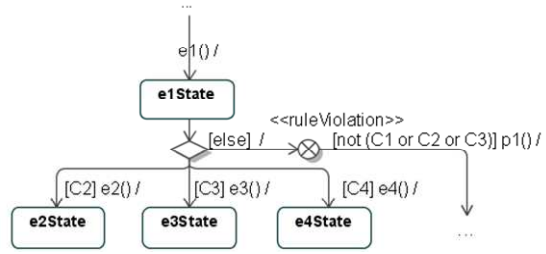
Similarly, the *Synchronizing merge* may be represented by using join pseudostate (Figure 6).




```
<<EventRule>> On <<finish>> e1() and
   <<finish>> e2() then <<start>> e3()
```

Figure 6. Synchronizing merge

Exclusive choice is represented by using choice pseudostate (Figure 7). Note that `ExitPoint` pseudostate with stereotype `<<ruleViolation>>` may be used for representation of `else` condition in the case when neither of exclusive conditions is satisfied. In a similar way, exception and time events (`<<exception>>`, `<<relativeTimeEvent>>` and `<<absoluteTimeEvent>>`) inherent for service execution states may be represented.




```
<<constraintRule>> On <<finish>>e1() if not
   (C2 or C3 or C4) then <<ruleViolation>> p1()
<<eventRule>> On <<finish>>e1() then
   <<call>>e2() or <<call>>e3() or <<call>>e4()
<<constraintRule>> On <<call>>e2() if C2
   then <<start>>e2()
<<constraintRule>> On <<call>>e3() if C3
   then <<start>>e3()
<<constraintRule>> On <<call>>e4() if C4 then
   <<start>>e4()
```

Figure 7. Exclusive choice

Constraint rules and "`then`" part of event rules may be handled by State Coordinator [8]. However, for handling "`on`" part of complex event rules, State Coordinator architecture should be supported with more enhanced capabilities to handle complex event patterns.

5. SBVR, TBL and OCL Representation of Event and Constraint Rules in the Context of EMDA Process

Now we should consider how to define event and constraint rules using SBVR specifications. Events are not explicitly represented according to SBVR standard. Though it is possible somehow to adjust SBVR specifications for representing events, it is not clear, should SBVR rules represent processes or not? There are other questions concerning specification of "real" business rules. For example, business rule for giving a loan may be specified as a single rule (Figure 8).

```
It is obligatory that a bank gives a loan
  if the debtor is the owner of the bail
  or the bail has a consent of the sponsor,
  who is the owner of the bail and
  the initial date of the consent is not
  greater than the issue date of the loan
  and the end date of the consent is not less
  than the planned return date of the loan and
  each loan of the debtor is returned loan and
  the return date of the loan is not greater
  than the planned return date of the loan.
```

Figure 8. SBVR rule for giving a loan (`noun concepts` correspond to entity types, roles and attributes in UML class diagram; *`verb concepts`* – to associations)

Such rules could be implemented as integrity constraints enforced by functionality of databases or

software components. In real life, business processes exist, whose activities are performed by different roles of persons. These processes require different specifications of business rules. For example, a part of the process of giving a loan may be executed by sequential or parallel actions in Figure 9 (a) and (b).

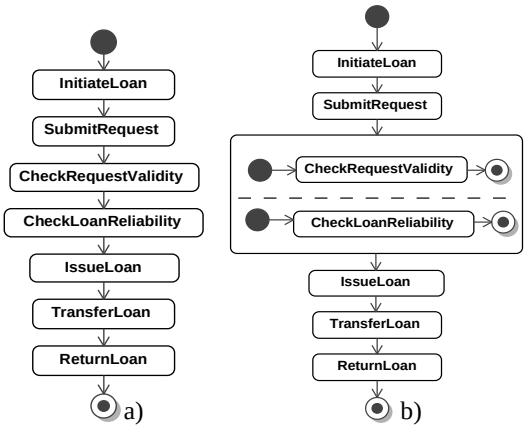


Figure 9. Business process for a loan when actions are performed in a sequence (a) and in parallel (b)

So the rule, presented in Figure 8, should be split in four rule sets that are given in Figures 10–12: the `RuleSet1` defines the obligation of the bank to give a loan for each person who gives a request if the requested loan is the valid and reliable; the `RuleSet2` defines what loan is valid, and the `RuleSet3` – what loan is reliable. The `RuleSet4` defines the reliable person. This rule is the requisite for deriving the reliable loan. We address these rules as "rule sets" because SBVR business rules have associated structural rules (definitions), also supporting fact types, related fact types, synonyms etc. omitted here for brevity.

```
<RuleSet1> It is obligatory
  that the bank issues a loan
  if the loan is the valid loan
  and the reliable loan.
```

Figure 10. The operative rule for a bank to issuing the loan

```
<RuleSet2> It is necessary that
 a loan is the valid loan
 if the debtor is the owner of the bail
 or the bail has a consent of the sponsor
 who is the owner of the bail and
 the initial date of the consent is
 not greater than the issue date
 of the loan and the end date
 of the consent is not less than
 the request date of the loan.
```

Figure 11. The structural rule that defines a valid loan

```
<RuleSet3> It is necessary that
  a loan is the reliable loan
  if the person is the reliable debtor.
<RuleSet4> It is necessary that
  the person is the reliable debtor if
  each loan of the person is the returned loan
   and the actual return date of the loan is not
   greater than planned return date of the loan.
```

Figure 12. The structural rules that define the reliable loan

The example of definitions and supporting fact types are presented in Figures 13–14.

```
It is necessary that the loan has exactly
   one debtor.
It is necessary that the loan has exactly
   one request date.
It is necessary that the loan  has exactly
  one amount.
```

Figure 13. The example of structural rules (definitions)

```
Loan has debtor      Loan has request date
Loan has bail        Loan has amount
The noun concept "loan request" is a role
that ranges over noun concept "loan".
```

Figure 14. The example of supporting fact types

According to EMDA, use cases for the implementation of the loan service (Figure 15) are mapped to interfaces comprising sets of abstract operations (events). Use case specifications are written using business rules and vocabulary terms representing entities, roles, attributes and states of the conceptual model. In the next step, sequence diagrams are created representing interactions between interfaces (the main scenario is presented in Figure 16). State machines of entities (e.g. in Figure 17) and protocol state machines of interfaces (e.g. in Figure 18) serve for reconciliation of various scenarios from several sequence diagrams.

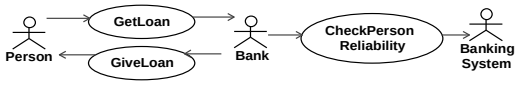


Figure 15. Use Cases of the Loan Service system

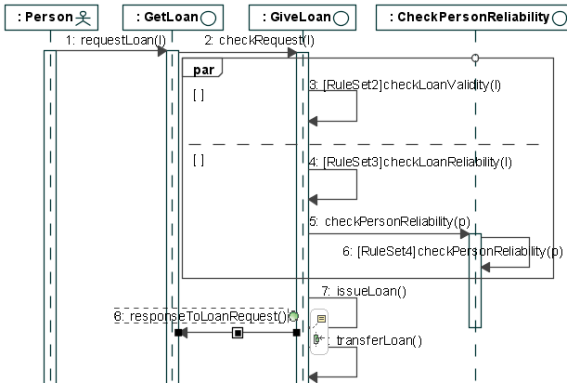


Figure 16. The sequence diagram for giving a loan

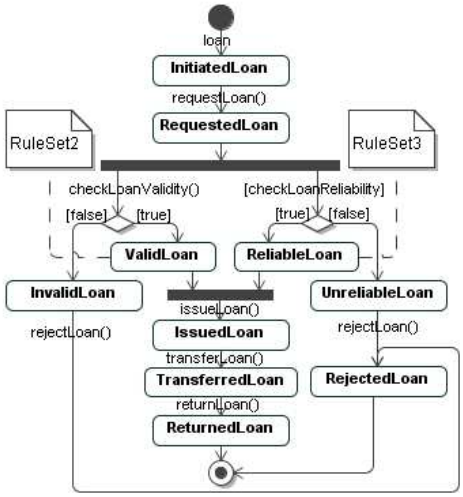


Figure 17. State machine of the `Loan` entity

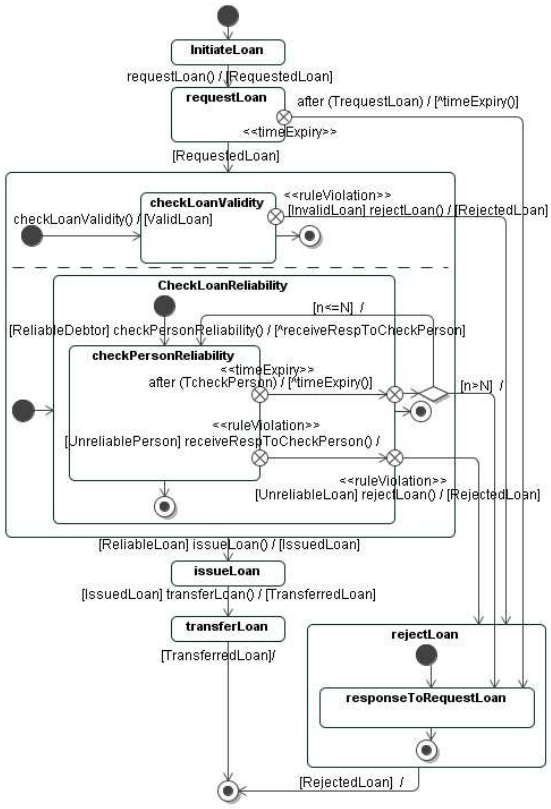


Figure 18. Protocol state machine for loan service process

All business constraint rules in process models are expressed through pervasive states of business entities. These states are defined as state invariants. Definitions of state invariants can change while state concepts remain permanent. When business processes change, new states can be added and existing states can be removed. Note that all terms in models have qualified names, which are not shown in diagrams, relating them with the corresponding context.

In Figure 17, `RuleSet2` and `RuleSet3` denote rule sets whose fragments are depicted in Figures 11–12. These rule sets may be expressed in SBVR, OCL or other formal or informal rule languages. In EMDA, the final specification of service system (Figure 19) uses OCL. However, for the purpose of facilitating easier coping with Business Rule approach, we have created a Template Based Language (TBL) that allows entering of business rules in a simpler but strong enough form based on First Order Logic. An example of TBL expression is presented in Figure 20. The same expression in OCL is presented in Figure 21, and the example of event rule (`RuleSet1`) in OCL is presented in Figure 22.

The first trial version of Template Based Language was implemented in plug-in of CASE tool MagicDraw UML for input of class invariants into UML class diagrams [28]. Currently this project is extended for enabling the input of TBL rules into behavioural models – state machines, sequence and activity diagrams.

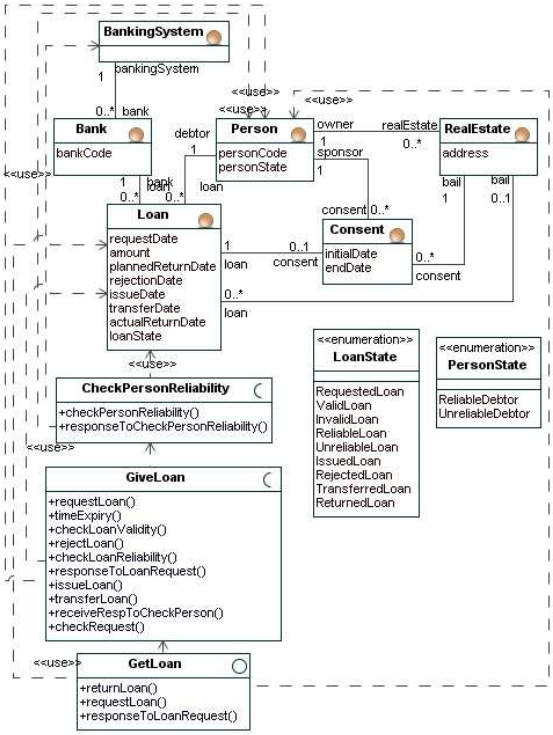


Figure 19. Design Independent Model obtained from business rules of the Loan domain

```
context: GiveLoan::checkLoanValidity()
 pre:([Loan.state]=[RequestedLoan]) and
  ([Loan.debtor]=[Loan.bail.owner]) or
   {([Loan.consent.sponsor]=
    [Loan.consent.bail.owner]) and
   ([Loan.consent.initialDate]<= ([Loan.requestDate])
  and([Loan.consent.endDate<=Loan.plannedReturnDate])}
 post:[Loan.state]=[ValidLoan]
```

Figure 20. Example of constraint rule, obtained from `RuleSet2`, represented in TBL

```
Context:GiveLoan::checkLoanValidity(l:Loan):Boolean
  pre: (if l.debtor→notEmpty() then
    l.debtor=l.bail.owner
     else if l.consent.sponsor→notEmpty() then
      l.consent.sponsor=l.concent.bail.owner
       and l.concent.initialDate<=l.requestDate
      and l.concent.endDate>=l.plannedReturnDate
     else false endIf endIf
  post: l.oclInState(ValidRequest)and result=true
```

Figure 21. Example of constraint rule `RuleSet2` in OCL

```
Context: GiveLoan::checkRequest(l:Loan):Boolean
 post: let message1:oclMessage=
 self.^checkRequestValidity(l:Loan),
 message2:oclMessage=
 self.^checkLoanReliability(l:Loan),
 in if message1.hasReturned() and
 message1.result=true
 and message2.hasReturned() and
 message2.result=true
 then self^issueLoan(l:Loan)else false endif
```

Figure 22. Example of event rule (RuleSet1) in OCL for issuing a loan (this rule defines the obligation of issuing a loan if loan was checked and request is valid and loan is reliable)

TBL rules can be related to classes, properties, operations, states, transitions, activity flows, decision points, sequence diagram messages and interaction fragments. The structure of TBL expressions is simple, yet powerful through recursion (the similar ExeRule language was

implemented in XML [39]). TBL covers a subset of the First Order Logic; therefore, a mutual translatability exists between TBL and subsets of OCL and SBVR. However, TBL was only an intermediate step devoted for assuring a possibility of separating constraint rules from event rules and applying them in a meaningful way for enhancing the EMDA process. The true enhancement of Model Driven approaches should be based on SBVR standard that currently provides the most complete basis for that purpose.

6. Conclusions and further work

During our ongoing research that is performed according to High Technology Development Program Project "Business Rules Solutions for Information Systems Development (VeTIS)" we have encountered problems that also have an impact on other researchers from the area of Model Driven Engineering, Service Oriented Architecture and Business Rules Approach, namely, with the necessity of separating business process rules from business constraints for making them agile. We have proposed a way of separating constraint rules from event rules governing the business process by defining business processes with UML state machines and specifying business rules in the independent way.

We have created the simplified Template Based Language (TBL) that allows easier input of business rules into UML CASE tool environment. The current prototype of TBL allows using UML model elements, representing business vocabulary concepts, in rule specifications. These rules may be transformed to pre-conditions and post-conditions of service operations that are implemented in State Coordinator Pattern based architecture for service oriented information systems. Presented research fragments allow making some assumptions about the feasibility of such an approach for modelling and implementing service oriented information systems according to Business Rule Approach and Model Driven Development.

UML models supplemented with constraints are suitable for the easier representation, checking and implementing business rules. Proposed separation of event rules and constraint rules is important due to emerging technologies of Complex Event Processing, Event Driven architectures and Event Rule Engines. To our knowledge, the complete solution to modelling complex events and business rules in development of information systems for their implementation using both Business Rule Approach and Event Driven technologies currently is not proposed. We have described sequence, parallel split, merge, choice and other event patterns, and investigated possibilities of representing them using UML state machines, SBVR standard, our own Template Based Language and OCL.

Our very initial prototype of TBL contributed to assuring the weightiness of the proposed modelling approach, but TBL is not the final solution. The future work is directed to implementing a more powerful language on the base of SBVR standard and means for translating this language to OCL and implementation languages.

References

[1] **W. M. P. van der Aalst, A. H. M. ter Hofstede, M. Dumas.** Patterns of Process Modeling. *In: M. Dumas, W. M P. van der Aalst, and A. H. M. ter Hofstede (Eds.), Process-Aware Information Systems: Bridging People and Software through Process Technology, Wiley & Sons*, 2005, 179–203.

[2] **B. Benatalah, M. Dumas, M. C. Fauvet, F. A. Rabhi, Q. Z. Sheng.** Overview of some patterns for architecting and managing composite web services. *ACM SIGecom Exchanges archive*, 2002, *Vol. 3, Issue 3*, 9−16.

[3] **D. Berardi, D. Calvanese, G. De Giacomo, M. Lenzerini, M. Mecella.** A foundational vision of e-services. *In: Proc. CAiSE 2003: Workshop on Web Services, LNCS*, 2003, *Vol. 3095*, 28−40.

[4] **F. Bry, M. Eckert, P. L. Patranjan, I. Romanenko.** Realizing Business Processes with ECA Rules: Benefits, Challenges, Limits. *In: Proceedings of 4th Workshop on Principles and Practise of Semantic Web Reasoning, LNCS*, 2006, *Vol. 4187*, 48–62.

[5] **D. Bugaite, O. Vasilecas.** Events linking with rules – Business system, information system and software system. *In: Information technologies' 2008, Proceedings of the 14th International Conference on Information and Software Technologies, Kaunas, Lithuania*, 2008, 324–333.

[6] **J. Cabot, E. Teniente.** Constraint Support in MDA tools: a Survey. *In: European Conference on Model-Driven Architecture 2006, LNCS*, 2006, *Vol. 4066*, 256−267.

[7] **L. Ceponiene, L. Nemuraite.** Design independent modeling of information systems using UML and OCL. *In: Databases and Information Systems: selected papers from the 6th International Baltic Conference on Databases and Information Systems, Riga, Latvia, June 06-09, 2004, IOS Press, Amsterdam*, 2005, 224–237.

[8] **L. Ceponiene, L. Nemuraite.** Transformation from Requirements to Design for Service Oriented Information Systems. *In: Proc. ADBIS 2005: Advances in Databases and Information Systems, Tallinn, Estonia*, 2005, 164−177.

[9] **D. Costal, C. Gómez, A. Queralt, R. Raventos, R. Teniente.** Improving the definition of general constraints in UML. *Software and systems modeling, January*, 2008, 1–18.

[10] **F. Daniel, G. Pozzi.** An Open ECA Server for Active Applications. *Journal of Database Management, Vol. 19, Issue 4*, 2008, 1–20.

[11] Defining Business Rules ~What Are They Really? *The Business Rules Group, formerly, known as the GUIDE Business Rules Project, Final Report, Revision 1.3*, 2000, 1–77.

[12] **A. V. Deursen, E. Visser, J. Warmer.** Model-Driven Software Evolution: A Research Agenda. *In: D. Tamzalit (Ed.), Proceedings of 1st International Workshop on Model-Driven Software Evolution*

(MoDSE), University of Nantes, France, 2007, 41–49.

[13] **T. van Eijndhoven, M. I. Iacob, M. L. Ponisio**. Achieving Business Process Flexibility with business rules. *In: Enterprise Distributed Computing Conference, 2008, EDOC'08, 12th International IEEE*, 2008, 95–104.

[14] **S. Goedertier, J. Vanthienen.** Declarative Process Modeling with Business Vocabulary and Business Rules. *In: On the Move to Meaningful Internet Systems 2007, OTM 2007 Workshops, LNCS*, 2007, *Vol. 4805*, 603–612.

[15] **T. Graml, R. Bracht, M. Spies.** Patterns of Business Rules to Enable Agile Business Processes. *In: Proceedings of the 11th IEEE International Enterprise Distributed Object Computing Conference, IEEE Computer Society*, 2007, 365–375.

[16] **S. Gudas, T. Skersys, A. Lopata.** Approach to Enterprise Modelling for Information Systems Engineering. *Informatica*. 2005, *Vol.16, No.2*, 175–192.

[17] **S. Gudas, T. Skersys, A. Lopata.** Framework for knowledge–based IS engineering. *In: T. Yakhno (Ed.), Proceedings of third international conference „Advances in Information systems (ADVIS2004)", Izmir, Turkey, October 20–22, 2004, LNCS*, 2004, *Vol. 3261*, 512–522.

[18] **J. D. Haan.** 8 Reasons Why Model-Driven Approaches (will) Fail. *InfoQ*, 2008, http://www.infoq.com.

[19] **K. Kapocius, R. Butleris.** Repository for business rules based IS requirements. *Informatica*, 2006, *Vol. 17, No. 4*, 503–518.

[20] **S. Kent**. Model Driven Engineering. *In: Proceedings of the Third International Conference on Integrated Formal Methods, LNCS*, 2002, *Vol. 2335*, 286–298.

[21] **J. Kobielus.** Complex event processing: still on the launch pad. *Network World*, 2007, http://www.computerworld.com.au/.

[22] **M. H. Linehan.** SBVR Use Cases. *In: Rule Representation, Interchange and Reasoning on the Web, Proceedings of the International Symposium, RuleML 2008, Orlando, FL, USA, October 30-31, LNCS*, 2008, *Vol. 5321*, 182–196.

[23] **M. H. Linehan.** Semantics in Model-driven Business Design. *In: 2nd International Semantic Web Policy Workshop (SWPW'06), Athens, GA, USA*, 2006, 1–8.

[24] **P. Loucopoulos, W. M. N. W. Kadir.** BROOD: Business Rules-driven Object Oriented Design. *Journal of Database Management, Vol. 19, Issue 1*, 2008, 41–73.

[25] **S. Lukichev, G. Wagner** Visual Rules Modeling. *In: Perspectives of System Informatics, LNCS*, 2007, *Vol. 4378*, 467–473.

[26] **A. Michlmayr, F. Rosenberg, P. Leitner, S. Dustdar.** Advanced Event Processing and Notifications in Service Runtime Environments. *In: DEBS '08, July 1-4, 2008, Rome, Italy*, 2008, 1–11.

[27] **L. Motiejunas, R. Butleris,** Business rules manipulation model. *Information technology and control*, 2007, *Vol. 36, No. 3*, 295−301.

[28] **L. Nemuraite, L. Ceponiene, G. Vedrickas.** Representation of Business Rules in UML&OCL Models for Developing Information Systems. *In: J. Stirna, A. Persson (Eds.), The Practice of Enterprise Modeling. Proceedings of First IFIP WG 8.1 Working Conference, PoEM 2008, Stockholm, Sweden, November 12-13, 2008, LNBIP*, 2008, *Vol. 15*, 182–196.

[29] Object Constraint Language OMG Available Specification, Version 2.0. *OMG document formal/06-05-01*, 2006, http://www.omg.org.

[30] OMG Unified Modeling Language (OMG UML) Superstructure, V2.1.2. *OMG Available Specification formal/2007-11-02*, 2007, http://www.omg.org.

[31] **E. Pakalnickienė, L. Nemuraite, B. Paradauskas.** The orderliness and precision in conceptual modelling. *In: Current Trends in Informatics. Vol. A. PCI'2007: 11th Panhellenic Conference in Informatics, 18-20 May, 2007, Patras, Greece, Athens: New Technologies Publications*, 2007, 341–350.

[32] **E. Pakalnickiene, L. Nemuraite.** Checking of conceptual models with integrity constraints. *Information technology and control*, 2007, *Vol. 36, No 3*, 285–294.

[33] Production Rule Representation. *Submission to Business Modeling and Integration Domain Taskforce. Fair Isaac Corporation, ILOG SA*, 2007.

[34] **A. Raj, T. V. Prabhakar, S. Hendryx.** Transformation of SBVR business design to UML models. *In: ISEC '08: Proceedings of the 1st conference on India software engineering conference, ACM, Hyderabad, India*, 2008, 29–38.

[35] **R. G. Ross.** Principles of the Business Rules Approach. *Addison-Wesley, Boston*, 2003.

[36] **R. G. Ross.** The Business Rule Book: Classifying, Defining an Modeling Rules. *Business Rule Solutions, Houston*, 1997.

[37] **M. Schacher.** Business Rules from an SBVR and an xUML Perspective (Parts 1–3). *Business Rules Journal*, 2006, *Vol. 7, No. 6–8.*

[38] Semantics of Business Vocabulary and Business Rules (SBVR), v1.0. *OMG Available Specification, OMG Document No. formal/2008-01-02*, 2008.

[39] **G. Vedrickas, L. Nemuraite,** Achieving business flexibility by empowering business component system with business rules technology: Executable rules. *In: Vasilecas, O., Eder, J., Caplinskas, A. (Eds.), Databases and Information Systems: Seventh International Baltic Conference on Databases and Information Systems. Communications, Materials of Doctoral Consortium, Vilnius, Lithuania, 3–6 July, 2006, Technika, Vilnius*, 2006, 193–158.

[40] **G. Wagner, A. Giurca, S. Lukichev,** A Usable Interchange Format for Rich Syntax Rules. Integrating OCL, RuleML and SWRL. *In: Proceedings of Reasoning on the Web, WWW Workshop, Edinburgh, Scotland*, 2006.

[41] **G. Wagner, S. Tabet, H. Boley,** MOF-RuleML: The Abstract Syntax of RuleML as a MOF Model, 2004, http://www.ruleml.org.

[42] **Wahler, M., Ackerman, L., Schneider, S.** Using IBM Constraint Patterns and Consistency Analysis. *IBM Developer Works, May*, 2008.

[43] **M. Wahler, J. Koehler, A. D. Brucker.** Model-driven constraint engineering (2006). *In: MoDELS Workshop on OCL for Meta-Models in Multiple Application Domains, Electronic Communications of the EASST, Vol. 5, 2006, Technische Universität Dresden, Germany*, 2006, 1–15.

[44] **G. Xue, J. Lu, S. Yao.** Investigating Workflow Patterns in Term of Pi-calculus. *In: Proceedings of CSCWD, IEEE Computer Society*, 2007, 823−827.

DOI: 10.5755/j01.itc.38.1.11920