Representation of Business Rules in UML&OCL Models for Developing Information Systems

Lina Nemuraite, Lina Ceponiene, and Gediminas Vedrickas

Kaunas University of Technology, Studentu 50-308, LT 51368 Kaunas, Lithuania
{Lina.Nemuraite,Lina.Ceponiene}@ktu.lt, G.Vedrickas@erp.eu

Abstract. Currently Business Rules Approach has attained the great interest in business and software development communities. According to SBVR standard, business rules are the rules under business jurisdiction. Consequently, business rules should be managed by business people. However, acquiring and supporting the correct and reconciled SBVR style rule sets is the difficult problem. In practice, business rules often are managed by software developers and system analysts; more straightforward and safer processes for capturing and modifying business rules are related with visual business process models. In this paper, possibilities of representing business rules by UML&OCL models and their applicability in modern development processes are investigated[1].

Keywords: Business rule, UML, OCL, stereotype, constraint, invariant, event, action, condition.

1 Introduction

Nowadays every complex and long-term information system has to deal with business rules (BR). Business process management, Service Oriented Architecture (SOA), Web Services, Component-based development, Semantic Web and ordinary databases – some day everyone working in these areas will confront with necessity to separate business rules from the rest of the considered domain. Currently business rules-based IS development methods are evolving rapidly; nevertheless the business rules are implemented mainly in the so called business rules engines that are suitable for specific type of business rules (i.e. production rules). Business rules engines are appearing everywhere as a standalone tools (Blaze Advisor, ILOG Rules, Haley systems etc.) as well as components of the middleware, accompanying databases (Microsoft, IBM, Oracle), ERP tools (SAP) etc.

Regardless of these facts, Platform-Independent business rules can be specified in a very limited number of CASE tools (mostly, using informal natural language sentences or OCL); some CASE tools provide limited rule checking functionality; none of the CASE tools provide a full set of business rules management functions.

Business rules techniques are not widely used in the methods of Model Driven Development (MDD); the existing solutions are still of experimental nature. Recently

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

J. Stirna and A. Persson (Eds.): PoEM 2008, LNBIP 15, pp. 182–196, 2008.

OMG has issued Semantics of Business Vocabulary and Business Rules (SBVR) [26] for documenting and the interchange of business vocabularies and business rules among organizations and software tools. SBVR business rules are based on predicate logic with a small extension in modal logic. They are applicable for linguistic processing and are designed for business people and business purposes, independent of information systems designs.

The core idea of the Business Rules Approach is to take semantic business rule representations and convert them to information system designs. However, this idea is still far from a reality. Business rules in the inputs of business rule engines have the shape of platform-specific representations, i.e. they are very far from SBVR rules. For the present, business rules are entered into IS design process in various phases, mostly – during implementation. This situation may change soon, as OMG already has issued the Production Rule Representation (PRR) standard for platform-independent production rules; RuleML initiative and REWERSE group are working on interchangeable specifications for other kinds of rules, devoted for Semantic Web and Object-Oriented systems; there even are proposals for transformation of SBVR specifications to UML models [11, 21]. Obviously, in the near future we will be able to manage the overall path from business design to implementing software code, inter-relate them and rapidly reflect business changes in supporting information technologies.

While platform-specific representation of business rules is essentially incompatible with requirement of being "designed for business people", semantic SBVR style representation is none the less difficult. Processes for specifying well-formed, reconciled sets of pure declarative business rules, advocated by Ross [22, 23], conformable to the problem domain, are not defined yet. Business rules are better understood and reconciled when they are captured during business modelling by the use of visual modelling tools. Therefore we are discussing the possibilities of using UML diagrams supplemented with Object Constraint Language (OCL) for expressing different types of rules defined in [5, 28, 30] or elsewhere. In UML metamodel, rules can be expressed in OCL or any other language as `OpaqueExpressions`, representing `ValueSpecifications` of constraints, parameters and other elements of UML models. As single UML constraints are not able to describe all known types of business rules, they are suitable to do this together with other UML elements (objects, operations, actions, events etc).

The rest of the paper is organized as follows. In section 2 the related work is analysed. Section 3 presents the life cycle of information systems augmented with business rules solutions. In section 4, applicability of UML&OCL models for representation of business rules is analysed. Sections 5 and 6 shortly present UML constraint extensions and Template-Based Language for representing business rules. Finally, section 9 draws some conclusions and highlights the future work.

2 Related Work

Addressing to the numerous related work, we have raised two questions. The first of them – *how to structure expressions representing business rules*? Business rules are becoming more and more important in information systems. Ontology developers, Business rules community, UML modellers are developing different business rules

languages and tools. As business rules are independent from their representation, different business rules systems should be able to communicate and business rules in any language should adhere to a single, unified abstract syntax (metamodel). In SBVR [26] and Ontology Definition Metamodel (ODM) [18] standards such syntax is the Common Logics (CL).

Common Logic (CL) is essentially different from other rule metamodels since it is rather the dialect of first order logic than a modelling language. First order logic is the foundation of knowledge representation models, including relational databases. CL is a modern form of first order logic, oriented to new application domains. For example, CL can represent irregular sentences, required for SBVR modal expressions.

If we look at business rule metamodels related with different business rule modelling languages we would see the very similar but fairly inconsistent pictures. RuleML [30], designed for interchange between different languages and types of rules, is based on the construct common for all types of rules – logical sentence (or logical formula). In SBVR, such construct is called as logical formulation [26], in ODM – logical sentence [18]. The structure of these constructs is slightly different. Such diversities sometimes raise the need to develop own business rule metamodels, similar to OMG and other standards but having own specific properties (e.g. [27]).

OMG has chosen CL over OCL as business rules representation language, because, first, SBVR vocabularies are using natural, but not object-oriented language; secondly, OCL still lacks inference essential for ontological models. OCL is translatable to first order logics [1, 2] and quite capable to represent PIM business rules. As SBVR and ODM standards should be interoperable with other OMG standards it is purposeful to consider them during develment of methodology for modelling PIM business rules. Here comes the REWERSE Rule Markup Language [14, 15, 27] – a format for rule interchange between RuleML, Semantic Web Rule Language (SWRL) and OCL. These languages have rich syntax for representing rules, i.e. they are supporting typed atoms and terms that is unattainable in standard predicate logic. R2ML format retains constructs of different languages by reconstructing them to logical formulas that are suited to access differences between OCL and ontological languages (OWL and RDF) for ensuring lossless transformations. R2ML encompasses integrity, derivation, production and reaction rules (also defined in RuleML).

OMG PRR standard [20] is the UML extension designed for object-oriented PIM level production rules, where production rule extends UML metaclass `NamedElement`; as one of possible implementations, PRR specification is considering rule components as OCL expressions, supplemented by imperative expressions to representing actions (namely, invoke, update state, and assign). In general, the upper PRR structure of production rules is similar to proposed by RuleML and R2ML, but PRR is not explicitly considering correspondence to logical expressions or Semantic Web Languages.

Our approach differs from PRR in two essential points: we are relying on direct usage of UML models with OCL expressions for capturing components of business rules; and we propose constraint stereotypes for aligning OCL expressions with logical expressions of business rules.

URML [13] is the interesting proposal of REWERSE group for visual modelling of derivation and production rules; the approach was implemented in UML CASE tool "Strelka". Rules are represented as the first class entities in class models and have

relationship (supplemented with expressions) with concepts involved. Unhappily, this approach differs from practice of expressing business rules in object oriented models and may be inefficient for the large sets of business rules.

The second question – *what process is appropriate for defining "real" business rules*? Current business-driven development, aiming at a tighter alignment of the IT infrastructure with business needs and requirements, centres on business processes when process management software or other artefacts are derived directly from process models [7, 9]. Alternatively, we see the perspective in deriving PIM artefacts from models of interactions and state machines [3]. 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 [25]. MDEE supports the smooth going from SBVR structural and operative rules to PIM Constraints, ECA and CA (Condition-Action) rules (corresponding to production 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. Finally, a very initial prototype, proposed by Linehan [11] for transformation of limited SBVR rules to OCL preconditions, also is related with business process modelling. Obviously, the desired methodologies for defining Ross's or SBVR business rules do not exist yet and they are the subject of the future research.

The simple, but proven BROOD approach recently published in [12] gives the partial answer[2] to both aforementioned questions by proposing 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 tool developed on top of the Generic Modeling Environment (GME) [10]. This approach (though it is not related with OCL) has many common points with our efforts and represents modern trends in Model Driven Engineering (MDE) [6], i.e. development of modeling environments tailored for specific domains and specific development methodologies.

Considering research related with OCL templates it is worth to mention general OCL templates implemented in Rational Software Architect [31, 32]; our own research in developing stereotypes and templates for integrity constraints [16, 19]; the work [4]; attempts to transform OCL to natural language [1].

3 Model Driven Development Process Incorporating Business Rule Approach

In Model Driven Development the biggest part of the work efforts is put into model development and most of the schemas and program code are generated automatically. Naturally, it becomes more and more vital to computerize and automate the very early

[2] BROOD process starts from the analysis phase.

stages of IS development. However, this is not so easy to accomplish. IS developers and business people are still suffering from the absence of a common language that would be understandable for all parties, formal enough to be unambiguous, and not too excessive.

This problem concerns all phases of the IS development life cycle where the collective participation of business people and developers is necessary. Figure 1 shows the vision of the business rule-extended IS development method related with the OMG standards and MDA.

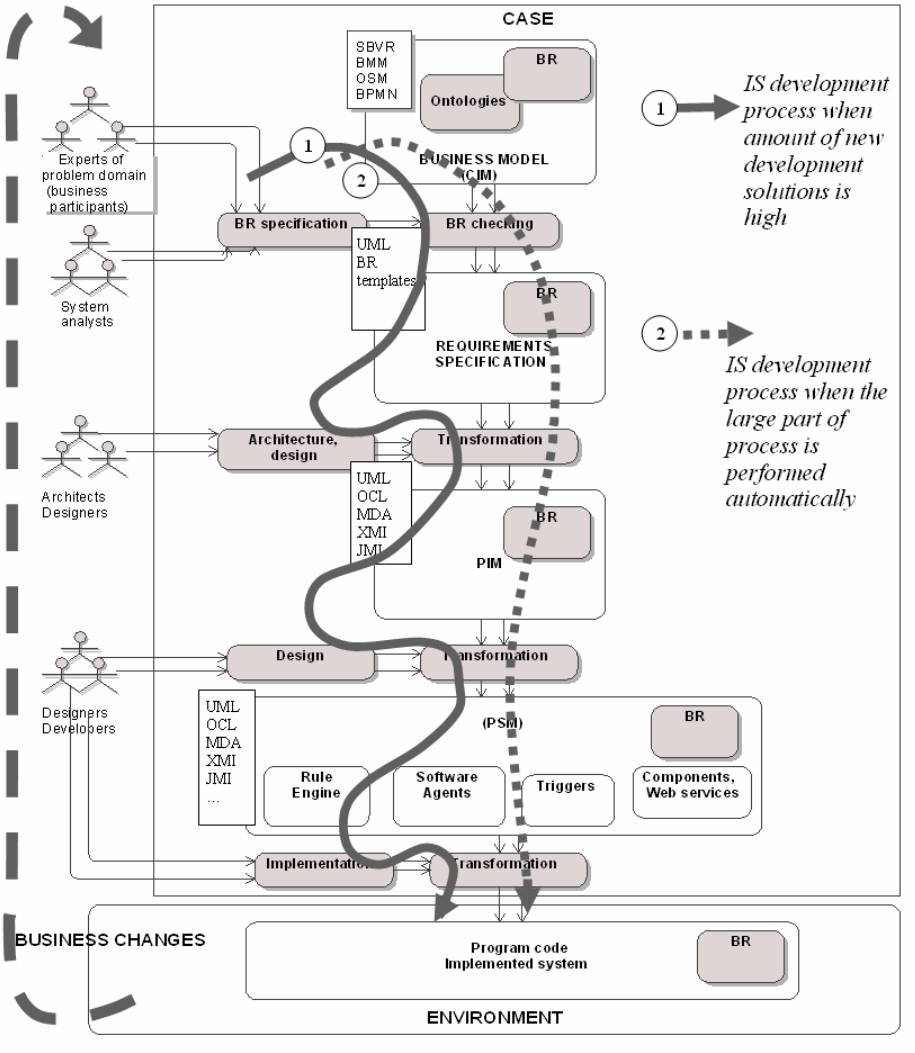

Fig. 1. The life cycle of information systems augmented with business rules solutions

At the beginning of the development process, experts of the problem domain are using the natural language to describe the problem domain. Such descriptions lack precision; they are ambiguous and can be interpreted incorrectly. Later they experience difficulties verifying domain models that were developed by the system analyst. One of the main reasons of this problem are notations (e.g. UML), hardly readable for non-technical people. Business rule specification language could be used as one of the most effective techniques to communicate. This language should be close enough to the natural language and at the same time formal enough to be precise and unambiguous.

Business rules are intensively used during the software development process and should be traceable through the overall IS development life cycle thus linking the program code with problem domain and enabling rapid reflection of changes in business environment to the changes of supporting Information Technology system. Though the ideal Business Rule Approach is to take semantic business rule representations and convert them to information system designs, in many cases these rules would be entered by system analysts during requirements phase. Such rules would have stricter and less end-user friendly form.

The key part of the process is the specification of business rules using natural language based templates, thus preserving their natural form. Such approach allows active participation of the stakeholders in the process of BR specification. The process as well as the ways of the business rules support can have a lot of variations [8, 24, 29]. Figure 1 presents two extreme cases. The first one – when a lot of new design related decisions have to be made and in every stage of the process active participation of analysts, designers and programmers is required. In the second case the main task of IS design is the specification of business rules and requirements, while the subsequent representation of all that in the program code is carried out almost automatically, according to the known, repetitive architectural and implementation solutions. It must be noted that qualified specialists would be required to write the specific code or to set the parameters of the BR templates, but the efficiency of the design process increases dramatically.

Furthermore, developers are relieved of repetitive and uncreative job and the amount of errors decreases. The most important achievement, however, is the strong connection between the business semantics and its implementation. This enables the quicker and more efficient identification of the points influenced by the changes.

4 Applicability of UML&OCL Models for Representation of Business Rules

According RuleML, PRR and R2ML, PIM level business rules may be classified into integrity, derivation, production, event-condition-action (ECA), event-condition-action-postcondition (ECAP) and transformation rules, having condition, action, postcondition, left hand side (LHS), right hand side (RHS) and other expressions as their components (Fig. 2). These components are expressible using UML and OCL constructs. Main UML constructs for business rules representation are *Constrains* that have their context, `constrainedElements` and `ValueSpecifications` [17] (Fig. 6). Value specification may be provided as `OpaqueExpression` using any

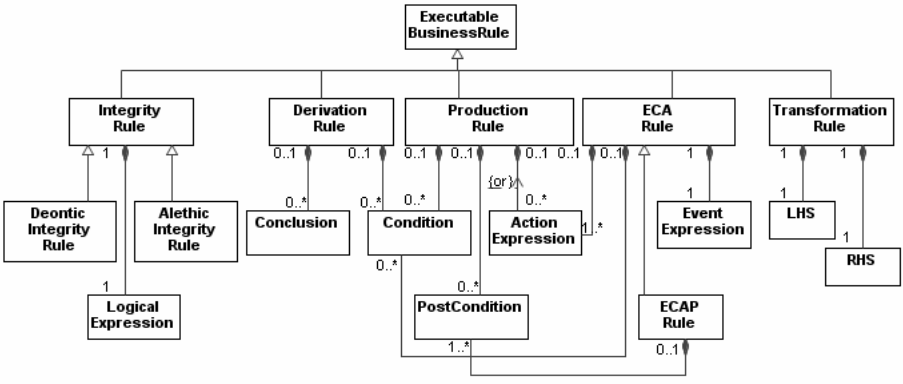

Fig. 2. Classification and composition of executable business rules (adapted from RuleML, R2ML [28, 30])

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), and user defined predicates (i.e. functions or relations).

Integrity rules are represented by OCL invariants. This type of rules is best understandable and supported in several UML CASE tools (e.g. Magic Draw, Rational Software Architect, USE, OCLE etc.). OCL invariants still may represent other types of rules: derivations, transformations and even action expressions (by explicit specification of operations).

State machines are capable to express derivation rules for object states (via state invariants); event and action expressions; conditions and postconditions. Event expressions represent names of called operations, received signals; time and change events and possibly others. For example, `Order` state machine fragment on Figure 3 b is represented by the following OCL expression on Figure 4.

Guards on transitions of state machines should be pure expressions without side effects; they correspond to conditions of PR or ECA rules (Table 1).

Table 1. Correspondence between ECAP rule and UML &OCL model on Figures 3–4

Rule component	UML&OCL element	Example
`Event`		`Order::approve()`
`Condition`	`Source state`[3] `and guard`	`self.customer.oclInState(Registered)` `self.customer.rating>=0,4`
`Action`	`Effect`	`Order::approve()` `Order::verifyCustomer()`
`Postcondition`	`Target state`	`self.oclInState(Approved)`

[3] Condition implied by a source state often is accepted by default as the condition of existence of operation parameters.

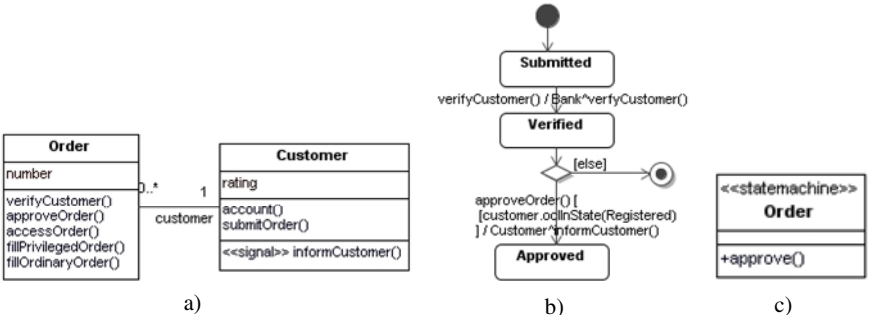

Fig. 3. Fragment of state machine (b) for `Order` (a). Called operations (e.g. `approve()`) may belong to class `Order` (a) or directly to its behaviour (i.e. state machine (c)).

```
context Order::approve():Boolean
pre: self.oclInState(Verified) and
   self.customer.oclInState(Registered) and
      self.customer.rating>=0,4
post: self.oclInState(Approved)
         and self.customer^informCustomer()
```

Fig. 4. OCL constraint for operation `approve()`

Unlike `if-then` constructs in programming languages, production rule languages in BR engines often do not allow `if-then-else` rules, but the action of a PR can represent operation invocation, branching or looping constructs. Similarly, effects of state machine transitions may represent any kind of behavior – state machine, activity, interaction or opaque behavior. Conversely, branching of transitions using decision points and `else` conditions helps avoiding ambiguities and incompleteness, i.e. ensures that for any occurrence of the event only one transition can fire, and an alternative transition exists if event would not occur [17]. As branching transitions may be reconstructed to simple transitions, this factor does not prevent the mapping between state machines and PR, ECA, or ECAP rules. UML protocol state machines directly represent preconditions (subsetting guards) and postconditions (instead of effects) on transitions.

Postconditions often are not needed as postconditions of one rule become the preconditions of another rule(s). Postconditions are important when transition has no event and action. Then we have a derivation rule, i.e. the PR of type `if condition then postcondition`. Postconditions include the target state of the transition (possibly with state invariant). State invariants are derivation rules explicitly stating the meaning of the state.

We can obtain derivation rules from invariants having expressions "`if-then`" and "`implies`". For example, the derivation rule expressing the meaning of `Customer` state `Registered` may be defined as in Figure 5.

However, the implication does not mean a derivation by itself; it may be translated to derivation by making the corresponding methodological solution.

OCL postconditions may include message sending and receiving expressions that in general may represent composite states, activities or interactions. In UML state

machine notation, such cases are represented by additional diagrams referenced by an effect. For representation of messages, interactions or sequence diagrams are more suitable. For example, sequence diagram in Figure 6 may be represented by the following OCL expression on Figure 7.

```
Customer inv inv1:
self.oclInState(Registered) implies self.account()=true
or
Customer inv inv1:
  if self.account()=true then self.oclInState(Registered)
     else false endIf
```

Fig. 5. State invariant as a derivation rule

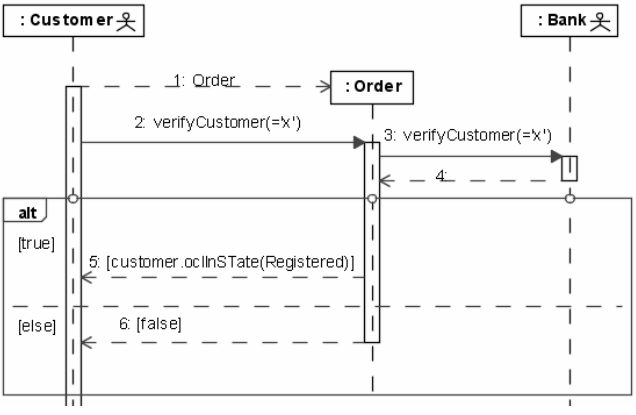

Fig. 6. Sequence diagram for effect `verifyCustomer` (Fig. 3, b)

```
context Order::verifyCustomer(x:Customer):Boolean
post:let message:oclMessage = Bank^verifyCustomer(x:Customer) in
      if message.hasReturned() and message.result=true
      then self.customer.oclInState(Registered) and
      self.customer=x
      else false endIf
```

Fig. 7. OCL expression for operation `VerifyCustomer()` (Fig. 6)

Here we will have ECAP rule with `ElseCondition` (Table 2). Expression `self.customer=x` may be named as "`CreationCondition`"; such expressions are able to represent bodies of updating or creation operations by assigning values of operation parameters to objects and their properties.

Activity diagrams in some aspects are similar to state machines. For example, we can define operations, corresponding to `callOperation` actions from activity diagram fragment on Figure 8, by the OCL expressions presented on Figure 9.

Table 2. Correspondence between ECAP rule and UML &OCL model on Figures 6–7

Rule component	UML&OCL element	Example
`Event`	`message`	`verifyCustomer(x:Customer)`
`Condition`	`guard`	`message.result=true`
`Action`	`message`	`Bank^verifyCustomer(x:Customer)`
`Postcondition`	`guard`	`self.customer.oclInState(Registered)` `self.customer=x`
`ElseCondition`	`guard`	`false`

Fig. 8. Activity diagram fragment

```
context Order::fillPrivilegedOrder():Boolean
pre: self.oclInState(Approved) and
   policy.oclInState(Valid) and
      self.customer.rating>=0,7
context Order::fillOrdinaryOrder():Boolean
pre: self.oclInState(Approved) and
   policy.oclInState(Valid) and
      self.customer.rating<0,7
```

Fig. 9. OCL expression for `CallOperation` actions from Figure 8

UML 2 structured activities (conditional activities, loops, sequences), expansion regions, interaction constraints and other also give many possibilities for representing business rules. Generalizing UML possibilities for representing business rules we can conclude:

- Integrity rules are directly represented in class diagrams as invariants;
- Derivations are indirectly represented in class diagrams as invariants; in state machines and interactions – as state invariants or transitions without events and effects;
- Action rules (PR, ECA, ECAP) are indirectly represented in several ways: in class diagrams as preconditions, postconditions or body conditions of operations; guards and other types of constraints on messages, transitions, control and object flows, interaction fragments, structured activities etc in behavioral diagrams (interactions, state machines and activities).
- For explicit representation of business rules it is purposeful to extend UML models reusing existing UML elements as much as possible. In OMG standards semantics of OCL expressions for representing UML constraints is clearly defined

only for invariants and operation constraints in class diagrams. For other diagrams the semantics of OCL expressions is not defined and should be clarified.

5 UML Constraint Extensions for Representing Business Rules

Performed analysis lets us make an assumption that the following extensions (the generalization set is overlapping and incomplete) would be useful for specification of business rules in UML models (Fig. 10).

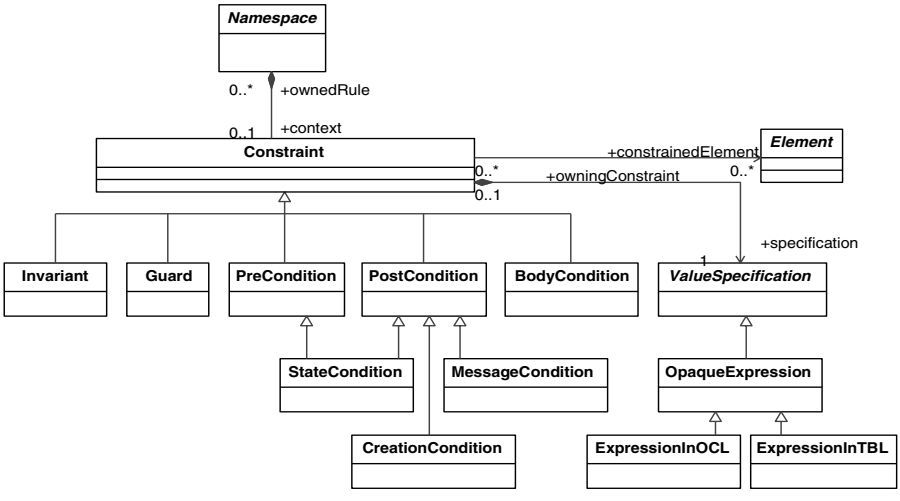

Fig. 10. UML constraint extensions for business rules

Components of business rules represented by UML constraints have value specifications as `OpaqueExpressions` described in OCL (`ExpressionInOCL`) or in any other language, for example, Template Based Language (TBL) as `ExpressionInTBL` (Fig. 10). Example of constraint stereotypes is presented on Figure 11.

```
<<preCondition>><<stateCondition>>: self.oclInState(Submitted)
<<preCondition>><<stateCondition>>self.customer.oclInState(Registered)
<<preCondition>> self.customer.rating>=0,7
<<postCondition>>>><<stateCondition>>: self.oclInState(Approved)
<<postCondition>>>><<messageCondition>>self.customer^verifyCustomer()
```

Fig. 11. Stereotypes for constraints of operation `approve()` (Fig.4)

The greatest obstacle for expansion of business rules approach is the lack of means for supporting the entering of business rules to design models. In the following section we present the prototype of Template Based Language for entering business rules during development of IS.

6 Entering Business Rules Using Template Based Language

Template Based Language currently is a trial language implemented as plug-in of UML CASE tool Magic Draw for input of class invariants into UML class diagrams.

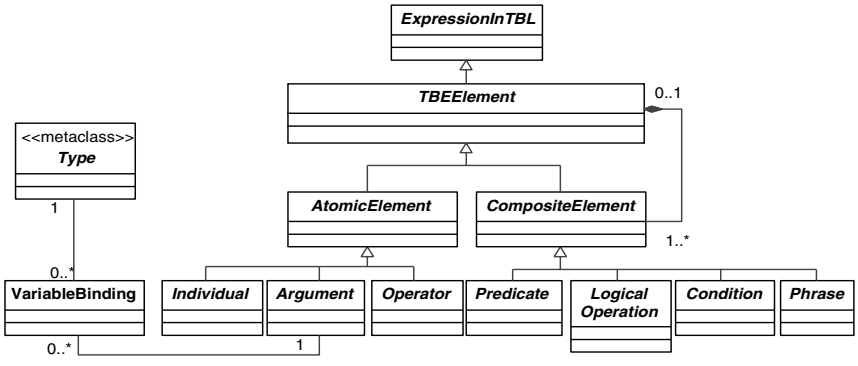

Fig. 12. TBL expressions

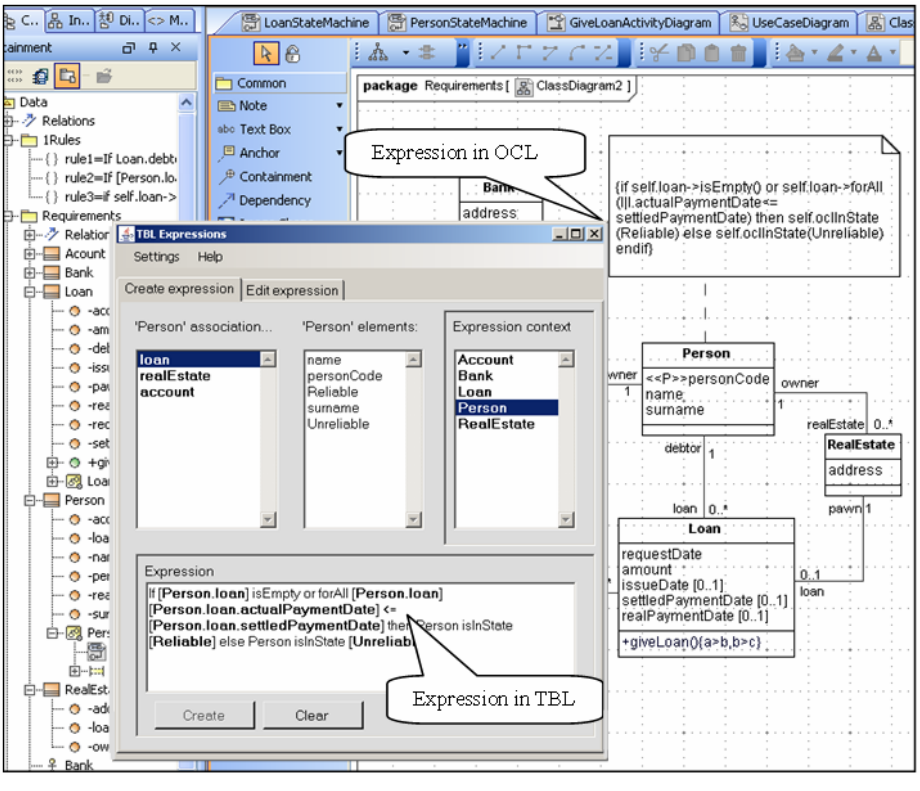

Fig. 13. Interface for entering TBL expressions

Currently this project is under extension for enabling the input of TBL rules into behavioral diagrams – state machines, interactions and activities. TBL rules can be related to classes, properties (attributes, operations and association ends), state machine states and transitions; activity flows and decision points; sequence diagram messages and interaction fragments. Structure of TBL expressions is simple, yet powerful through recursion (Fig. 12) (the similar ExeRule language was implemented in XML [27]). The interface for entering TBL expressions is presented in Figure 13.

Currently TBL expressions are very simple, having a little similarity with natural language. However, they are promising to significantly improve preciseness of IS designs supplementing them with business rules translatable to OCL or rule implementation languages. The most of business rules are simple and mainly require tool support with unambiguous representation and interpretation.

7 Conclusion and Future Work

We have presented the ongoing research that is performed according to High Technology Development Program Project "Business Rules Solutions for Information Systems Development (VeTIS)". In this paper we do not claim to solve all problems related with business rules, or present the overall results of this project. We would rather like to demonstrate our research fragments for discussion about the feasibility and efficiency of such approach.

The raising numbers of research and CASE tool implementations, concerning business rules, natural language and OCL, allow us to hope for the progress in this area. It is also a fact that UML models supplemented with constraints are suitable for the easier representation and implementation of business rules. These direct possibilities of UML may found the broad applicability in practical development of enterprise information systems based on different, specific architectures and development methodologies.

Categorization and structuring UML constraints allows representing them in a limited natural language translatable to OCL and other rule languages. Our very initial prototype for entering template-based constraints is still not the final answer; the future work will extend the TBL for UML behavioral diagrams; translate TBL to OCL, and, of course, investigate transformation of Template Based Expressions to rule implementation languages.

References

1. Ahrendt, W., Baar, T., Beckert, B., Bubel, R., Giese, M., Hähnle, R., Menzel, W., Mostowski, W.: The KeY Tool. Software and Systems Modeling 4(1), 32–54 (2005)
2. Beckert, B., Keller, U., Schmitt, P.H.: Translating the Object Constraint Language into First-order Predicate Logic. In: VERIFY, Workshop at Federated Logic Conferences (FLoC), Copenhagen, Denmark, pp. 1–11 (2002)
3. Ceponiene, L., Nemuraite, L.: 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, pp. 224–237. IOS Press, Amsterdam (2004)

4. Costal, D., Gómez, C., Queralt, A., Raventos, R., Teniente, R.: Improving the definition of general constraints in UML. Software and systems modeling, pp. 1–18 (January 2008)
5. Defining Business Rules ~ What Are They Really? The Business Rules Group, formerly, known as the GUIDE Business Rules Project, Final Report, revision 1.3, pp. 1–77 (July 2000)
6. Deursen, A.V., Visser, E., Warmer, J.: Model-Driven Software Evolution: A Research Agenda. In: Tamzalit, D. (ed.) Proceedings 1st International Workshop on Model-Driven Software Evolution (MoDSE), pp. 41–49. University of Nantes, France (2007)
7. Gudas, S., Skersys, T.: The Enhancement of Class Model Development Using Business Rules. In: Bozanis, P., Houstis, E.N. (eds.) PCI 2005, LNCS, vol. 3746, pp. 480–490. Springer, Heidelberg (2005)
8. Kapocius, K., Butleris, R.: Repository for business rules based IS requirements. Informatica, Vilnius, 17(4), 503–518 (2006)
9. Koehler, J., Hauser, R., Küster, J., Ryndina, K., Vanhatalo, J., Wahler, M.: The Role of Visual Modeling and Model Transformations in Business-driven Development. In: Electronic Notes in Theoretical Computer Science (ENTCS), vol. 211, pp. 5–15. Elsevier Science Publishers, Amsterdam (2008)
10. Ledeczi, A., Maroti, M., Bakay, A., Karsai, G., Garrett, J., Thomason, C., Nordstrom, G., Sprinkle, J., Volgyesi, P.: The Generic Modeling Environment. In: Workshop on Intelligent Signal Processing, Budapest (2001)
11. Linehan, M.H.: Semantics in Model-driven Business Design. In: 2nd International Semantic Web Policy Workshop (SWPW 2006), Athens, GA, USA, pp. 1–8 (2006)
12. Loucopoulos, P., Kadir, W.M.N.W.: BROOD: Business Rules-driven Object Oriented Design. Journal of Database Management 19(1) (2008)
13. Lukichev, S., Wagner, G.: Visual Rules Modeling. In: Virbitskaite, I., Voronkov, A. (eds.) PSI 2006. LNCS, vol. 4378, pp. 467–473. Springer, Heidelberg (2007)
14. Milanović, M., Gašević, D., Giurca, A., Wagner, G., Devedžić, V.: On Interchanging between OWL/SWRL and UML/OCL. In: Proceedings of 6th Workshop on OCL for (Meta-)Models in Multiple Application Domains (OCLApps) at the 9th ACM/IEEE International Conference on Model Driven Engineering Languages and Systems (MoDELS), Genoa, Italy, pp. 81–95 (2006)
15. Milanovic, M., Gasevic, D., Giurca, A., Wagner, G., Devedzic, V.: Sharing OCL Constraints by Using Web Rules. In: Engels, G., Opdyke, B., Schmidt, D.C., Weil, F. (eds.) MODELS 2007. LNCS, vol. 4735, pp. 1–15. Springer, Heidelberg (2007)
16. Miliauskaite, E., Nemuraite, L.: Representation of integrity constraints in conceptual models. Information technology and control 34(4), 355–365 (2005)
17. OMG Unified Modeling Language (OMG UML), Superstructure, V2.1.2. OMG Available Specification formal/2007-11-02 (2007)
18. Ontology Definition Metamodel Specification, OMG Adopted Specification ptc/2007-09-09 (2007)
19. Pakalnickiene, E., Nemuraite, L.: Checking of conceptual models with integrity constraints. Information technology and control 36(3), 285–294 (2007)
20. Production Rule Representation. Submission to Business Modeling and Integration Domain Taskforce. Fair Isaac Corporation, ILOG SA (2007)
21. Raj, A., Prabhakar, T.V., Hendryx, S.: Transformation of SBVR business design to UML models. In: ISEC 2008: Proceedings of the 1st conference on India software engineering conference, pp. 29–38. ACM, Hyderabad (2008)
22. Ross, R.G.: The Business Rule Book: Classifying, Defining an Modeling Rules. Business Rule Solutions, Houston (1997)

23. Ross, R.G.: Principles of the Business Rules Approach. Addison-Wesley, Reading (2003)
24. Ross, R.G., Lam, G.S.W.: The Do's and Don'ts of Expressing Business Rules. Business Rule Solutions, http://www.brsolutions.com/rulespeak_download.shtml
25. Schacher, M.: Business Rules from an SBVR and an xUML Perspective (Parts 1–3). Business Rules Journal 7(6-8) (2006)
26. Semantics of Business Vocabulary and Business Rules (SBVR), v1.0. OMG Available Specification formal/2008-01-02 (2008)
27. Vedrickas, G., Nemuraite, L.: 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, Technika, Vilnius, July 3-6, 2006, pp. 193–158 (2006)
28. Wagner, G., Giurca, A., Lukichev, S.: 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)
29. Wagner, G., Lukichev, S., Fuchs, N.E., Spreeuwenber, S.: First-Version Controlled English Rule Language. In: The Rewerse Group, pp. 1–47 (2005)
30. Wagner, G., Tabet, S., Boley, H.: MOF-RuleML: The Abstract Syntax of RuleML as a MOF Model, http://www.ruleml.org
31. Wahler, M., Ackerman, L., Schneider, S.: Using IBM Constraint Patterns and Consistency Analysis. IBM Developer Works (May 2008)
32. Wahler, M., Koehler, J., Brucker, A.D.: Model-driven constraint engineering. In: MoDELS Workshop on OCL for Meta-Models in Multiple Application Domains, Electronic Communications of the EASST, Technische Universität Dresden, Germany, vol. 5, pp. 1–15 (2006)