MANAGEMENT OF BUSINESS RULES USING DECISION TABLES*

Kestutis Normantas, Olegas Vasilecas, Sergejus Sosunovas

Vilnius Gediminas Technical University, Information Systems Research Laboratory, Sauletekio al. 11, LT20221, Vilnius, Lithuania, {kestutis.normantas,olegas,sergejus}@isl.vgtu.lt

Abstract UML has established itself as the leading OO analysis and design methodology. This paper presents a business rules engineering approach to define decision tables in consistency with Unified Modeling Language (UML) model. Supporting tools are under the development on top of a commercial CASE tool for UML.

Keywords: Business rules, decision tables, UML, OCL

1 Introduction

Business rules are evidently important for organizations as they describe how they are doing business. Their value has also been recognized within the information system (IS) domain, mostly because of their ability to make applications flexible and amenable to change [1]. During the last years together with increasing importance of research in semantics the interest in rule based software systems and their development is renewed and increasing. This results in a numerous rule engines available today on the market.

Decision table is one of the oldest techniques to enter business rules in the rule engine [2]. Official decision table components and terminology were standardized by the cooperative efforts of a number of organizations, including the Conference on Data Systems Languages (CODASYL) in 1962. The basic format of decision tables remains largely unchanged today [3]. At the moment this method is successfully used in the leading business rules product (e.g. Corticon BRM Platform, Fair Isaac Blaze Advisor, ILOG JRules, Resolution EBS iR Platform etc.) [4]. Decision tables are a form of structured lookup capability that provides the ability to construct and update conditions and actions for rules as a cross-indexed chart or table.

It is important to notice that business rule engine cannot be used alone; more often it is integrated into a bigger information system. The recent research on the development of business rules based systems reveals that the half of the business rules systems contains not more than 120 rules and takes 9 man months to develop. At the same time at the average the rule base is part of a much larger software system that takes almost 60 person months to develop [5]. Therefore the methods to develop and present business rules systems should be seamlessly integrated with the methods used to develop general purpose software systems.

UML is a popular way to model software system. In order to model information system using UML it is necessary to introduce a way business rules in the form of the decision tables can be integrated with UML model. In this paper we advocate an approach of using metamodel based integration of decision tables and UML. The necessity to use decision tables with UML is mentioned in [6].

This paper starts by giving a brief overview of related work. Then in Section 2, it gives an overview of proposed method for business rules specification and implementation in modeling process. Next in Section 3, it illustrates the approach by applying it to a simple problem. Finally in Section 4, the paper discusses the results of the research presented here and makes the conclusions.

2 Related work

Business rules are important in the modeling of business system: they help to define the business terms and facts (structural assertions) as well as the constraints underlying the business behavior (action assertions). According to [7],[9] business rules represent core business policies that capture the nature of an enterprise’s business model and define the conditions that must be met in order to move to the next stage of the process. Business rules are represented as compact (declarative) statements about an aspect of the business that can be expressed within an application in unambiguous terms that can be directly related to the business and its collaborators and as such they determine the route of action to be followed [8].

Business rules can be integrated with enterprise applications so that they can be used for business decision making, using ordinary business data. Business rules, in general, automate and facilitate business processes. They allow business analysts and even users to create, understand and maintain the rules and policies of the business and associate them with relevant business processes. They are usually grouped into independent but chainable rule-sets and perform inferences within and over such rule-sets [9].

* The work is supported by Lithuanian State Science and Studies Foundation according to High Technology Development Program Project “Business Rules Solutions for Information Systems Development (VeTIS)” Reg. No. B-07042

- 144 -
Since its introduction, UML has quickly been adopted as the standard Object-Oriented (OO) modeling language for modeling software systems [16]. The Object Constraint Language (OCL) [10][11][12] was introduced to supplement UML to capture more details of modeling domain. Nowadays, UML/OCL is supposed to be used for specification of business systems as well and it seems that popularity and number of tool supporting business related UML notation increase. CASE tools, such as MagicDraw UML, support easy construction of models which capture business processes and rationale. Once embedded in the model, however, this information becomes somewhat inaccessible. In order to be able to extract the information from the model one generally needs a working knowledge of the UML/OCL and some knowledge of the operation of the CASE tool. Thus UML lacks both a reference implementation and a human-readable semantic account to provide an operational semantics [6].

We suppose that decision table technique could partially fill this gap. Because of this technique is easily integrated with other techniques, binding with UML/OCL would not produce any problems in the modeling process: the UML provides three mechanisms for extending the language's syntax and semantics: stereotypes (which represent new modeling elements), tagged values (which represent new modeling attributes), and constraints (which represent new modeling semantics) [16]. On the contrary, it would supplement OO modeling technique with understandable for non-technicians form for specification of subset of business rules. As a result of decision table constructing, different artifacts may be produced, such as OCL expressions or activity diagrams. Of course, reverse binding is possible as well and will be provided in further research. Unfortunately, limited subset of business rules can be modeled with decision table technique. This includes conditional constraints, derivations and process rules [15].

Binding of decision table technique with Entity Relationship (ER) seems to be successfully realized in research made in [13]. In this research was used metamodel based transformation approach to produce ER model from DTs. It was shown that decision tables are applicable for business rule specification stored in XML using decision table metamodel. Related approach could be adopted to recent research as well.

A decision table is a business rule in form of table, which consists of a set of conditions and a set of actions or sometimes called conclusions. Conditions usually are represented as rows and actions as intersection points of the conditional cases in the table. Decision tables are best suited for business rules that have multiple conditions. Augmenting table with another condition is done by simply adding another row or column. In a decision table, the action is decided by more than one condition, and more than one action can be associated with each set of conditions. If the conditions are met, then the corresponding action or actions are performed.

Though originally used as a technique to support programming, decision tables have proven a useful aid in modeling complex decision situations of various sorts. In the field of knowledge-based systems research, there has been a renewed interest in decision tables over the past years. Here, decision tables have been studied or applied in the following contexts: verification and validation of knowledge-based systems, efficient execution of knowledge-based systems, knowledge base maintenance, knowledge acquisition, knowledge discovery, several application domains such as medicine, law, etc [14].

Because of so large interest in decision table technique, different kinds of them originate. The main difference is depending on whether all columns are mutually exclusive or not. If so, then each possible combination of conditions matches exactly one column. Otherwise, some combinations of conditions match more than one column, which may lead to ambiguity or inconsistency. In such case sequence of table conditions hits (selection) is very important: the first hit (when scanning the table from left to right) will determine which set of actions has to be executed, thus preventing contradictions. In this paper, we stand with mutually exclusive arrangement of columns as it will be shown in the next subsections.

3 Decision table based approach for business rules representation and implementation in OO modeling process

3.1 Decision table construction process

The main idea for binding of decision table technique with UML/OCL is to invoke business people to participate in the software development process in the earliest stages and to talk to them in understandable to them form. Therefore, participation of business related people (e.g. stakeholder) in construction of decision table could produce precise understanding of particular business rules.
Figure 1 represents decision table construction process. The first step in the construction process is identification of condition labels. In the simplest case condition labels may be either attributes names, or operations (which return some result) names. The second step is to identify conclusions (or actions). Conclusions as the conditions may either attributes names, or operations names. The third step is identification of rules. In this step possible alternatives of condition labels are in consideration. Identification involves filling of possible values for condition labels and introduction of new alternative rules in the model. After, if it is required, then reordering of the condition labels is made. In the next step conclusions are filled according to formed rules. After it, validation of decision table with stakeholders could be applied to check whether the constructed table is appropriate to business rule. In the case if it is not so, additional check of decision table construction should be provided.

3.2 Business rules specification and artifacts generation

To illustrate our approach consider a simple problem. Calculation of delivery price depends on the price of the order, preferred delivery time and on the customer status in business accounting system. Delivery time may be a month, a week or a day. Customer status may gold, silver or bronze. Related class diagram is presented Figure 2.

Possible delivery prices according to mentioned above parameters are presented in decision table (Table 1). Decision table’s abstract sphere consists of condition labels and appropriate actions. For this example, it is composed of only attributes of classes for the simplicity of the example; additionally, it may be composed of operations of classes or OCL expressions returning values to be assigned to the concrete sphere’s rules.
Table 1. Decision table for calculation of delivery price

<table>
<thead>
<tr>
<th>Abstract sphere</th>
<th>Concrete sphere</th>
</tr>
</thead>
<tbody>
<tr>
<td>Order price</td>
<td></td>
</tr>
<tr>
<td>&lt;1000</td>
<td>&gt;=1000</td>
</tr>
<tr>
<td>Delivery time</td>
<td>day</td>
</tr>
<tr>
<td></td>
<td>week</td>
</tr>
<tr>
<td></td>
<td>month</td>
</tr>
<tr>
<td>Customer status</td>
<td>G</td>
</tr>
<tr>
<td>S</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td></td>
</tr>
<tr>
<td>G</td>
<td></td>
</tr>
<tr>
<td>S</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td></td>
</tr>
<tr>
<td>G</td>
<td></td>
</tr>
<tr>
<td>S</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td></td>
</tr>
<tr>
<td>G</td>
<td></td>
</tr>
<tr>
<td>S</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td></td>
</tr>
<tr>
<td>Actions</td>
<td></td>
</tr>
<tr>
<td>Delivery price</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>15</td>
</tr>
<tr>
<td></td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>12</td>
</tr>
<tr>
<td></td>
<td>16</td>
</tr>
<tr>
<td></td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>7</td>
</tr>
<tr>
<td></td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>5</td>
</tr>
<tr>
<td>Rules</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

* (G-gold,S-silver,B-bronze)

The result of decision table depends on the related model elements where decision table is applied. In current example it is applied to deliveryPrice attribute of the class Order. Thus result could be invariant on order class or initialization expression of deliveryPrice attribute. OCL invariant generated from decision table would be:

```ocl
class Order
inv deliveryPriceConstraint:
  if self.price < 1000 then
    if self.deliveryTime=DeliveryTime::day then
      if self.customer.status=Status::gold then
        self.deliveryPrice = 10
      else
        self.deliveryPrice = 10
      endif
    else
      self.deliveryPrice = 10
    endif
  elseif self.price >= 1000 then
    if self.customer.status=Status::gold then
      self.deliveryPrice = 10
    else
      self.deliveryPrice = 10
    endif
  else
    self.deliveryPrice = 10
  endif
```

As we can see from the above example, description of simply problem may become complicated enough when it is expressed by if-then-else statements. Modification of condition value would lead to modification of OCL expression in general. However a table with many different parameters quickly becomes complex what diminishes its usefulness as a simplification concept. Therefore additional algorithms should be invoked to optimize it as well as friendly graphical user interface for filling of decision table form should be provided.

If table is applied to an operation of a class, then result may be a post-condition or a body of an operation. Construction of OCL expression would be very similar to expression above. Result of an operation may be activity diagram as well, as shown in Figure 3.

Figure 3. Fragment of resulted activity diagram from decision table

Figure above presents fragment of activity diagram generated from relating decision table. Activity diagram represents the logic of delivery price calculation process. Conversions of decision table to program algorithms are Parameters of table columns are used as guards on activity flows. Conclusion is represented as
actions `setDeliveryPrice` with different input pins which are evaluated depending on activity flows. Regarding to OCL usage with UML, guards on activity flows as well as input pin could expressed as OCL expressions.

3.3 Mapping of decision table metamodel to UML metamodel

We have used simplified MOF metamodel for decision table presented in [15] and MOF metamodel for UML to show how decision table could be used to specify business rules for a class diagram. Additional mapping of decision table metamodel with OCL and UML for activity diagrams will be provided in further research.

![Mapping of decision table metamodel to UML metamodel](image)

Figure 4. Mapping of decision table metamodel [15] with UML

Figure 4 presents suggested mapping of decision table metamodel elements to UML metamodel elements. The composition of concrete sphere’s rule set consists of an ordered composition of condition labels. These condition labels group the conditions of the rule set in sets of exhaustive and mutually exclusive conditions. Each condition is a logical formula that refers to the domain atoms and variables of the rule set. The actual rules of the rule set are an ordered conjunction of conditions, such that a rule contains at most one condition of each condition label. Notice that not every conjunction of conditions is necessarily meaningful. In other words, it might be the case that a specific condition is only meaningful in combination with other specific conditions. To express this dependency, a decision table can make use of so called condition dependency rules [15]. UML metamodel, used in CASE tool MagicDraw UML, fragment represents the structure of the class. Class consists of operations and properties. Operation consists of parameters and is extended by setter and getter stereotypes. Property (an attribute) is extended by propertyGroup, hasGroupName and suggestedValues stereotypes. As it was mentioned before, conclusion and condition label could be mapped to operation and property metaclasses. For the implementation we use those metaclasses instances to construct decision table.

3.4 Experimental evaluation of the approach

Currently, proposed approach is implementing as the plug-in for the CASE tool MagicDraw UML (Figure 5). It is in the earliest stage of the development, therefore only generation of invariants on attribute values has been implemented.
In the current implementation, decision table may be called either from common actions toolbar, or from class context menu. Called from context menu, decision table has already assigned context class for generation of invariant. In other case, context class should be assigned configuration settings. Decision table conditions as well as conclusion values may be selected/removed from model elements selection window. Values for conditions or conclusions are inserted manually. At the moment, refinement of graphical user interface for decision table filling form as well as development of plug-in to result more artifacts is in progress.

3.5 Discussion

Proposed approach showed that decision table may simplify modeling a subset of business rules with UML/OCL. Of course, decision table is one of many techniques which could be used with UML/OCL to represent business rules to business related people in familiar to them form. We have shown that decision table may be used as graphical user interface for modeling particular subset of business rules to provide OCL expressions. Resulting artifacts of decision table is an open question and topic for the further research. Implementation details of OCL are not in scope of this paper, because there was presented the main idea; therefore, no related algorithms were discussed here. Algorithms for conversion of decision tables to OCL invariants and to activity diagrams will be published in the future.

4 Conclusions

In this paper we have set out the usage of decision table technique with standard OO modeling method UML to provide familiar to business people form for specification of particular subset of business rules. We have shown that decision tables usually used as business rule visualization technique are applicable for business rule specification with OCL.

The proposed approach allows expressing business rules in user-friendly manner as well as supplementing related UML model by adding additional elements when construction of decision table is being processed. All the limitations of the decision tables belong to the proposed approach too and should be solved using other engineering methods.

References


