Business Rule Based Configuration Management and Software System Implementation Using Decision Tables

Olegas Vasilecas, Aidas Smaizys

Vilnius Gediminas Technical University,
Sauletekio av. 11, LT-10223 Vilnius, Lithuania
olegas.vasilecas@fm.vgtu.lt, aidas@isl.vgtu.lt

Abstract. Deployment and customization of the software in different information systems of separate organizations challenge large requirement conformity, project and specification management, design and architecture complexity, code integration, compatibility and interoperability issues, frequently causing the need of reengineering through the entire all the system development lifecycle. The paper proposes new business rule based method of information system configuration management and automated way of configuration implementation into the final software system code.

1 Introduction

Business systems are functioning according to business rules in specific business domain. Due to dynamic of its nature, business environment is changing frequently according to internal and external influences such as changes in law, new competition etc. [1]. However Business rules (BR) of particular problem domain are similar in several organizations but they are never applied the same way even they depend on the same legal regulations.

Production and implementation of such a software system into the large amount of slightly different information systems of separate organizations challenge large requirement conformity, project and specification management, design and architecture complexity, produced code integration, compatibility and interoperability problems through entire all the software system development lifecycle.

The complexity of slightly different application of the business rules on every specific business system cause to chose from several alternative options:

- Splitting the source code line into the alternative paths, implementing specific business requirements by redesigning models through all the entire layers (business system, information system and software system layer) and producing separate executable code;
- Design and implementation of modular software system architecture and customization of the separate modules according to the customers needs;
- Implementation of all the possible variations of the same business rules application into the software code and activation using specific parameter values for every particular implementation;

Y. Ioannidis, B. Novikov and B. Rachev (Eds.): Local Proceedings of ADBIS 2007, pp. 27-37

– Change of business system according to constrains caused by particular implementation of business rules hard coded into the software system.

Every solution proposed before has its own specific limitations. Separate code lines result in dramatic usage of resources to manage the changes and are rarely used in practice, unless it's planed to split and develop separate software because of different target market segments, pricing policy etc.

Modular architecture allows separation of core functionality and creation of separate modules prepared for customization according to the customers needs, but it results integration problems after core modules are renewed. Such upgrades often result unacceptable risk, especially in case of business critical applications.

The solution based on configuration parameters require redesign of software system code for every implementation of every different aspect of business rule implementation according to the specific customer needs and redesign of the configuration parameter storage or at least change of configuration values. In large systems such configuration parameter set can grow dramatically and cause large configurations and complicated management. Usually it is very difficult to follow the meaning of every parameter and prevent duplication or contradicting settings that result in unpredictable behaviour of the system.

The accommodation of business system according to the proposed software or information system is the easiest way for developer, but the least acceptable for business people, because of unpredictable influence of such a modification into the business system efficiency, competitiveness and etc. This way is applied very rarely and only partly changing business processes mostly critical for implementation.

The paper proposes new business rule based method of information system configuration management and automated way of configuration implementation into the final software system code using metamodel transformations.

The paper is structured as follows: Section 2 discusses related work, in Section 3 we introduce decision table based rule model driven method for automated way of configuration implementation into the final software system code using metamodel transformations. In Section 4 we introduce current state of our research experimental implementation. Finally we present our conclusions and discuss future research in Section 5.

2. Related Work

A decision is a choice about a "course of action". A course of action may include many individual actions. A decision may be characterized on a continuum from unstructured to structured [7].

Rules can be graphically represented in multiple ways for usability purposes. Two structures are primarily used: decision trees and decision tables (DTs). Rule engines do not really care about the rule representation since either structure is compiled into similar executable rule statements.

According to [4, 6] business rules are involved in capturing requirements and creating business or information system model and stored in graphical specifications. Although the main reason why business patterns or models are not reused without

further manual interpretation is because the computation of a floating decimal number remained in gross-to-net pay obviously did not remain the same even in two installations in the same city. That means - design process does not eliminate or control further manual transformation of the requirements and the final result usually depends on the developer. Thereby by implementing the requirement in development process the relations to the business rules are lost. The other problem is that the meaning of the Business rule embedded in a graphical model can change already in design process by involvement of new business rules represented by adding new components and relations to the model. Such changes can result in contradictions and incompleteness of business rule set.

The decision table formalism is not an isolated technique and shows a lot of interfaces to other representation formalisms such as code, trees, rules, etc. Making good use of these connections, however, is only possible through flexible computer support. Therefore a variety of bridges have been built between the decision table workbench and other representations, resulting in a large application domain for decision table modelling [8]. Decision tables are context free. Unlike decision trees, no prior path must be completed for the table to arrive at a decision. Only input values for the necessary parameters along the axes are required. Decision tables are best used for a consistent, but limited, set of parameters, where potentially a large number of possible values persist for those parameters.

That is the main reason why we have selected DTs as the best solution for software configuration embedded business rules used for process logic modifications according to the business object condition represented by values of selected business object attributes. However a table with many different parameters quickly becomes complex which diminishes its usefulness as a simplification concept.

Whatever DTs is the technique used to visualize business rules (Decision table or tree viewer), it is necessary to have a language which makes it possible to specify the syntax and the semantics of the rules [8, 3]. We disagree with such a point of view into the Decision table usage. We think that in some cases it’s a perfect solution for BR specification interface design.

The business analysts or domain experts need to develop clear, precise and complete business rules that represent real life business process logics and these processes need to be logically robust as a software implementation. From the other side the programmers use the same business rules, interpreted as software system functional requirements that are later translated into a programming language that becomes part of the application. Software development involved code writing covers every possible combination of circumstances – user input, database transaction, or any other event – that the program will encounter. When the programmer or analyst overlooks a possible combination of inputs or events, the program will do things in unpredictable way and produce unexpected results. Even worse are errors made by programmers who do not fully understand the interaction between different options and their combination. The automated implementation of the DTs represented rules into the software system code would allow error prone solution of all the possible problems like omitted, duplicated, or contradictioning conditions and inadequate implementation.

There are two main views in dynamic business rule driven software system design. One of them is to design predefined executable processes and execute them by using rules in software system, where processes and execution rules are derived from business rules using transformations [5]. And other one discussed in our papers [10], where business rules and facts describing current business system state are loaded into inference engine of the software system and transformed into software system executable data analysis process according to the results of logical derivations.
The authors in [2] say that „model translations can be specified in a truly generic way: the source and the target metamodels are parameters of the entire procedure and specify the operation of model translation as a special diagram operation well known in category theory under the name of pull-back. At the heart of the approach is a mapping between the two metamodels, which governs the entire translation procedure“. We agree with the authors and have used similar approach for BR transformation, although we have used XML based transformations as described in [10]. This approach is used for automated generation of design models and final software system components according to the method proposed in next section.

3. Decision table based method for formal business rule representation and automated implementation

One part of a decision tables contains conditions that can be linked to create a rule, and the other part contains actions that are related to the rules. Once you've defined the rules, it's possible using special editor to automatically analyze them, add missing rules, and remove those that are redundant or contradictory.
In this paper we propose configuration management based on business system rules. According to the framework presented in [11] business rules are captured in business system usually interviewing business people, analyzing business processes, studying documentation and legal regulations. Captured informal business rules are placed in repository and transformed in to the information system rules using decision tables. Decision tables are directly implemented into the software system database using automated logical data model generator according to the method schema (Fig. 1.).
After that data model from DBMS is to be reverse engineered back into the software system model and used for building business logic (BL) classes and data access (DA) classes in the software system model allowing automated access and manipulation from information or business system model, implementing business services for processing business objects automatically into the separate software system module (GUI executable) later on.

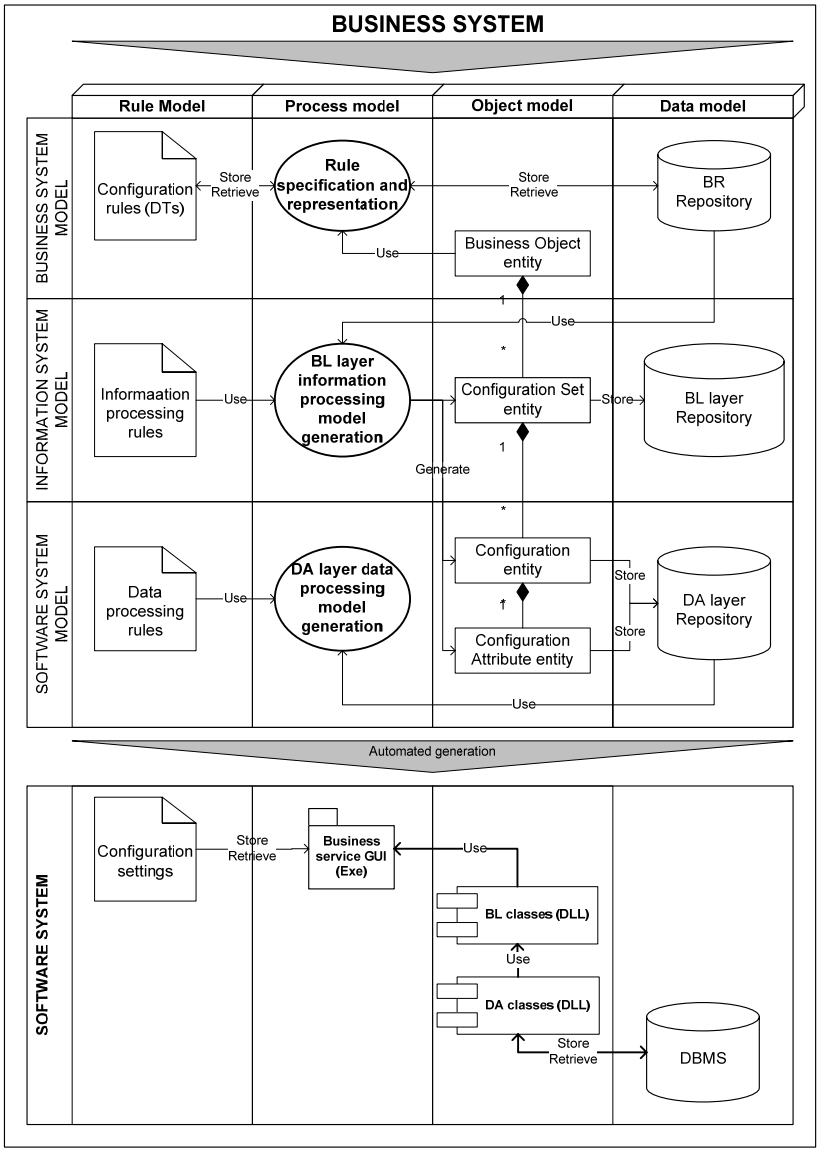

Fig. 1. The method for automated business rule based configuration implementation.

3.1 Decision table based business system process rule specification and representation

Business analysts often use decision tables to represent a particular set of requirements, so replication of that structure in the rule engine reuses a familiar

concept. Decision tables can quickly highlight where an outcome or decision is missing. Each decision table should only make one particular type of determination. Also, not all parameters have valid decision outcomes in all the cases.

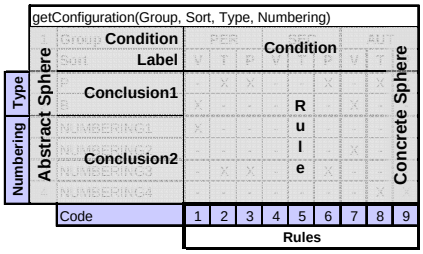

Fig. 2. Main DTs metamodel nodes

Figure (Fig. 2.) displays MOF/UML metamodel based rule set represented as decision table that captures business rules used to represent business process model which logics depends on the configuration parameters stored in the final software system. Each condition is a logical formula that refers to the domain atoms and variables of the rule set.

Business system object process rules & preconditions DTs:

getConfiguration(Group, Sort, Type, Numbering)										
1	Group	PER			SEC			AUT		
2	Sort	V	T	P	V	T	P	V	T	P
Type 1	B	X	-	-	-	-	-	X	-	-
Type 2	P	-	X	X	-	-	X	-	X	X
Numbering 1	NUMBERING1	X	-	-	-	-	-	-	-	-
Numbering 2	NUMBERING2	-	-	-	-	-	-	X	-	-
Numbering 3	NUMBERING3	-	X	X	-	-	X	-	-	-
Numbering 4	NUMBERING4	-	-	-	-	-	-	-	X	X
	Code	1	2	3	4	5	6	7	8	9
		Rules								

Rule representation for business system users:

1 IF Pass Group IS Person AND Pass Sort IS Visitor THEN issue Pass Type BarCode AND use NUMBERING1
2 IF Pass Group IS Person AND Pass Sort IS Temporary THEN issue Pass Type Proximity AND use NUMBERING3
3 IF Pass Group IS Person AND Pass Sort IS Permanent THEN issue Pass Type Proximity AND use NUMBERING3
4 IF Pass Group IS Security AND Pass Sort IS Visitor THEN issue Pass is Not Available
5 IF Pass Group IS Security AND Pass Sort IS Temporary THEN issue Pass is Not Available
6 IF Pass Group IS Security AND Pass Sort IS Permanent THEN issue Pass Type Proximity AND use NUMBERING3
7 IF Pass Group IS Auto AND Pass Sort IS Visitor THEN issue Pass Type BarCode AND use NUMBERING2
8 IF Pass Group IS Auto AND Pass Sort IS Temporary THEN issue Pass Type Proximity AND use NUMBERING4
9 IF Pass Group IS Auto AND Pass Sort IS Permanent THEN issue Pass Type Proximity AND use NUMBERING4

Fig. 3. Pass issue business process rules represented by Decision Table

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. In addition to conditions, a rule refers to one or more conclusions.

Figure (Fig. 3.) presents an example of such a DTs used for security system Pass issue process logic representation and (Fig. 4.) presents its subjects, properties and variables. As displayed in rules the selection of numbering and type of the pass is preceded according to the group and sort of the pass.

Subjects, properties, variables and values:

	ID	Code	Name	Description, processing requirements
Group	1	PER	Person	Entrance through pedestrian gates allowed
	2	SEC	Security	All security operations allowed
	3	AUT	Auto	Entrance through automobile gates allowed
Sort	1	V	Visitor	Pass used for one entrance and exit.
	2	T	Temporary	Pass used for multiple acces for temporary period
	3	P	Permanent	Pass used for unlimited access
Type	1	B	BarCode	Pass issued using BarCode card
	2	P	Proximity	Pass issued using Proximity card

	ID	Code	StartFrom	FinishOn	Current
Numbering	1	NUMBERING1	PB0001	PB9999	PB0000
	2	NUMBERING2	AB0001	AB9999	AB0000
	3	NUMBERING3	PP0001	PP9999	PP0000
	4	NUMBERING4	AP0001	AP9999	AP0000

Fig. 4. Pass issue process rule DTs subjects, properties, variables and values

We have used the especially produced DTs management software to allow addition of the attributes for condition labels and conclusions for better understanding of the DTs and generation of the configuration data tables and filling them with configuration values. Next Section presents transformation of DTs rules into the Entity-Relationship (ER) model.

3.2 Metamodel based transformation of Decision tables

We have used slightly modified MOF/UML DTs metamodel presented in [3] and created similar ER metamodel. Such metamodels were used for specification of the transformations needed to produce ER model from DTs created as described in previous Section. Models represented in typed graphs were transformed into XSD schemas and mapped according to the method described in [9]. All the mapping functions were collected into XSLT transformation schema and the transformation executed using XML parser. As a result of transformations we have got ER model used for creation of the database and filling it with data entered in DTs using generated SQL clauses.

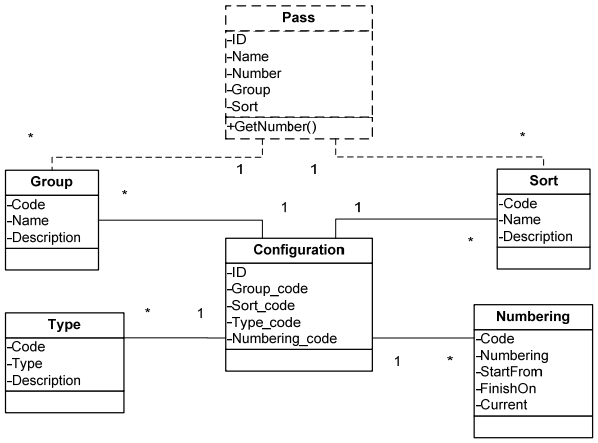

Fig. 5. The result of metamodel based DTs transformation into the ER model.

ER model was used for automated generation of the database using produced SQL CREATE TABLE, JOIN and INSERT clauses to fill the data from DT's. As the result we have got database with value filled tables related according to the ER model displayed in (Fig. 6.).

	ID	GroupCode	SortCode	TypeCode	NumberingCode
Configuration	1	PER	V	B	NUMBERING1
	2	PER	T	P	NUMBERING3
	3	PER	P	P	NUMBERING3
	4	SEC	P	P	NUMBERING3
	5	AUT	V	B	NUMBERING2
	6	AUT	T	P	NUMBERING4
	7	AUT	P	P	NUMBERING4

Fig. 6. Example of produced business system object Pass Configuration Table

Generated database was used for business object processing automation presented in next Section.

3.3 Business process automation and software system code generation

Because we have business process logics represented as DTs, data model in ER and data values filled into the database it is possible the generation of information system (IS) process code (Fig. 7.) in the software system (SS) according to the Information processing rules (Fig. 1.) that allows generation of the business object processing interface in business service tier and necessary classes in BL tier used for automation of the collection and entity operations like save, get, delete, filter etc. The access to the database is performed through generated DAO classes in DA tier according to the data processing rules stored in Template.

```
    'Generated Software system Business service tier GUI Business object processing Form code
001 Imports BL.CollectionClasses
002 Imports BL.EntityClasses
003 Imports BL.FieldClasses
004 Public Class frmPassEdit
005     Inherits System.Windows.Forms.Form
006     Private _PassEntity As PassEntity
        ...
007         Me.cbGroup.DataSource = GroupCollection()
008         Me.cbGroup.ValueMember = GroupFields.Code.Alias.ToString()
009         Me.cbGroup.DisplayMember = GroupFields.Name.Alias.ToString()
        …
010     Private Sub btnSave_Click _
011             (ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnSave.Click
012         Dim Pass As PassEntity
013         Dim Numbering As String
014         Dim Err As Boolean = ConfigurationEntity.getConfiguration _
015             (cbGroup.SelectedValue, cbSort.SelectedValue, Pass.Type, Numbering)
016         If Err then Exit Sub
017         lblType.Text = Pass.Type.Name        'Get Pass Type Name through Entity relations in BL
018         Pass.Number = NumberingEntity.getNumber(Numbering)
019         Pass.Save
020     End Sub
021 End Class

    'Generated Software system Business logic tier code using ER model
021 Imports DA.DaoClasses
022 Namespace BL.EntityClasses
023     Public Class ConfigurationEntity
            ...
024         Public Shared Function getConfiguration _
025                 (ByVal Group As String, ByVal Sort As String, _
026                  ByRef Type As String, ByRef Numbering As String) As Boolean
027             Dim Configuration As ConfigurationEntity = GetConfigurationEntity(Group, Sort)
028             If IsNothing(Configuration) Then Return False
029             Type = Configuration.Type
030             Sort = Configuration.Sort
031             Return True
032         End Function
033     End Class
034 End Namespace
```

Fig. 7. Pass issue process SS code in Basic generated from ER model and DTs

The code provides interface for business object “Pass” processing. We displayed an example of code used to save the “Pass” data. Because Type and Numbering of the “Pass” is based on Configuration – procedure code (lines 14 - 15) selects Configuration data according to the Group and Sort combo boxes in PassEdit Form. Group and Sort combo boxes are created (lines 7 - 9) using Collection and Field classes generated in BL tier. The Function getConfiguration is stored in BL tier and returns false if there is no configuration records available for selected preconditions. If configuration is available the function sets Type and Numbering values returned into the Form (lines 29 – 30). Configuration data is got from database using function getConfigurationEntity in DA tier and uses generated DaoClasses to get data from database (line 27).

4. Experimental evaluation of the method

The proposed method was used for development of automated security system used for transport and visitor control and access data exchange system in sea port. According to the proposed method we have collected informal business rules used for access restrictions and regulations and transformed them according to the proposed method presented in Section 3 into the formal decision rules in information system layer.
Such decision tables were implemented directly into the software system database and used for automated software system code generation producing separate parts of the software code allowing implementation of the business rules logics and automated software built decisions according to the parameters filled into the previously built database used as configuration storage.
We have experienced that such decision tables are very common and well understood by system engineers, but difficult to read for business users. So we designed transformations used to extract business rules implemented into Software system Configuration according to the currently entered configuration parameter values and represent them in plain text.
The same decision tables where used for transformation into the Horn clauses, used as logical control code uploaded into the security system controllers.

5. Conclusions

In this paper, we have set out for using a business rule model stored in decision tables to represent the semantics of domain specific business object attribute values dependent process logics and created a method that allows automated implementation of them into the configuration of the N-tier architecture based software system and process code.
In particular, we have shown that Decision tables usually used as business rule visualization technique are applicable for business rule specification.
The proposed method of automated software configuration implementation allows changes of process logic by changing configuration values directly in the software system and is limited only by data model implemented into the database, because if such modifications will be needed it should be produced all the automated reengineering process starting from the changes of DTs rules adding new conditions and conclusions and finishing with application of automated transformations.
All the limitations of the DTs belong to the proposed method too and should be solved using other engineering methods.
At the moment we are continuing our research in the following directions. First of all, we are working on information processing rule templates for the validation of business object entities and their attributes in the software system business logic and service tier. In addition we envision extension of the presented method, metamodels, transformations and templates to allow automated incorporation of business rules used for validation into the generated software code.

References

1. Barbara von Hale, Business Rules Applied: Building Better Systems Using the Business Rules Approach. John Wiley & Sons, Inc., New York, 2001.
2. Diskin Z., Dingel J., *A metamodel independent framework for model transformation: Towards generic model management patterns in reverse engineering* // Proceedings of the 3rd International Workshop on Metamodels, Schemas, Grammas and Ontologies for reverse engineering (ATEM-2006), 2006, pp. 55-66.
3. Goedertier S., Vanthienen J., *Rule-based business process modeling and execution* // Proceedings of the International IEEE EDOC Workshop on Vocabularies, Ontologies and Rules for The Enterprise (VORTE 2005), 2005, pp. 67-74.
4. Kilov H., Simmonds I., *Business patterns: reusable abstract constructs for business specification.* Implementing Systems for Supporting Management Decisions: Concepts, methods and experiences, Edited by Patrick Humphreys et al, Chapman and Hall, 1996, pp. 225-248.
5. Orriens B., Yang J., Papazoglou M. P., *A Framework for Business Rule Driven Service Composition*. // Proceedings of the Fourth International Workshop on Conceptual Modeling Approaches for e-Business Dealing with Business Volatility, Chicago, United States, October 13-16, 2003. Lecture Notes on Computer Science, Springer-Verlag, Berlin, Germany.
6. Orriens B., Yang J., *Model driven service composition //* Proceedings of the First International Conference on Service Oriented Computing, Trento, Italy, December 15-18, 2003.
7. Simon H. A., The new science of management decision, Prentice Hall PTR, 1977.
8. Vanthienen J., *Ruling the Business: About Business Rules and Decision tables.* Vandenbulcke J., Snoeck M. (eds.): New Directions in Software Engineering, Leuven University Press, Leuven, 2001, pp. 103–120.
9. Vasilecas O., Smaizys A., *Business rule specification and transformation for rule based data analysis //* Proceedings of 19th International Conference on Systems for Automation of Engineering and Research (SAER 2005), Sofia: CAD Research and Development Centre "Progress", 2005, pp. 35-40.
10. Vasilecas O., Smaizys A., *Business rule based data analysis for decision support and automation*, International Conference on Computer Systems and Technologies - CompSysTech'06, 2006, pp. II.9-1-II.9-6.
11. Vasilecas O., Smaizys A., The *framework for business rule based software modeling: An approach for data analysis models integration.* Databases and Information Systems IV, vol. 155 Frontiers in Art, IOS Press, Amsterdam, 2007, pp. 175-188.