An Architecture to Infer Business Rules from Event Condition Action Rules Implemented in the Persistence Layer

Chapter · October 2013

DOI: 10.4018/978-1-4666-4667-4.ch008

CITATIONS
5

READS
1,033

4 authors:

Carlos Arévalo
Universidad de Sevilla
10 PUBLICATIONS **52** CITATIONS
SEE PROFILE

María Teresa Gómez López
Universidad de Sevilla
98 PUBLICATIONS **404** CITATIONS
SEE PROFILE

Antonia M. Reina Quintero
Universidad de Sevilla
55 PUBLICATIONS **226** CITATIONS
SEE PROFILE

Isabel Ramos
Universidad de Sevilla
72 PUBLICATIONS **761** CITATIONS
SEE PROFILE

An Architecture to infer Business Rules from Event Condition Action Rules implemented in the Persistence Layer

Carlos Arévalo Maldonado, M. Teresa Gómez-López, Antonia M. Reina Quintero, Isabel Ramos
E.T.S de Ingeniería Informática. Departamento de Lenguajes y Sistemas Informáticos, Universidad de Sevilla, Spain
{carlosarevalo, maytegomez, reinaqu, iramos}@us.es

ABSTRACT

The business rules that govern the behaviour of a business process can be hardcoded in different ways in a software application. The modernization or improvement of these applications to a process-oriented perspective implies typically the modification of the business rules. Frequently, legacy systems are not well-documented, and almost always, the documentation they have is not updated. As a consequence many times is necessary the analysis of source code and databases structures to be transformed into a business language more understandable by the business experts involved in the modernization process.

Database triggers are one of the artefacts in which business rules are hardcoded. We focus on this kind of artefacts, having in mind to avoid the manual analysis of the triggers by a database expert, and bringing it closer to business experts. To get this aim we need to discover business rules that are hardcoded in triggers, and translate it into vocabularies that are commonly used by business experts. In this paper we propose an ADM-based architecture to discover business rules and rewrite then into a language that can be understood by the business experts.

KEY TERMS: Model Driven Modernization, Legacy Systems, Architecture-Driven Modernization (ADM), SQL, Metamodel, ECA Rules, Triggers.

INTRODUCTION

In recent year's management theory, it has been attached high relevance to a process-oriented perspective on organizational (re)structuring. One of the main reasons for the evolution of information systems in organizations is the need for changing their business processes and requirements. Yet to date, organizations still experience difficulties to adapt their information systems to this process-oriented perspective, especially because it requires them to undergo a modernization process, in which business experts are involved. Unfortunately, it is common to find that the documentation about business rules is outdated and, as a consequence, the only source of information about current business rules is databases and source code. But these technical artefacts are hard to understand by business experts.

Modernization processes typically start with a discovery phase in which technical artefacts, such as source code and databases structures, are analyzed. Triggers or integrity constraints are one of the technical artefacts in which business rules are hardcoded. These represent rules for correct persistent

states that a database can take and define the bounds for well-formed transactions that are allowed against database.

The modernization of software applications in a system described by means of a business process model involve two important activities: the definition of the model and the description of the business rules. The process model is described in a language such as Business Process Model and Notation (BPMN) (OMG, 2010b). However, the policies or statements that govern the behaviour of the company need to be described by means of business rules, for example with a language such as Business Vocabulary and Business Rules (SBVR) (OMG, 2008). Business rules can be seen as a common language between the business-side and the IT-side of organizations. The necessity to combine both perspectives (imperative and declarative) has been analyzed in papers such as (Skersys, T. et al., 2012a), (Skersys, T. et al., 2012b). This paper is focuses on the modernization process of the declarative description of the business process by means of business rules. Therefore, the business rules need to be captured from legacy information systems, since often the documentation is outdated or, simply, does not exist. In these cases, several types of sources must be analyzed to discover the business rules. Legacy Information Systems represent a serious problem for software maintenance process (Bisbal J., Lawless D., Wu B., Grimson J., 1999). (Stavru S., Krasteva I., Ilieva S., 2013) analyze the challenges for legacy information systems in the scope of Model Driven Modernization, with an enumeration of the main organizational and technical categories of challenges

One type of source is triggers, which are hardcoded in databases. They describe the relation between data values and are commonly written in proprietary languages, such as PL*SQL or Transact*SQL, which are difficult to understand by a non-database expert. In these cases, it is common to manually translate the source code of triggers to natural language or a business language, in order to make it easier to understand by the business expert. Triggers are, on the one hand, a well-structured knowledge base, linked to tables in databases and, on the other hand, a clear specification of the events that launch actions or methods to be executed. However, the procedural code of triggers is close to programmers but too distant from natural language, and, therefore, far from the language that business experts handle.

To bring these two worlds closer, in this work we consider a model-driven reverse engineering process, in which a set of metamodels at different levels of abstraction are provided. The levels of abstraction are proposed in Architecture Driven Modernization (ADM) (OMG, 2010), that correspond, bottom-up, to Platform-Specific Model (PSM), Platform-Independent Model (PIM) and Computation-Independent Model (CIM). As is analysed in the section of related works, although there are several solutions that analyse the modernization process in business processes, none of them tackling all the steps of the problem.

The paper is organized as follows:

- First section exposes foundations for most of relevant aspects and technologies used in this paper in order to help the reader to understand the context of the paper.
- Second section shows a running example to illustrate our proposal.
- Third section presents the details for artefacts needed to discover business rules hardcoded inside a set of triggers.
- Next section describes the ADM-based architecture proposed in this paper and the metamodels used in the different levels of modelling defined in the architecture.
- Then, an analysis of the main works related to our proposal and the gaps that can be covered is made.
- Finally, conclusions and further work are presented.

FUNDAMENTALS

The aim of this section is to lay the foundations of the approach presented in this paper. For that, and in order to better understand the paper, some terminology and technologies are introduced. Firstly, the technologies and artefacts used in a model-driven modernization process are explained. Then, the most relevant business rule classifications are presented in the subsection *Business Rules Classification and Data Rules*. After that, SBVR as a language to describe business rules is briefly presented. Finally, a tour of the technologies used to implement database triggers is made.

Model-Driven Modernization

Modernization or reengineering processes are guided by the horseshoe model (Kazman, R., et al., 1998), a well-known framework which integrates the different activities and abstraction levels involved in this kind of processes. The horseshoe model aims to obtain an abstract representation of the source system (legacy system) in order to improve its understandability and its transformation to the target system (improved system. Figure 1 shows a picture representing the horseshoe model. In this model there are three main phases: reverse engineering, restructuring and forward engineering. Firstly, reverse engineering is the process of analyzing a subject system to identify the systems' components and their interrelationships and create representations of the system in another form or at a higher level of abstraction (Chikofsky, E. & Cross, J., 1990). Secondly, restructuring takes as input the abstract representation obtained previously and converts it into an enhanced representation at the same level of abstraction, while maintaining its external behaviour. Finally, forward engineering takes the abstract representation and generates the physical implementation of the target system to one lower level of abstraction. This paper presents an approach focused on the reverse engineering phase.

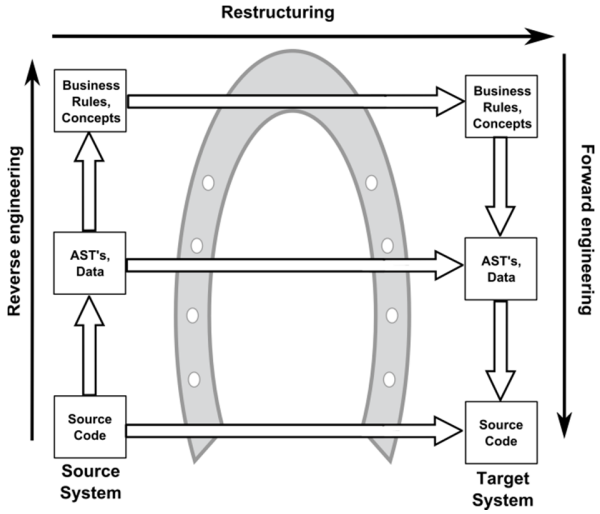

Figure 1.The horseshoe model

Model-driven development (MDD) is a style of software development in which the primary software artefacts are models from which code and other artefacts are generated by the application of successive transformations. A model is a description of a system from a particular perspective, omitting irrelevant details so the characteristics of interest are seen more clearly. MDD is gaining acceptance mainly because it raises the level of abstraction and automation of software construction (Deltombe et al., 2012). MDD

techniques such as metamodelling and model transformation can be used not only to develop new software, but also in reverse engineering (Favre, 2010) (Bruneliere et al., 2010).

Architecture Driven Modernization (ADM) (OMG 2010) is the OMG initiative that has as mission to standardize techniques, methods and tools for modernization processes using the horseshoe model as framework. ADM deals with all the software artefacts involved in modernization processes as models, and it facilitates the formalization of transformations between those models. It also advocates carrying out re-engineering processes following the standard Model Driven Architecture (MDA), which makes it possible to work with all the software artefacts in legacy systems as models and using different levels of abstraction.

MDA (OMG, 2003) is the particular realization of MDD proposed by the OMG and uses models at different levels of abstraction to separate the logic that underlies a specification from the particular properties of the middleware where the application is going to be deployed. The different modelling levels of which the architecture is composed are: Computation Independent Model (CIM), Platform Independent Model (PIM) and Platform Specific Model (PSM). In this context, model transformations are the way of obtaining one model in one level (target model) from another model or set of models from other level (source model).

Figure 2 shows the adaptation of the horseshoe model to ADM. As it can be seen, the different levels of modelling proposed in MDA (CIM, PIM and PSM) are used to guide the whole modernization process. The forward engineering stage starts from a Computer-Independent Model (CIM) that serves as the basis for code generation. According to the MDA specification (OMG, 2003), a CIM is a "view of a system from a computation independent viewpoint". CIMs do not include any details about the structure of systems, and sometimes they are called domain models or business models. They are responsible of bridging the gap between domain experts and design experts. Code generation is the result of successive model transformation stages. CIMs are transformed into Platform Independent Models (PIM). A PIM is a model without any details of the platform in which the application is deployed. That is, it describes the system, but it does not show any details of the platform. Afterwards, PIM models are transformed into Platform Specific Models (PSM), models that combine the specifications in the PIM with the details that define how that system uses a particular type of platform.

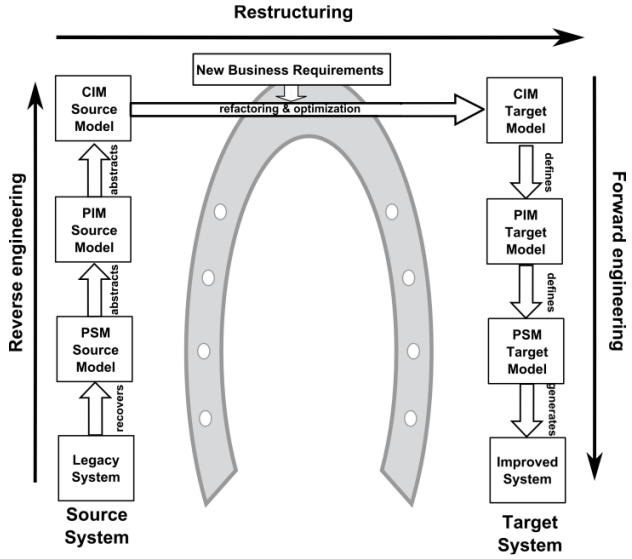

Figure 2. Adaptation of the horseshoe model to ADM

The reverse engineering stage extracts elements from legacy code and data description, rendering them into a PSM. For extracting code into a PSM the OMG's task force on modernization promoted two

metamodels: the Abstract Syntax Tree Metamodel (ASTM) (OMG, 2011c), which is a metamodel focused on syntax trees, and the Knowledge Discovery Metamodel (KDM) (OMG, 2011a), which is the pivot metamodel for modelling the whole software system and it is based on representing artefacts of existing software as entities, relationships and attributes. PIM's (or technology-neutral models) are obtained by means of applying model transformations to KDM/ASTM models. Figure 3 shows a scheme of the code extraction process proposed by the OMG's task force on modernization. The process starts by representing legacy code as ASTM models. ASTM is composed of the Generic Abstract Syntax Tree Metamodel (GASTM), a standardized language-independent metamodel and the Specific Abstract Syntax Tree Metamodel (SASTM), a user-defined metamodel which defines the concepts that are related to a particular language (Java, PL/SQL and so on). Finally, KDM is used to represent the code at flow level.

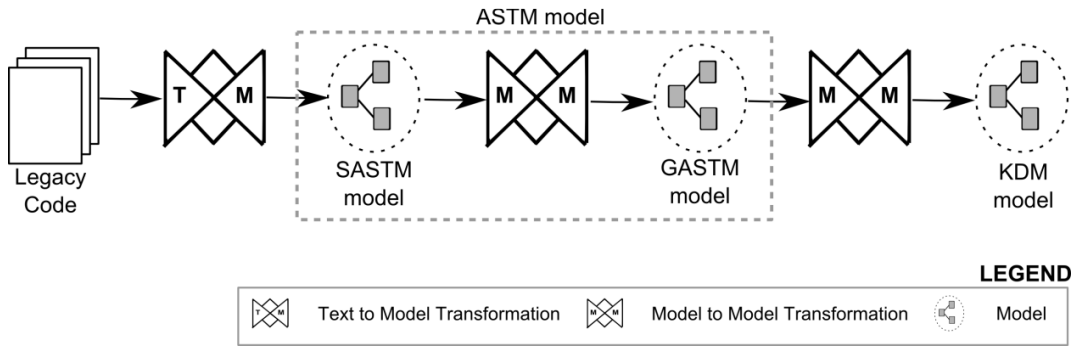

Figure 3. Code extraction process according to OMG's task force on modernization

Business Rules Classification

Although there is not a standard definition of Business Rule, it is generally understood as a rule that defines or constraints the behaviour of a company, the policies, or the used standardized. Business rules were also defined by Ronald G. Ross (Ross, Ronald G, 2003) as rules that are under business jurisdiction. This means that the business experts can enact, revise, and discontinue their business rules as they see. If a rule is not under business jurisdiction in that sense, then it is not a business rule. For example, the 'law' of gravity, obviously, is not a business rule. As there are many different types of rules, there is not a single way to treat them. In (Goedertier, S., et al., 2007) a total of sixteen business rule types are identified, that can be classified depending on the feature used to catalogue the rules.

One of the main important classifications is shown in Figure 4. This classification scheme of rules was defined by Gerd Wagner in the RuleML Initiative (Wagner, G., 2005). The classification divides the rules into five different types: Integrity, Derivation, Reaction, Production and Transformation. Since our proposal is based on the extraction of business rules from triggers, we focus on Reaction Rules, which state under which conditions actions must be taken in response to events. Reaction Rules are divided into Event-Condition-Action-Postcondition (ECAP) Rule and Event-Condition-Action (ECA) rule (Wagner, G., 2005). The semantics of ECA rule is: "*when the event has been detected, evaluate the condition, and if the condition is satisfied, execute the action*". Broadly speaking, triggers represents the same concepts as ECA rules, and for this reason, *Trigger* or *Reaction Rule* can be used as synonyms of ECArules.

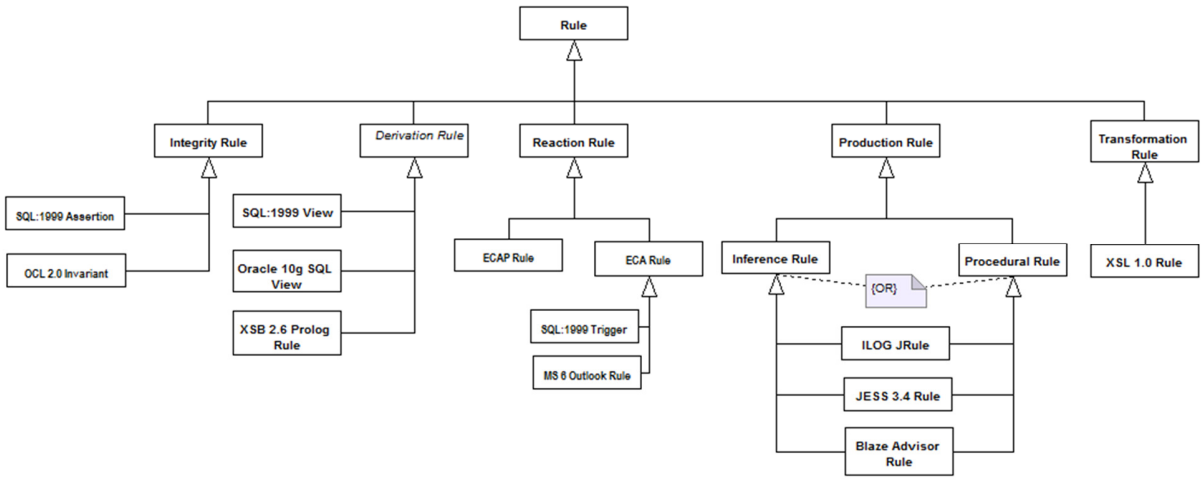

Figure 4. OMG Rules Classification

Unfortunately, the classification of Gerd Wagner is a bit far from the business process description, being necessary another classification oriented to aspects such as: control flow, for the relation that defines the ordering on the activities; organization, for the process management; and data, for the description of the data values during the instances. Figure 5 shows the classification of business rules according to the aspect they describe proposed in (Jablonski, S. & Bussler, C., 1996). Note that the types of business rules have been identified by Wagner are also included in this classification: integrity, derivation and reaction rules, but related to business process aspects.

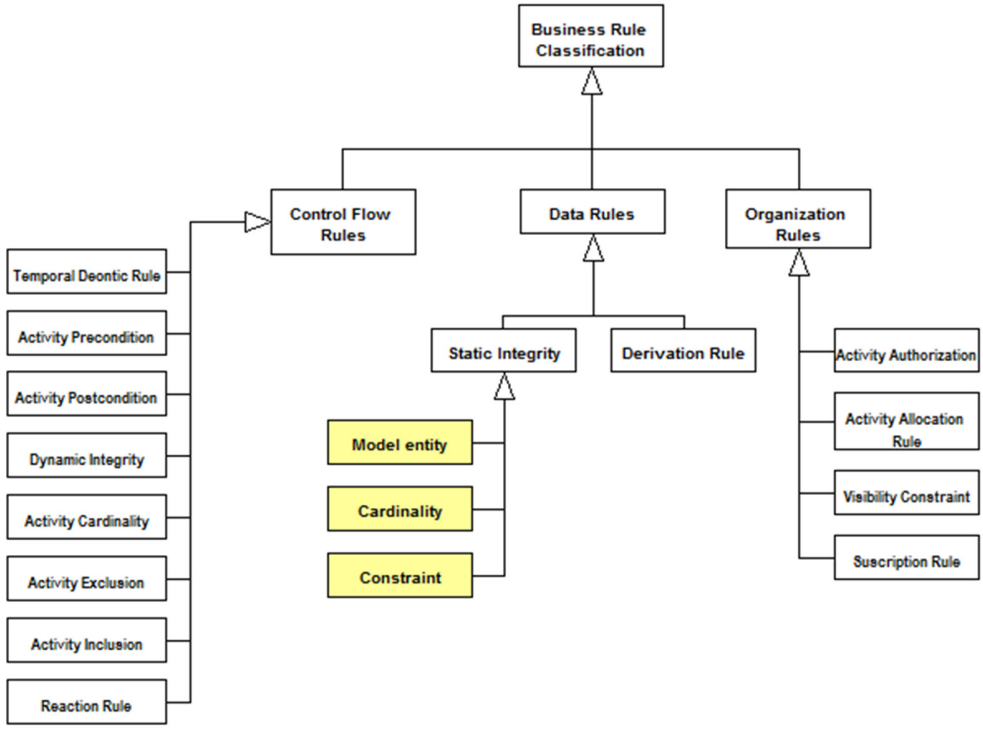

Figure 5. Business Rules Classification

In this paper we focus on those rules that can be classified as "data rules" (static integrity and derivation rules) and the data aspect of the reaction rules. Furthermore, we propose a refinement of static integrity rules, classifying them according to the kind of elements they constrain. This refinement is highlighted in Figure 5. We classify static integrity rules in: those that constrain the static model of the database; those that constrain the cardinality of the relationships between tables; and those that constrain the data values of the tuples.

A static integrity constraint is a business rule that constrains the domain over which business facts can range by expressing a logical assertion that can, cannot, must or must not remain true (Wagner, 2003). The execution of manipulation operations (addition, removal or update) can imply the evaluation of the static integrity constraints. Only those types of static integrity constraints that are related to cardinality or data values should be evaluated if the constraint is related to data involved in the operation.

A derivation rule is a business rule that defines a business fact in terms of existing business facts (Wagner, 2003). In case of derivation rules, they can be also found hardcoded as triggers that fill attributes in function of the value of other attributes, or by means of views where derived attributes are included. It means that a reaction rule can also be understood as a derivation rule that is executed when any involved data is modified. As occurred with static integrity constraints, only rules related to data that are being modified have to be evaluated. In both cases, if rules are implemented by means of triggers its evaluation is automatic, and is done exclusively for data involved in the rules.

The implementation of these "data rules" can be found in different parts of an application: triggers, embedded in source code, in the declaration of business process models. Table 2 shows a set of examples of "data rules". Each example is placed in one row of the table. The Example column holds the rule, the Type column represents the type of rule, according to the classification shown in Figure 5, and, finally, the 'Implementation in the database' column indicates the artefact used to implement the data rule.

Example	Type	Implementation in the database
The employees have name, address, department, ID, …	Model Integrity	Relational model
Each sale has mandatory one and only one agent in charge	Model Integrity and required attribute	Relational model
Each order has at least one order line	Cardinality Integrity of an association between two entities	Trigger
A responsible of a department cannot have more than 10 agents in charge	Cardinality Integrity of an association between two entities	Trigger
The agreed price of a sales item is less than or equal to the standard price of the sales item	Integrity Constraint	Trigger or with a check constraints
A luxury product has a value-added-tax of 20 percent	Derivation Rule	Trigger or a view with derived fields
The evaluation of quality of each employee is the minimum between his personal quality and his department quality	Derivation Rule	Trigger or a view with derived fields

Table 2. Examples of data rules and the type of implementation

SBVR: A language to specify the business rules

In 2008 the Object Management Group (OMG) released the Semantics of Business Vocabulary and Business Rules (SBVR) (OMG, 2008b). SBVR provides vocabulary and rules to define the semantics of business vocabularies, facts, and rules. Business-level specification aims at enterprises to formally express their operations.

SBVR proposes Structured English to specify a business model. Structure English is based on a fusion of linguistics, logic, and computer science. SBVR is a declarative language, instance of other imperative languages used in business processes, and it is close to business experts. The use of SBVR improves the understanding, creation, finding, validation, and management of business rules, and it can be further used to formalize complex compliance rules related to software. The use of natural language in SBVR provides explicitly a model of formal logic that includes concepts and terms that are more natural to business experts.

There are various candidate ontology languages (or metamodelling languages) that can be used to define a metamodel for declarative process modelling: the tandem Meta Object Facility (MOF) / Unified Modelling Language (UML), or the combination of Web Ontology Language (OWL) and Web Service Modelling Language (WSML) (Roman et al., 2005).

In (Goedertier, S. et al., 2007), the advantages of the use of SBVR were analyzed, for our approximation, these advantages have or not influence when SBVR is used to express implemented triggers:

- Model granularity. Information models can use different levels of granularity to represent concepts in the world, and postpones implementation decisions. This allows the representation of the same business rule can be implemented with different triggers, or even with another type of artefact.
- Local Closure. In SBVR it is possible to indicate the predicates (fact types) over which the model has complete knowledge. This is an aspect that can occur in the conceptual model implementation by means of databases such as production rules implemented in the business process engine.
- Business Rules as natural language expressions. Business rules are most often expressed in natural language. Consequently, the SBVR combines linguistics and formal logic. For the triggers, the formal logic is necessary for the existing formulas that describe the relation between data values.
- Rule modality. One of the characteristics of declarative process models is that they make a distinction between those business rules that cannot be violated, those ones that can be violated and guidelines. The current SBVR specification requires business rules to be a necessity, an obligation, a prohibition or a possibility. This cannot be expressed by means of triggers, where they only have the capacity to implement mandatory rules.

SBVR is situated at CIM modelling level. This level constructs business solutions for business problems. SBVR allows representing a set of concepts of a community, in which there is a shared understanding. These concepts contain noun concepts, fact types and business rules. Noun concepts represent the meaning of business objects such as Order. In the same way, fact types represent the meaning of a relation between concepts, i.e. Order contains OrderLine. Business rules are built on top of fact types and allow to constraint these fact types: Order contains at least one OrderLine.. In SBVR a vocabulary and a set of rules make up a so called conceptual schema. A conceptual schema with an additional set of facts that adheres to the schema is called a conceptual model.

Different language or speech communities can then assign a representation to these concepts making possible to talk about the same concepts in different languages. One way of representing concepts in SBVR is by means of a structured, English vocabulary for expressing vocabularies and rules, called SBVR Structured English. A technique used by SBVR structured English is the use of font styles to designate statements with formal meaning. In particular,

- the term (written in green font) is used to designate a noun concept.
- the name (written in green font) designates an individual concept.
- the verb (written in blue font)is used for designation for a verb concept.
- the keyword (written in red font)is used for linguistic particles that are used to construct statements.

Triggers implementation technologies

Triggers are a well-structured knowledge base, linked to tables in the database. They have a clear specification of the events that launch the action or method to be executed. However, the procedural code is closer to the programmer but too distant to natural language or language that commonly handles a business expert. Database triggers have been extended widely used technique to enforce business rules with database transactions. A trigger in a database is a common way to specify complex restrictions that cannot be declared with other clauses of the Data Definition Language (DDL) (uniqueness, referential, ranges, etc.). Triggers are closed to the ECA-rules in the sense that they are executed when a database event related to a table or attribute is reached.

There is a large sample of complex legacy systems, whose persistence layer is represented by databases supported by robust Database Management Systems (DBMS) like ORACLE, SQL*Server, Postgres, MySQL, etc. These systems allow the encapsulation of significant number of business rules, but are hard to understand because, typically, database administrators and programmers structure these rules by triggers written in proprietary languages such as PL*SQL, Transact*SQL, PL/pgSql, SPL, etc.

DBMS software vendors recommend triggers as the way for implementing a variety of rules, like more is not expressive enough to allow these assertions. It is recommended that triggers do not have more than fifty or one hundred lines of code, however, it would be able to find programs including simple code or very complex code.

RUNNING EXAMPLE

In order to understand the different levels of abstraction, and what can be obtained in each phase of the reverse engineering process, we propose an example where some simple triggers are associated to a table (Figure 6a); the triggers are coded in PL/SQL for Oracle 9i (Figures 6b and 6c).

```
create table employees    -- Table employees
        (       oidEmp  smallint,
                employee varchar(25) NOT NULL,
                salary number(11,2) NOT NULL CHECK (salary between 0 and 10000),
                startDate date NOT NULL,  -- Date of entrance in the company
                endDate date,             -- Date of leaving the company
                salaryLastDate date,      -- Date to apply the salary
                primary key(oidEmp)
        );
```

Figure 6a. Running example: Oracle table definition

```
create or replace trigger "R01_Remp_001"
before insert or update of startDate,salaryLastDate on employees
for each row
when (new.salaryLastDate<new.startDate OR new.salaryLastDate is null)
begin
        :new.salaryLastDate := Greatest(:new.startDate,:new.salaryLastDate, :old.salaryLastDate);
end;
```

Figure 6b. Running example: trigger for *salaryLastDate* state change

```
create or replace trigger "R02_Remp_002"
before update of salary,salaryLastdate on employees
for each row
  begin
    if (:old.salary is not null and :new.salary is not null) then
        if :new.salary>1.2*:old.salary then
              raise_application_error(-20000, 'R02_Remp_002_A02: The salary cannot grow over 20%');
        end if;
        if :new.salary<0.85*:old.salary then
              raise_application_error(-20000, 'R02_Remp_002_A02: The salary cannot decrease under 15%');
        end if;
    end if;
  end;
```

Figure 6c. Running example: trigger for *salary* state change

"R01_Remp_001" trigger encodes a single reaction rule for the applicable date to a new established salary:

Rule A: "The date to apply a new salary must be not before the starting date of working in the company and the date applied to the last salary. It must be the greatest one of the three dates".

"R02_Remp_002" trigger is a representation of a ruleset; each rule is a constraint rule. The rule set is:

Rule A: "The salary cannot grow over 20%"
Rule B: "The salary cannot decrease fewer than 15%"

These triggers, implemented at the PSM level, are transformed into a more abstract model at a PIM level, for example, using UML and OCL. In a PIM scenario, with UML2 and OCL, we can capture inside classes the corresponding entities that are associated to database tables, and also, their declarative constraints, but not, all the rest of rules inside triggers. Those additional rules can be represented in OCL. In Figure 6d, we show the OCL rules equivalent to the above triggers.

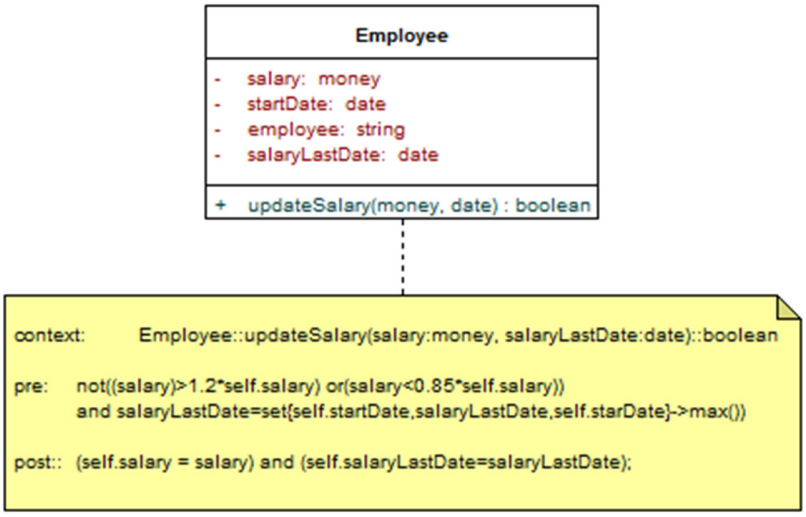

Figure 6d. Running example: a PIM model in UML+OCL

In order to transform the trigger into a model described in UML and OCL, it is necessary to define the transformation of the 'conditions', 'events' and temporal logic represented by the table columns referenced by '*:new.x'* and '*:old.x',* that allow the programmer to show values before and after the trigger is executed. In OCL, one way to represent this behaviour is the use of pre-conditions and post-conditions associated to the execution of methods. Thus, we map the behaviour of the *updateSalary()* method and we put the corresponding pre-conditions and post-conditions. In Figure 6e we show two possible scenarios or running states: the first one illustrates a scenario in which changes are committed, while the second one illustrates a scenario in which changes are rollback because of rule violation.

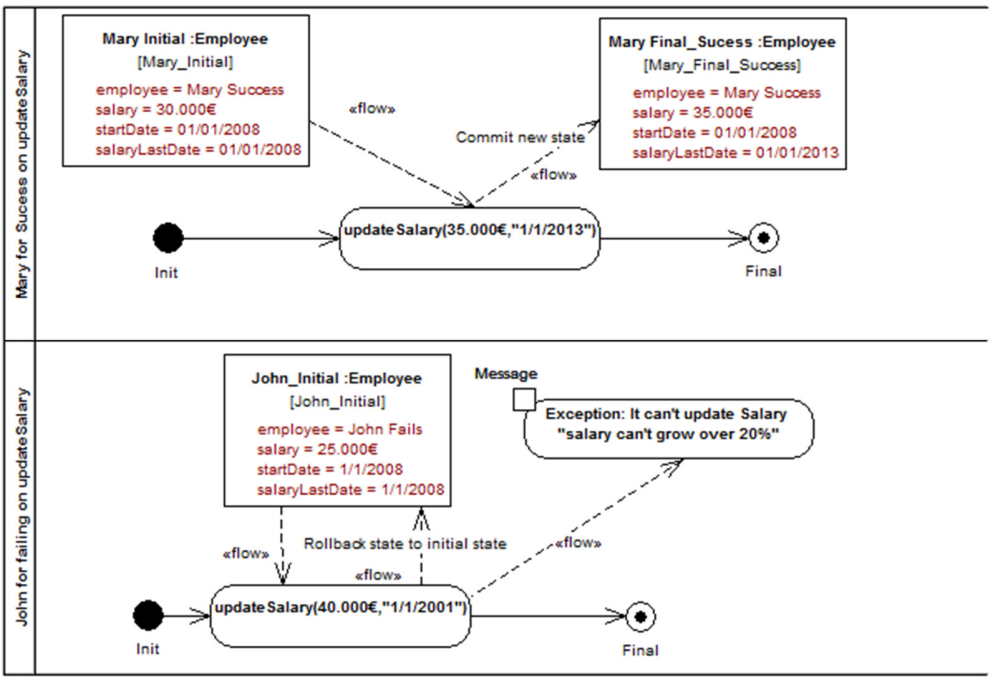

Figure 6e. Running example: Dynamic scenarios for salary updating

In order to represent the semantics of triggers in a CIM level, SBVR can be used. Figure 6f shows the SBVR rules, which correspond to the triggers shown in Figures 6b and 6c, respectively. This representation of triggers is more understandable by the business expert. In following sections, details of the different metamodels in the different levels of abstractions are presented.

The value of salary last date modification of an employee is equal to the maximum between the value of the date where started to work, the new salary last date or the old value of salary last date.

It is not possible that the salary of an employee was updating more than 1.2 or less than 0.85 of the old salary.

Figure 6f. Running example: SBVR corresponding rules

PROPOSED ADM-BASED ARCHITECTURE

As it has been introduced in the previous section, triggers need to be transformed from the code implemented in the database, to a language that can be better understood by business experts. The process consists of a set of successive transformations that involved models expressed in different levels of abstraction (PSM, PIM and CIM). In this section we explain an ADM-based architecture and the metamodels that we propose to model artefacts at each modelling level. Figure 7a illustrates the three levels of modelling, the type of transformations between them and the metamodels used at each level. As it can be seen, a SBVR metamodel is used at CIM level, an ECA Rules metamodel is used at PIM level, and a trigger metamodel based on the SQL99 standard is used at PSM level.

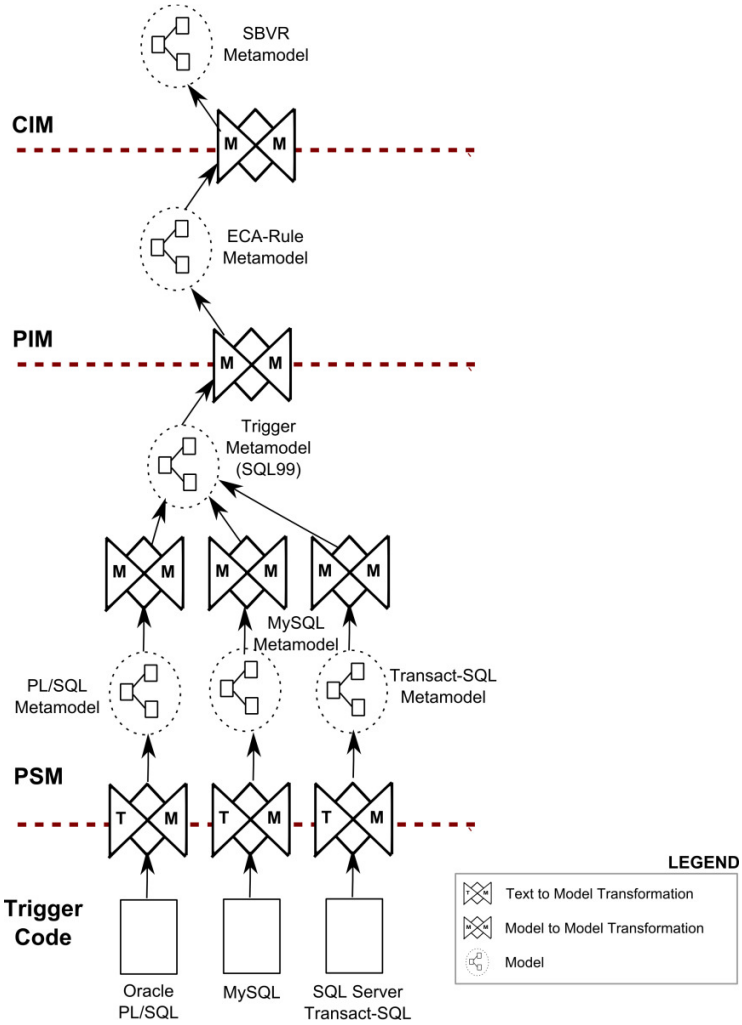

Figure 7a. The modelling levels in the reverse engineering phase of our ADM-based approach.

CIM: SBVR

The SBVR specification essentially defines two metamodels in the form of "vocabularies":

- the SBVR vocabulary for Describing Business Vocabularies, and
- the SBVR vocabulary for Describing Business Rules, which builds on the Vocabulary for Describing Business Vocabularies. A business vocabulary is defined to contain "all the specialized terms and definitions of concepts that a given organization or community uses in their talking and writing in the course of doing business". The SBVR business vocabulary metamodel is rather large with more than one hundred concept definitions. The SBVR business rule metamodel, containing 33 concept definitions, is more handy but still sizeable.

Rules that describe data_rules, and that can be implemented by means of triggers, correspond with the definition of structural (business) rules in SBVR definition (OMG SVBR, 2008). A structural (business) rule is a (business) rule that is intended as a definitional criterion. A structural rule expresses a necessity that cannot be violated.

In SBVR, meaning remains separate from expression. The SBVR provides a vocabulary called the Logical Formulation of Semantics Vocabulary to describe the structure and the meaning of vocabulary and business rules in terms of formalized statements about the meaning. Such formalized statements are semantic formulations (Baisley et al., 2005)(See a reduced version of the MOF metamodels for vocabularies and logical formulas in Figures 7b and 7c).

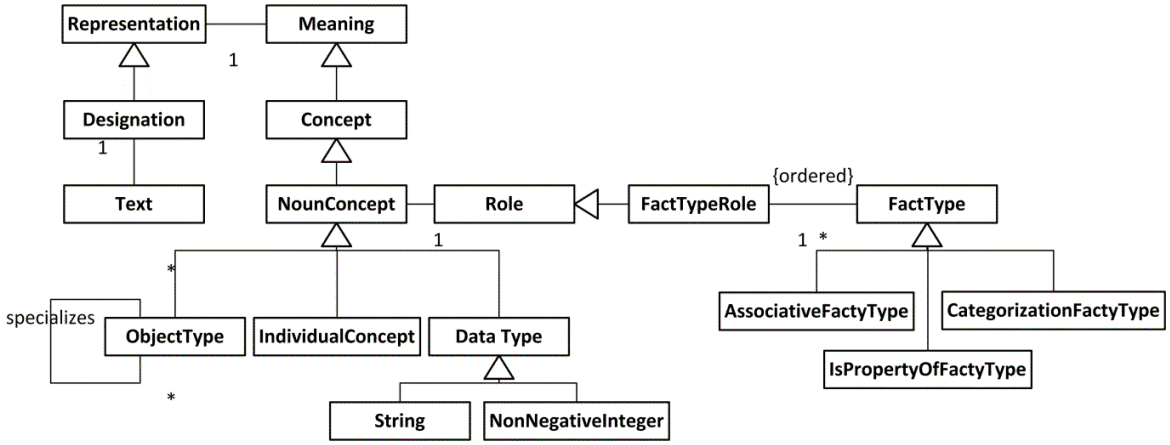

Figure 7b. Reduced Business Vocabulary Metamodel of SBVR

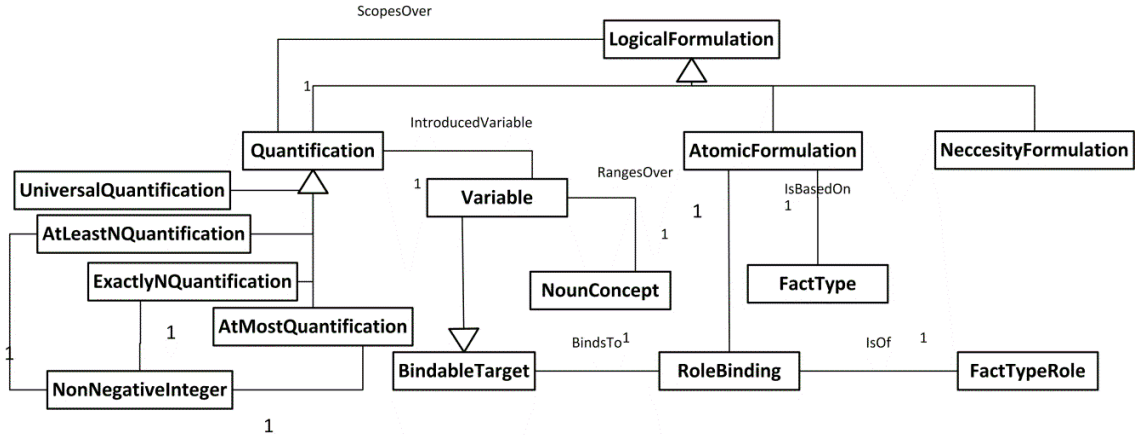

Figure 7c. Reduced Business Rules Metamodel of SBVR

SBVR allows business rules to be written by the business experts and for the business experts regardless of IT. However SBVR does not mention how these business rules can be enforced. Simple and durable rules can easily be converted into a database model and OCL constraints. But more complex and volatile rules cannot be hardcoded and thus they need another approach. How complex and volatile business rules can be translated into a uniform event mechanism is analyzed in (De Roover,W. & Vanthienen,J., 2010), such that the event handling could provide an integrated enforcement of business rules, providing a pattern mechanism to transform SBVR rules into event-driven enforcement rules.

The most related to database persistence layer is the vocabulary aspects, where describe that the performer of an activity can perform particular manipulations (addition, removal or update) of business facts. These state transitions can be constrained by integrity constraints, derivation rules, or reaction rules.

PIM: An extension of PRR

OMG has not yet included the ECA rules metamodel between their proposals. But production rules (PRR-Production Rule Representation) have been included, whose definition is related to ECA rules. A production rule is a statement of programming logic that specifies the execution of one or more actions in the case that its conditions are satisfied. Between the future extensions that the OMG wants deal with, it is possible to find:

- To include Event-Condition-Action (ECA) rules.
- Perform transformations between PRR and other MDA models such as SBVR.

Both proposals are exactly what we need, for this reason, in order to model ECA rules; we follow the guidelines described to model the PRR. For that purpose it is necessary to include the event aspect in the model (as shown in Figure 8a, as a derived metamodel from the current proposal included in PRR, where production rules are the only kind of computer executable rules that are allowed, but in this sense, there exist RFPs for including ECA Rule support in new versions of PRR.). ECA Rule inherits from Computer Executable Rule like current Production Rule in the PRR MOF Metamodel. In the same sense that PRR is based on a subset of OCL expressions (OMG, 2006) (Cabot J.& Gogolla M., 2012), the metamodel of ECA rule that we propose follows it as well. Although OCL has been used as a specification language for integrity values in databases (Demuth,B., Hußmann ,H., 1999) (Demuth, B. et al, 2001), the main characteristics of triggers, as when they are thrown, are omitted. Following the use of OCL, we propose the association of the event to the context where the triggers need to be validated. When data changes, generally, triggers that are fired for each row contains actions that involves managing *:old.variable* and *:new.variable*. We use of OCL contracts to catch this behaviour of triggers, expressing it with pre and post condition as it is shown in the example of the following section.

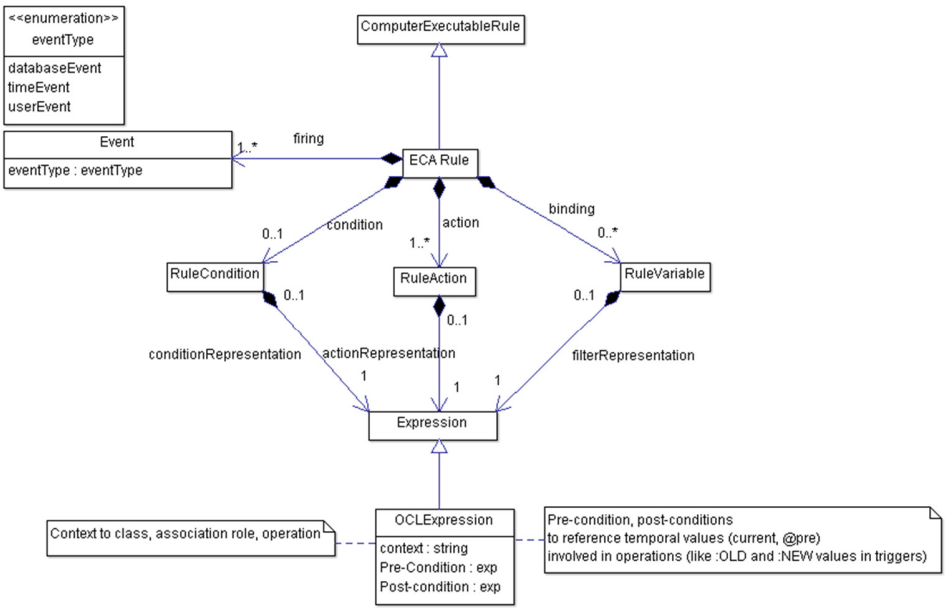

Figure 8a. ECA Rule Metamodel based on PRR Metamodel

For the running example (Figures 6a,6b,6c,6d,6e), the ECA rule model is shown in Figure 8b.

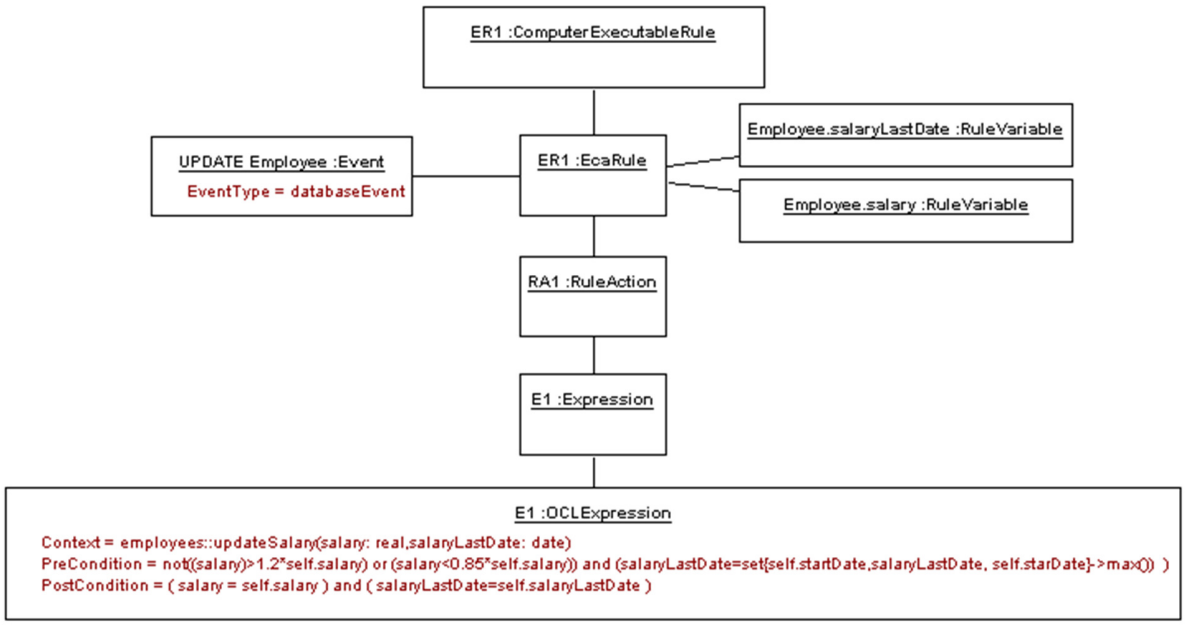

Figure 8b. ECA Rule Model for running example

PSM: An adaptation of SQL:1999

Depending on the database management system that is used, the syntax and other details of the triggers can change. Between the more relevant proposals of languages for triggers we can find PL/SQL by Oracle and Transact-SQL used in SQL Server databases. From SQL:1999 standard, among all SQL sentences, a BNF grammar for triggers was firstly defined. A simplified MOF representation for triggers is shown in Figure 9a. The adaptation of the metamodel of PL/SQL for ASTM Metamodel is presented in (Cánovas Izquierdo J.L., García Molina J., 2010).

To define our proposal of a metamodel for triggers on relational databases we have analyzed existing metamodels for relational artefacts; for example, the general proposal of the OMG: Information Management Metamodel (IMM), although there are more recent and specific works as proposed in (Cánovas Izquierdo J.L., García Molina J., 2010), and (Gra2Mol(2013)) for the Gra2Mol DSL employed in Oracle PL/SQL legacy systems modernization. Based on this latest work, we have taken artefacts related to triggers.

The authors use two packages to specify Abstract Syntax Tree Metamodel (ASTM): a base metamodel called the Generic Abstract Syntax Tree Metamodel (GASTM), with factors common elements of most programming languages, and a metamodel called the Specialized Abstract Syntax Tree Metamodel (SASTM), that represents specific properties of each programming language (in our case: language to write triggers).

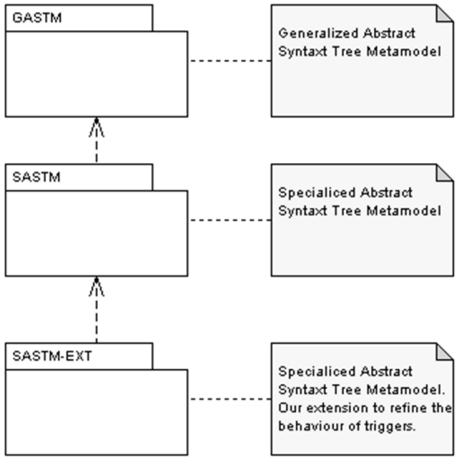

Figure 9a. ECA Rule MetaModel Package Diagram

We select only the classes that are related to the representation of triggers and we have added a new package to extend the SASTM, we call this SASTM-EXT. So classes are displayed with yellow fill pattern colour (classes extracted from GASTM), white fill pattern colour (extracted from SASTM) and new classes and enumerations with blue fill pattern colour (for classes included in our SASTM-EXT).

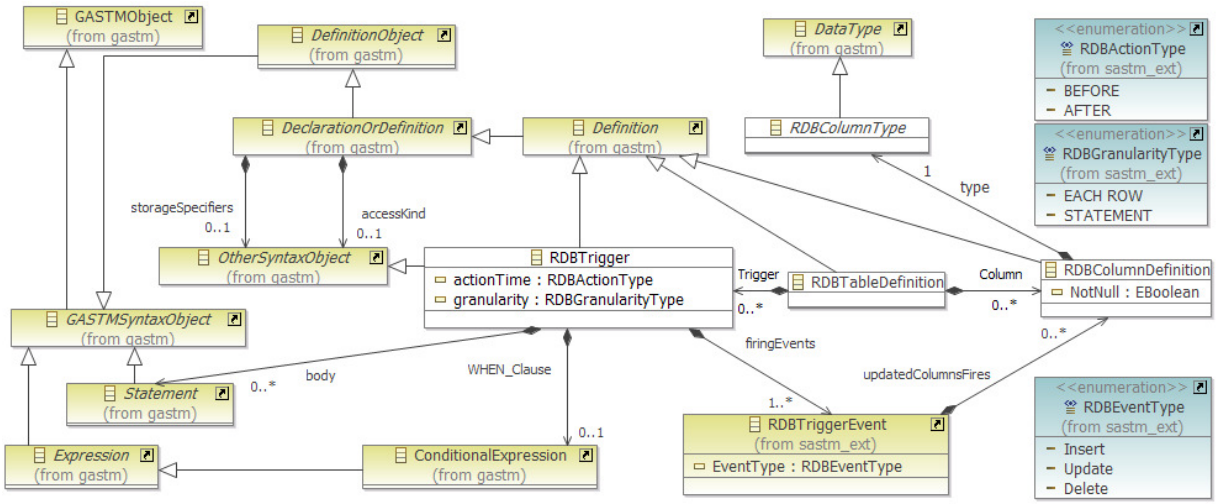

Figure 9b. ECA Rule MetaModel Class Diagram

We also have added attributes *actiontime* and *granularity* to *RDBTrigger* class whose domains are the enumerations *RDBActionType* and *RDBGranularityType* to represent this trigger behavior.

RELATED WORK

Related works can be found in different areas of research. The following subsections analyze the related works in the areas of modernization, metamodelling and model transformation.

Solution of modernization

This subsection analyses those works related to the reverse engineering of databases in the context of the modernization of systems. The analysis of databases in the modernization of systems is an active area. Works as (Pérez-Castillo R. et al.., 2011b), (Pérez-Castillo, R., et al., 2012a) and (Manzón, J. N. & Trujillo, J. 2007) present how it is possible to extract the conceptual model from databases or from the queries embedded in the code of the application. The necessity to extract from the legacy systems the business process knowledge was detected and treated in papers such as (Ricardo Pérez-Castillo, et al, 2011a), (Pérez-Castillo R. et. Al., 2012b) and (Pérez-Castillo, R. et al, 2012c), but neither of them oriented to business rules engine. Others works about modernization oriented to business rules are (Sánchez Ramón O., et al.., 2010) and (Heckel, R. et al., 2008) that use code mining in GUI interfaces for business rules extraction.

Related to business rules and their implementation by means of trigger, to the best of our knowledge, neither solution addresses the whole problem of business rules modernization discovered from trigger representation. Although there are works and standards that try to solve some parts of the problem, helping business experts to understand the business rules hardcoded in the persistence layer, the problem has not been completely analyzed. In the next subsection, some proposals for each part of the modernization process are studied.

Metamodels related to the problem

Model-Drive Architecture (OMG, 2003) defines three levels of abstraction CIM (Computation Independent Model), PIM (Platform Independent Model) and PSM (Platform Specific Model). This subsection analyses the metamodels proposed by other authors that can be used in these different levels of modeling.

- **PSM:** As this paper is focused on inferring business rules implemented in databases by means of triggers, at this level we need a metamodel to describe the constructs used in triggers and their relationship. It deserves to highlight that each database management system has a proprietary language to describe the triggers. In order to be independent of the database management system used, at this level is necessary to define one metamodel that includes the commonalities among the different trigger proprietary languages. Although a proposal as IMM (OMG, 2009a) focuses on the definition of a metamodel for triggers, the complete model of trigger is not proposed. The metamodel which includes the commonalities could be based on the standard SQL:1999 (Türker, C. & Gertz, M.,2001). There is no metamodel for SQL:1999 triggers, however, a BNF grammar for this language can be found in (Türker, C. & Gertz, M.,2001), and also there are bridges between grammar technical space and model technical space (Wimmer, M. & Kramler, G., 2005), (Alanen, M.&Porres, I (2003))
- **PIM:** Triggers are close to the definition of ECA Rules (Event-Condition Action rules), since triggers describe when a modification over a table of the database is done. At this level, the metamodel for production rules (Production Rule Representation -PRR) is proposed in (OMG, 2009b). Production rules are similar to reaction rules. The main difference between them is that ECA rules include events, while Production Rules do not support this concept. OMG plans to

include the ECA rule in its next version of the standard PRR, but currently does not support it. Although there is not a standard, there is other proposal that has tried to model ECA rules (Viana, S., et al., 2007). However, the metamodel is not detailed, and then it cannot be included in the transformation process for the modernization.

- **CIM:** The most used business rules language in CIM level is Semantics of Business Vocabulary and Business Rules Business (SVBR) (OMG, 2008b). This language permits to express different types of rules and is very close to business experts, since it allows the inclusion the concepts and vocabulary widely used by the community of specialists.

Transformations between the levels of abstraction

As it is stated in the previous section, it is important to model the artefact using different levels of abstraction. But it is not less important how the transformation between these levels of abstraction can be performed. There are some proposals that work in this field. These transformations are important from bottom-up and top down.

- **Transformation from PSM to PIM:** There are several works that use reverse engineering in the database area, as (Reus, T. et al., 2006). Also related to databases in (Manzón, J.-N.&Trujillo, J., 2007), a set of heuristics are presented to construct a conceptual model by using as source a relational model. As the body of triggers can contain procedures, also would be interesting to analyse how the code can participate in the modernization process, as was done in (Cánovas Izquierdo J.L & García Molina J., 2009) and (Cánovas Izquierdo J.L & García Molina J., 2012). For the triggers transformation from PSM to CIM, a combination of both disciplines will be necessary.
- **Transformation from PIM to CIM:** As the most used metamodel in the CIM level is SBVR, most of the proposals are centred in how to transform a PIM metamodel into a SBVR metamodel. One of the most important metamodels related to rules is OCL (OMG, 2006), a PIM language that can be associated to the conceptual model. Works as (Cabot, J. et al., 2010) and (Pau, R., &, Cabot, J., 2008) have proposed the transformation between OCL and SBVR.
- **Transformation from CIM to PIM**: In (Sellner, A., et al., 2011) a proposal on how SBVR can be translated to triggers in SQL but for IT Service Management and not using MDA for a generalization is presented. In (Marinos, A. et al., 2010) and (Moschoyiannis, S., et al., 2010), an automatic transformation from SBVR to SQL is proposed but not for triggers specifically, and ignoring the necessity of an intermediate level between SBVR and SQL. In (De Roover, W. & Vanthienen, J., 2010), the necessity of an intermediate level translating the business rules to an uniform event mechanism is analyzed, in such a way that the event handling could provide an integrated enforcement of business rules, but for a reduced set of patterns that follows SBVR. (Kamada, A., et al., 2010) presents another transformation from SBVR to Executable FCL Rules. (Kleiner, M., et al., 2009) propose how to transform a SBVR model extracted from a text and transform it into a UML class diagram.
- **From PIM to PSM:** We have not found any transformation from a PIM model to triggers in PSM.

FUTURE RESEARCH DIRECTIONS

Related to business rules discovery process, only simple types of triggers have been analyzed, where the action of the rule implies the assignation of variables depending on a condition. But it is possible to find in the body of the trigger very complex programs, SQL statement, procedures or function. For future work, we propose to enlarge our proposal with more complex triggers.

Also, in order to complete the modernization process, we think that would be very interesting to define heuristics to divide the implemented rules or combine them in new rules to be more understanding for the business expert. Also, it can be interesting them combine the rules inferred from the triggers with other types of rules of data implemented in the applications. We are also plan to analyze how to integrate our proposal in the Modisco framework (MODISCO) for testing real use cases.

CONCLUSION

In order to obtain a business process model from a legacy system, not only the activities that form the model are important, the business rules associated to the process need to be also extracted. Sometimes, these rules can be implemented by means of triggers. In order to transform the triggers hardcoded in an understanding language, in this work we propose an architecture based on ADM. The metamodels proposed are the use of SBVR metamodel, the extension of PPR metamodel for ECA rules, and the trigger metamodel adapted from SQL:1999 standard.

ACKNOWLEDGMENTS

This work has been partially funded by the Ministry of Science and Technology of Spain in the projects TIN2009-13714, TIN2010-20057-C03-02, TIN2010-21744-C02-1, and the European Regional Development Fund (ERDF/FEDER).

REFERENCES

Alanen, M., Porres, I (2003): *A Relation Between Context-Free Grammars and Meta-Object Facility Metamodels*. Technical report, Turku Centre for Computer Science, 2003

Baisley, D. (2005): *OMG and Business Rules*. Presentation available at ttp://www.omg.org/docs/omg/05-04-09.pdf.

Bisbal J., Lawless D., Wu B., Grimson J.(1999): Legacy Information Systems. Issues and Directions. IEEE Software (SOFTWARE) 16(5):103-111

Bruneliere, H., Cabot, J., Jouault, F., Madiot F. (2010). *Modisco: a generic and extensible framework for model driven reverse engineering*. In Proceedings of the IEEE/ACM international conference on Automated software engineering, ASE '10, pages 173–174, New York, NY, USA, 2010. ACM.

Cabot, J. & Gogolla,M. (2012): *Object Constraint Language (OCL): A Definitive Guide.* SFM pages:58-90

Cabot, J., Pau,R., Raventós,R. (2010): *From UML/OCL to SBVR specifications: A challenging transformation.* Inf. Syst. (IS) 35(4):417-440

Cánovas Izquierdo J.L., García Molina J. (2009): *A Domain Specific Language for Extracting Models in Software Modernization.* ECMDA-FA pages:82-97

Cánovas Izquierdo J.L., García Molina J. (2010): *An Architecture-Driven Modernization Tool for Calculating Metrics*. *IEEE Software Journal* Vol. 27, num 4, pages:37-43

Cánovas Izquierdo J.L., García Molina J. (2012): *Extracting Models from Source Code in Software Modernization*, Software & Systems Modeling

Chikofsky, E., Cross, J. (1990) Reverse engineering and design recovery: A taxonomy, IEEE Software, 7(1):13-17

De Roover,W. , Vanthienen,J. (2010): *A Transformation from SBVR Business Rules into Event Coordinated Rules by Means of SBVR Patterns,* ServiceWave Workshops: 172-179

Deltombe, G. , Le Goaer, O., Barbie, F. (2012): *Bridging KDM and ASTM for Model-Driven Software Modernization,* SEKE'12.

Demuth,B., Hußmann ,H.(1999): *Using UML/OCL Constraints for Relational Database Design.* UML pages:598-613

Demuth, B., Hußmann,H., Loecher,S. (2001): *OCL as a Specification Language for Business Rules in Database Applications*. UML pages:104-117

Favre, L. (2010): . *Model Driven Architecture for Reverse Engineering Technologies: Strategic Directions and System Evolution.* Premier Reference Source. IGI Global,

Goedertier, S., Haesen, R., Vanthienen,J. (2007): *EM-BrA2CE v0.1: A vocabulary and execution model for declarative business process modeling,* Open Access publications from Katholieke Universiteit Leuven.

Heckel, R., et al (2008): *Architectural Transformations: From Legacy to Three-Tier and Services.* Software Evolution, pages:139-170

Jablonski, S., Bussler, C.(1996): *Work flow Management. Modeling Concepts, Architecture and Implementation.* International Thomson Computer Press, London (1996)

Kamada, A., Governatori, G., Sadiq (2010), S.: *Transformation of SBVR Compliant Business Rules to Executable FCL Rules.*

Kazman R, Woods SG, Carrière SJ (1998): *Requirements for Integrating Software Architecture and Reengineering Models: CORUM II*. In Proceedings of the Working Conference on Reverse Engineering (WCRE'98). IEEE Computer Society.

Kleiner, M., Albert, P., Bezivin, J.(2009) : *Parsing SBVR-based controlled languages*. Model Driven Engineering Languages and Systems (2009) 122-136

Gra2Mol (2013): http://code.google.com/a/eclipselabs.org/p/gra2mol/, accesed 22/5/2013

Manzón, J.-N. & Trujillo, J. (2007) *Model-driven reverse engineering for data warehouse design*. JISBD 2007.

Marinos, A., Krause, P. (2009): *An SBVR Framework for RESTful Web Applications*, Springer-Verlag Berlin.

Marinos,A., Moschoyiannis, S., Krause, P. (2010): *An SBVR to SQL compiler, RuleML.*

Marinos,A., Moschoyiannis, S., Krause, P. (2010): *Generating SQL Queries from SBVR Rules.* RuleML:128-143

Modisco. Eclipse: Modisco, http://www.eclipse.org/MoDisco/

OMG (2003): *Model-Driven Architecture (MDA) Version 1.0,* http://www.omg.org/mda/presentations.htm.

OMG (2006): *Object Constraint Language (OCL) 2.0,* < http://www.omg.org/spec/OCL/2.0/PDF/>.

OMG (2006): *Meta Object Facility (MOF) 2.0 Query/View/Transformation Specification(QVT),* <http://www.omg.org/spec/QVT/1.0/PDF>.

OMG (2008b): *SBVR Semantics Of Business Vocabulary And Business Rules. Version 1.0,* <http://www.omg.org/spec/SBVR/1.0/>.

OMG (2009a): *Information Management Metamodel (IMM) Specification V.8.0.„* http://www.omg.org

OMG (2009b): *Production Rule Representation (PRR) Version 1.0,* <http://www.omg.org/spec/PRR/1.0/PDF/>, OMG.

OMG (2010): *Architecture-Driven Modernization (ADM).* http://adm.omg.org/

OMG (2011a). *Knowledge Discovery Meta-Model (*KDM) V. 1.3, http://www.omg.org/spec/KDM/1.3/PDF/.

OMG (2011b): *Business Process Model and Notation (BPMN) Version 2.0,* ttp://www.omg.org/spec/BPMN/2.0/

OMG (2011c): *Syntax tree metamodel.* v. 1.0. http://www.omg.org/spec/ASTM/1.0

Pau,R., Cabot,J. (2008): *Paraphrasing OCL expressions with SBVR*. 13th International Conference on Applications of Natural Language to Information Systems (NLDB'08), LNCS 5039, pp. 311-316

Pérez-Castillo,R. et al (2011a): *Business process archeology using MARBLE*, Information and Software Technology 53 (2011) p.1023–1044

Pérez-Castillo, R., García-Rodríguez de Guzmán, I., Piattini,M. (2011b): *Knowledge Discovery Metamodel-ISO/IEC 19506: A standard to modernize legacy systems*, Computer Standards & Interfaces 33 pages 519–532

Pérez-Castillo R., García I., Caivano D., Piattini M. (2012a): *Database Schema Elicitation to Modernize Relational Databases*. ICEIS.

Pérez-Castillo, R., García-Rodríguez de Guzmán, I., Caballero, I., Piattini,M. (2012b): *Software modernization by recovering Web services from legacy databases*, J. Softw. Maint. Evol.: Res. Pract.

Pérez-Castillo,R. et al (2012c): *A family of case studies on business process mining using MARBLE.* Journal of Systems and Software (JSS) 85(6):1370-1385

Reus T., Geers H., Van Deursen A. (2006): *Harvesting Software Systems for MDA-Based Reengineering.* ECMDA-FA pages:213-225

Roman, D., Keller, U., Lausen, H., de Bruijn, J., Lara, R., Stollberg, M., Polleres, A., Feier, C., Bussler, C., and Fensel, D. (2005). Web service modeling ontology. Applied Ontology, 1(1):77–106.

Ross, Ronal G (2003). The Business Rule Approach, IEEE Computer, vol. 36, num. 5, pages: 85-87.

Sánchez Ramón O., Sánchez Cuadrado J., García Molina J. (2010): Model-driven reverse engineering of legacy graphical user interfaces. ASE pages: 147-150

Sellner,A., Schwarz,C., Zinser,E. (2011): *Semantic-ontological combination of Business Rules and Business Processes in IT Service Management,* CoRR abs/1107.2090

Skersys,T., Tutkute, L., Butleris, R., Butkiené, R. (2012a): *Extending BPMN Business Process Model with SBVR Business Vocabulary and Rules*, ISSN 1392 – 124X INFORMATION TECHNOLOGY AND CONTROL

Skersys,T., Tutkute, L., Butleris, R., Butkiené, R. (2012b) : *The Enrichment of BPMN Business Process Model with SBVR Business Vocabulary and Rules.* Journal of Computing and Information Technology,CIT 20, (2012), p.143–150

Stavru S., Krasteva I., Ilieva S.(2013): *Challenges of Model-Driven Modernization: An Agile Perspective*, MODELSWARD_2013.

Türker,C., Gertz,M. (2001): *Semantic integrity support in SQL: 1999 and commercial (object-) relational database management systems,* VLDB J. (VLDB) 10(4):241-269

Viana,S., Rady de Almeida Jr.,J., Pavón, J. (2007): *A Rule Repository for Active Database Systems.* CLEI Electron. J. (CLEIEJ) 10(2)

Wagner, G. (2003): *The agent-object-relationship metamodel: towards a unified view of state and behavior*. Inf. Syst., 28(5):475–504.

Wagner, G. (2005): *Rule modeling and markup*. Proceedings of the First international conference on Reasoning Web 2005: 251–274. Springer-Verlag.

Wimmer, M. & Kramler, G. (2005): Bridging grammarware and modelware. In: J.-M. Bruel (Ed.): MoDELS 2005 Workshops, LNCS 3844, pp. 159–168, 2006. Springer-Verlag.

AUTHOR'S BIOGRAPHY

Carlos Arévalo is an Industrial Engineer from the University of Seville. He is a Lecturer in the Languages and Computer Science Department of Seville University. His research area is oriented to business object model and web services semantics.

María Teresa Gómez-López received her degree in Computer Science from the University of Seville in 2001 and her Ph.D. degree in 2007. She is currently a Ph.D. Lecturer in the Department of Languages and Computer Systems at the University of Seville, Spain. She worked as a researcher in the Spanish National Research Council. She has participated in Spanish research projects, and technology transfer projects, publishing numerous articles in Computer Science journals and international conferences. Currently, her main research interests are Business Compliance Rules, Business process management and Constraint Databases.

Antonia M. Reina Quintero works as a full-time Lecturer at the Languages and Computer Systems Department of the University of Seville since 2000 and received her Phd degree from this University. She also has worked as a computer engineer for a leading company in traffic control systems. Currently she is a member of for the Research Group on Distributed Systems. Her research interest focus on aspect-oriented programming, advanced separation of concerns and Model-Driven Engineering applied to web-based systems.

Isabel Ramos is doctorate in Industrial Engineer University of Seville (US). During many years she has lead the Foundation for Research and Development of Information Technologies in Andalusia (FIDETIA) and has been the director of the Department of Languages and Information Systems of the University of Seville. She also directs various research projects in the private and public arena. The research lines in which she is currently working on are centered in management and process improvements, in the application of testing techniques in the development of software processes improvement and in business model. She is author and coauthor of a great number of national and international publications.