Towards the Automatic Extraction of Structural Business Rules from Legacy Databases

Oscar Chaparro, Jairo Aponte
Universidad Nacional de Colombia
Bogotá, Colombia
{ojchaparroa, jhapontem}@unal.edu.co

Fernando Ortega
IT Consultores S.A.S.
Bogotá, Colombia
lfortega@itc.com.co

Andrian Marcus
Wayne State University
Detroit, MI, USA
amarcus@wayne.edu

***Abstract*—One of the most important problems in the evolution of legacy systems is the lack of knowledge about them. Since their documentation is usually outdated, the most reliable source of information for recovering this knowledge is the structure and behavior of the information system itself. In this work, we present an approach for extracting structural business rules from legacy databases. Based on the OMG SBVR standard, we defined a mapping between database and business rules components, implemented the mapping in a rule extraction tool and used it in a financial legacy information system. As a result, we extracted 756 structural business rules. The evaluation of the approach indicates that the proposed mapping is appropriate in terms of the number of real extracted business rules, resulting in around 85% of accuracy.**

***Index Terms*—Business Rules Extraction; Legacy Information Systems; Rule Components; Database Components; SBVR standard**

I. Introduction

Legacy information systems are systems that have had a long evolution, i.e., a long life cycle. Often, they are built with outdated technologies and paradigms [1][2], and have a big size in terms of information processing capacity, functionalities provided, program components, and lines of code. Even though legacy systems are resistant to modifications and integration with other systems [2], they are essential in business[1] [1][4], and have accumulated large amounts of information related to business processes and quality attributes, such as performance in algorithms.

These characteristics have some implications on the evolution of legacy systems. First, the development/maintenance team gradually loses knowledge about the system architecture and behavior, as a result of its long evolution. Second, these software systems require little successive modifications that do not alter their architecture but deteriorate it gradually i.e., the architecture follows a new path regarding the original design. And third, while the system changes over the time, its documentation is not updated [1], and therefore, the knowledge about the system remains only in the people have developed and maintained it, leading to a high probability of loss of information.

[1]*Business* is understood in this work as the "the activity of buying and selling goods and services, or a particular company that does this, or work you do to earn money" [3].

Consequently, the most important problem in the evolution of legacy systems is the lack of knowledge about them. The most faithful source of information for recovering this knowledge is the structure and the behavior of the information system itself; thus, it is relevant to apply knowledge extraction techniques to its source code. In this sense, *Business Rules Extraction* (BRE) is a reverse engineering process for recovering or extracting *Business Rules* (BR) from software information systems. In general, BR are constraints that define and guide the way a business operates [5][6]. BRE is important in software knowledge acquisition because:

- It is a means for software re-documentation [7] and functionality-code tracing [8].
- It builds BR-Code mappings that can support the understanding of the system [9].
- It supports the validation process for checking that the system fulfills its specification [8], i.e., that all the business rules are actually implemented [10].
- It is used in software re-engineering and migration [10].

Although BRE is important for software knowledge acquisition, the research performed in this area has been focused only on the identification of program elements that could lead to business rules (e.g., domain variables, program sentences or program slices) [9][11][12][13][14]. Moreover, the methods or processes for extracting business rules are not reproducible, and in some cases, they do not present clearly the extracted rules regarding the concepts of business rules, their composition and categorization [11][12][14][15][16]. These drawbacks lead to incomplete extraction of rules and large amounts of manual work to achieve a complete extraction and verification [10].

In this work, we present an approach for extracting structural BR from legacy databases. The BRE approach is part of a reverse engineering process applied to SIFI (SIstema Fiduciario Integrado[2]), an industrial legacy system implemented in PL/SQL and Oracle Forms technology. In order to define the BRE approach, we performed a revision of the BR concepts, its characteristics and categorization, based on the Semantics of Business Vocabulary and Business Rules (SBVR) standard [17][18] provided by the Object Management Group (OMG). Then, we analyzed the structural database (DB) components

[2]In English, Fiduciary Integrated System.

and the BR concepts to define a mapping among them. For this, we analyzed a set of database components of a SIFI module in order to find patterns that could lead to structural BR, and at the same time, to validate the mappings. Finally, we defined a BR format, followed the SBVR methodology for the extraction of rules and performed a qualitative evaluation of the approach.

The major contributions of this work are:

- A revision of the BR concepts for BRE based on SBVR standard.
- An approach based on DB-BR component mappings for extracting structural BR from databases.
- The practical application of the approach in an industrial legacy information system.

II. Business Rules Concepts

Business rule is a common term in business and software design. Intuitively, we consider BR as what the business and software does or operates. Although this conception is not wrong, it is quite inaccurate. Actually, BR delimit what is permitted and what is not in business and, therefore, in software. In other words, business rules define or constrain what is allowed in the operation of the business.

A. Definition of Business Rule

A rule is an explicit regulation or principle that governs the conduct or procedures within a particular area of activity, defining what is allowed [5]. A business rule is a rule under business jurisdiction, that is, a rule that is enacted, revised and discontinued by the business [5][18]. For example, the "law" of gravity can affect a particular business, but it is not a business rule because the business cannot govern it; instead, the business may create rules for adaptation or compliance.

Business rules guide the behavior or action of a particular business and serve as criterion for making decisions, as they are used for judging or evaluating a behavior or action. They shape the business (business structure) and constraint processes (behavior of the business), to get the best for the business as a whole [5][19]. This means that business rules must be defined and managed in an appropriate way to guide the business to an optimal state.

B. Structure of Business Rules

The SBVR standard [17][18] defines key concepts on BR:

- Business Vocabulary: is the common vocabulary of a business, built on concepts, terms, and fact types.
- Business Rules: they are statements/sentences based on fact types that guide the structure or operation of a business.
- Semantic Formulation: is a way of structuring the meaning of rules through several logical formulations [18], e.g., logical operators (*and*, *or*, *if-then*, etc.), quantification states (*each*, *at least*, *at most*, etc.), and modal formulations (*It is obligatory* or *It is necessary*).
- Notation: is the language used to write and express BR. SBVR standard uses three reference notations, namely SBVR Structured English, RuleSpeak and Object-Role Modeling.

Business rules are composed of a structured business vocabulary [5], which provides them with meaning and consistency. Structured business vocabulary comprises the following elements [5][19] (Figure 1):

- Noun concepts: they are represented by terms of the business. They are elemental, often countable and non-procedural. For instance, *Country*, *Car Brand*, *Customer*, *Operational Cost*.
- Instances: they are "examples" of noun concepts. Instances are always in the real world, not in a model. For example, *United States*, *Volkswagen*.
- Fact types: they are connections of concepts made by verbs or verb phrases. They give structure to the business vocabulary, recognize a known fact and organize knowledge about results of processes. Common shapes or classes of fact types are categorizations (e.g., *Coupe Automobile is a category of Automobile*), properties (e.g., *Car has Brand*), compositions (e.g., *Computer is composed of CPU, RAM, and Hard disk*) and classifications (e.g., *Volkswagen is classified as Car Brand*). Fact types also can be categorized by arity, which is the number of noun concepts in the fact type.

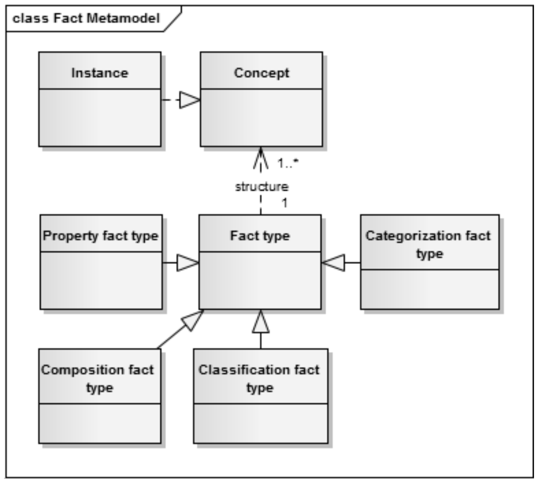

Figure 1. Fact Metamodel.

Through semantic formulations, business rules add a sense of obligation or necessity and remove degrees of freedom to the structured business vocabulary [5] (Figure 2). For example, for the fact type *Car has Engine*, a business rule could be *A Car must have only one Engine*. In this case, the business rule has quantifiers (*A*, *only one*) and an operative modal keyword (*must*) that restricts the fact type. Another form to express the same business rule is: *It is necessary that a Car has exactly one Engine*. In this case, the rule has the structural prefix modal keyword *it is necessary that* and the quantifiers *a* and *exactly one*.

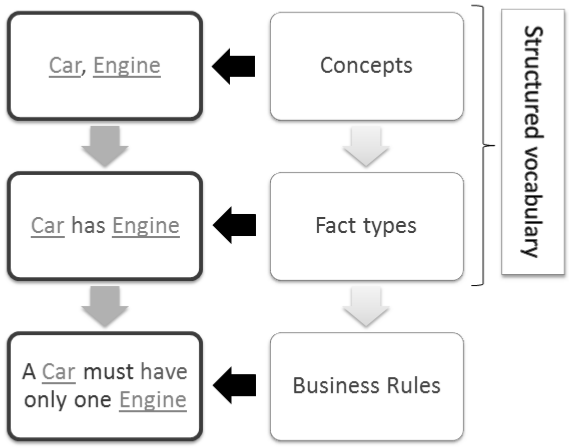

Figure 2. Business rules structure.

Although business rules can be expressed in several forms, it is required to follow only one specific notation, such as SBVR Structured English or RuleSpeak. Even so, and regardless the way business rules are expressed, they must be declarative and non-procedural, i.e., they must define the case/state of knowledge without describing when, where, or how the case/state is achieved [5][19][20].

C. Types of Business Rules

There are two types of business rules [5][17][21][22] (Figure 3):

- Behavioral or operative rules: they govern the behavior or business operations in a suitable and optimal fashion and, therefore, they are important for modeling business processes. These rules always carry the sense of obligation or prohibition, can be violated directly, and not all are automatable. Examples of this kind or rules are the following [5]:
  - Not automatable: *"A Nurse must visit a Patient at least every 1 hour"*.
  - Automatable: *"An Order over $10000 must be accepted on Credit without a Credit Check"*.
- Definitional or structural rules: they structure and organize basic business knowledge. They carry the sense of necessity or impossibility and cannot be violated directly [5]. Unlike behavioral rules, not all definitional rules are business rules (e.g., law of gravity or rules of math); however, all of them are automatable [5]. When evaluating structural rules, usually there are two possibilities: classifications (class membership) and computations (results). Some examples are:
  - Classification: *"A Customer is always a Premium Customer if the Customer has placed Orders of more than $10000"*.
  - Computation: *"The Total Price of an Order is always computed as the Sum of Order Item Prices"*.

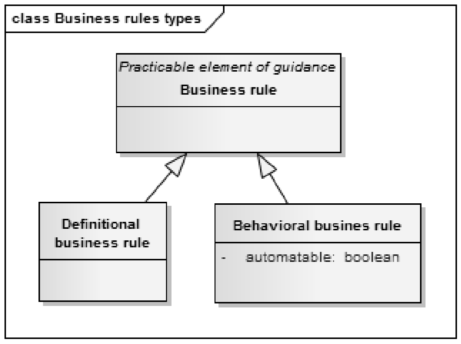
class Business rules types
Practicable element of guidance
Business rule
Definitional business rule
Behavioral busines rule
- automatable: boolean

Figure 3. Types of Business Rules.

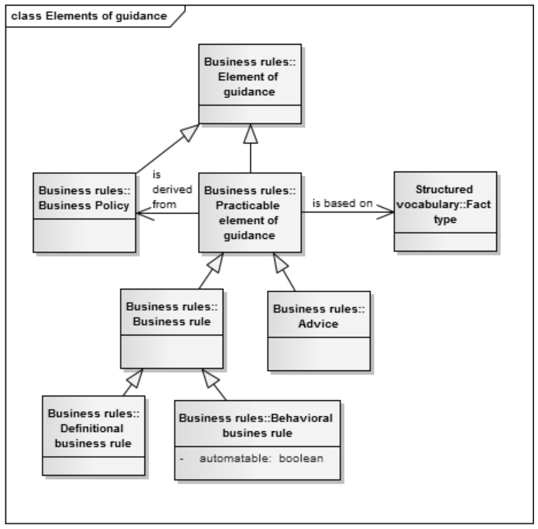
class Elements of guidance
Business rules:: Element of guidance
Business rules:: Business Policy
is derived from
Business rules:: Practicable element of guidance
is based on
Structured vocabulary::Fact type
Business rules:: Business rule
Business rules:: Advice
Business rules:: Definitional business rule
Business rules::Behavioral busines rule
- automatable: boolean

Figure 4. Element of guidance metamodel [5].

D. What are not Business Rules?

Business rules, business policies, and advices are elements of guidance (guidelines) (Figure 4). Then, what is the difference between them? why are all of them not considered as business rules? First of all, business rules must be practicable elements of guidance. Although business policies are guidelines, they are not practicable, so they are not business rules [5]. For example, the sentence *"Compliance to the Customer is our Priority"* is not a BR. Instead, business policies are reduced to practical guidelines, i.e., to business rules or advices. Advices and business rules are practicable guidelines because they are built upon fact types, but advices do not remove any degree of freedom from fact types [5]. In other words, they do not set any obligation or prohibition on business conduct, and any necessity or impossibility for knowledge about business operations [5]. The following is an example of an advice: *"A Customer Claim may be handled by a Personal Assistant"*.

In the same way, the following elements are not business rules:

- Events: they express actions performed by an actor in a specific moment in time. A business rule can be analyzed

to find events where it needs to be evaluated.
- CRUD[3] operations: they are events that always result on data rather than a business rule.

Exceptions are not BR per se but violations to rules. However, they are used to formulate and organize logical and coherent BR. In a business rules context, there are no exceptions; instead, there are well stated business rules [5].

E. Relation of Business Rules and Software Information Systems

What is the role of BR in software information systems? Business rules capture the decision logic needed for activities in business processes [5], and produce multiple events in processes [5]. Typically, software systems model and implement business process, so we can say that business logic and rules are in source code (Figure 5), at least implicitly. For example, the system messages presented, when an operative exception has occurred, have implicit BR, if they provide the users with guiding messages.

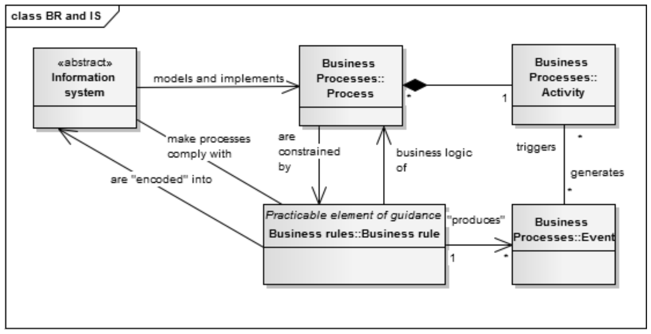
class BR and IS
«abstract» Information system
models and implements
Business Processes:: Process
Business Processes:: Activity
make processes comply with
are constrained by
business logic of
triggers
generates
are "encoded" into
Practicable element of guidance
Business rules::Business rule
"produces"
Business Processes::Event
1
*
1
*
*
*

Figure 5. Relationship between information systems and business rules.

Business rules are key components within the structure of business processes and software models [5][21]. They evaluate facts in a process and can control the basis for changes in the flows of processes in which decisions appear. At business level, business rules enable the business to make consistent operative decisions, to coordinate processes, and to apply specialized know-how in the context of some product/service [5]. At software level, business rules guide flows in procedures and constraints the facts allowed. In any case, if a business rule changes, the business processes and the software modules associated to that rule need to be changed [10], in order to continue the compliance with the business rules [8].

III. EXTRACTION OF STRUCTURAL BUSINESS RULES

Business rules rely on fact types, and fact types on business vocabulary. For the automatic extraction of structural BR from database components we followed the same reasoning. First, we extracted business concepts; then, we created fact types by relating the concepts through verb phrases and; finally, we generated sentences and business rules by adding quantifiers, modal and logical keywords to fact types. In this process, we tried to map every structural database component, including tables, columns, constraints and comments, to the business rules structures, i.e., concepts, verb phrases, fact types and rules. We proposed basic heuristics for BRE and also a business rule DB model, which represents the format of the extracted rules.

We also analyzed the portion of the database that corresponds to a specific module from the studied information system. This allowed us to refine the heuristics proposed and evaluate the extraction method. Finally, we developed a tool prototype that performs the automatic extraction of business rules.

A. Software Studied

SIFI is a financial information system that manages the information and the operations related to investments funds, financial investments, trusts, accounting, budget, treasury, accounts payable, billing and commissions portfolio. The system is maintained by a Colombian software development company, called IT Consultores, has a life cycle of more than fifteen years operating in several trust companies (banking companies), and currently, is deployed and used in nine large Colombian trust companies that represent the 60% of this market in the country. The system is developed in Oracle Forms technology, it has 1400 form modules, 3200 database tables, 3500 stored procedures, 700 database views, and around 800 KLOC in the database.

IT Consultores has no documentation of the BR involved in the SIFI's business processes. The knowledge about the system has remained in two groups of people: the technical group, who knows about SIFI's architecture, and the business group, who knows about the business processes supported by SIFI. The problem is that when people leave the company, a loss of knowledge about the system and the business occurs. Therefore, the client support, the maintenance and, in general, the evolution of the system is negatively affected. To mitigate this problem the company has started a reverse engineering process in which the extraction of BR is one of the most relevant steps.

Because of the large size of the system, we decided to work only on the *Programming and Payment* (PP) module, which is one of the largest modules of the system and is used by almost all of the other SIFI modules. Likewise, the module handles common tasks in many processes of a trust company: trust taxes payments, contracts and invoice payments, investments discharges and, in general, payments to trust suppliers. We think the module has a considerable number of BR.

B. DB - BR Mappings

We identified a set of database components that could lead to BR, and defined a DB-BR components mapping using basic heuristics. For this, we analyzed the DB core of the PP module, composed of 25 tables, and tried to consider standard DB elements, so that the heuristics could work in other systems.

[3]Create, Retrieve, Update and Delete.

Each of the following structural database components were considered[4]:

- Tables: they often represent noun concepts of a domain. For instance, the table called *FD_TMOVI* stores the movements of trusts payments, so the concept that represents the table is *Trust Payment Movement*. Nevertheless, not all tables represent concepts. There are tables that only relate two or more tables (many-to-many relationships), resulting in those that represent verb phrases. For example, the table called *FD_TMVCR* represents the relationship *Rejection Cause per Trust Payment Movement.* This table only relates the tables *FD_TCSRZ (Movement Rejection Cause)* and *FD_TMOVI*, thus representing a relationship instead of a concept. Actually, the table expresses the verb *has* and the fact type *Trust Payment Movement has Rejection Cause.*
- Table columns: they also represent noun concepts and fact types. Concepts are extracted from their comments and fact types from the relationship between their concepts and the concept of the table they belong to. Initially, the fact types that can be created with column concepts are those of class property. For example, the column *MOVI_ESTADO*, which belongs to the table *FD_TMOVI*, has the concept *State*, so the property fact type extracted from the column and the table is *Trust Payment Movement has State.*
- Foreign keys/constraints: these constraints represent verbs between table concepts. In some cases, the fact type class is different from property, e.g., the table *FD_TMOVI* has a foreign key that references the table called *FD_TOPER* (which has the concept *Trust Movement Operation*); therefore, the fact type that represents the foreign key is *Trust Payment Movement generates Trust Movement Operation*.
- Primary and unique keys/constraints: these constraints are used only to ensure uniqueness and identification of data. Therefore they do not map to any concept of BR.
- Table/column comments: comments are descriptions in natural language about tables and columns in databases. They can be used to extract noun concepts and fact types related to tables and columns. In the worst case, such descriptions are not available in the database, so concepts and fact types must be manually assigned or extracted from other sources, such as labels in the presentation layer of the information system. In the case of SIFI, it was possible to automatically identify the concepts of tables and columns from their comments, through a Part-Of-Speech tagger called TreeTagger[5]. By analyzing DB comments of the PP module, we realized that comments often contain more than one noun or composed nouns; thus, for columns, we join nouns to create a concept, while for tables, we look for the table nouns in the column comments for establishing the most important ones. For example, the identified nouns of the table *FD_TMOVI*, which comment is *"Are the payments, fundraisings or causations of money of a trust or an asset in a trust"*, are *Payment*, *Movement*, *Fundraising*, *Causation, Money, Asset* and *Trust*. To find and build the real concept of the table, an importance weight is assigned to each noun of the list, by computing their occurrences in the column comments. In this case, the words *Payment*, *Movement* and *Trust* are the nouns that appear the most in the column comments and, thus, the generated table concept is *Trust Payment Movement.* On the other hand, for creating fact types, on each column comment we expect to find verbs that associate concepts. When comments only contain nouns, the verb extracted is the one used for property fact types, i.e., the verb *has*. In contrast, when comments contain verbs, they are identified with the POS tagger and used to create fact types. For example, for the fact type *Trust Payment Movement generates Trust Movement Operation*, the verb *generate* is extracted from the comment of the column *MOVI_OPER* (table *FD_TMOVI*).
  In addition, the comments also are useful to identify other common classes different from property fact types, using check constraints.
- Check constraints: they are conditions that always should be satisfied when adding and updating data in tables. This means they cannot be violated and, therefore, they represent structural BR. We found three types of check constraints in the PP module: not null constraints, check list constraints and others.
  Not null constraints verify that columns always have not null values. For example, the table *FD_TMOVI* has a constraint that checks that column *MOVI_ESTADO* has not null values. Using this constraint, the BR that can be created is *A Trust Payment Movement always has a State*, which relies on the fact type *Trust Payment Movement has State*.
  Check list constraints verify that column values always are in a list of values. There are three possibilities regarding the meaning of values: values meaning classes of concepts, states of concepts or operative parameters. Regarding the first type of values, the nouns corresponding to each value are used to create categorization fact types. For example, the table called *GE_TFORMULA* (*Formula of Movement Concept*) has a column called *FORM_CLASE* (*Type of Formula*), and a check list constraint with the code

```
FORM_CLASE IN ( 'CD', 'SD' )
```

  The value *'CD'* corresponds to the concept *Discount Formula* and the value *'SD'* to the concept *Non-discount Formula*. Both concepts are extracted from the nouns of the comments and represent categories of the table concept. Then, the created categorization fact types are: *Discount Formula is a category of Formula of Movement*

[4]All the given examples in this and the next sections are taken from the PP module.

[5]TreeTagger can be found at http://www.ims.uni-stuttgart.de/projekte/corplex/TreeTagger/.

Concept, and *Non-discount Formula is a category of Formula of Movement Concept*.

Regarding the values that mean states of concepts, the words corresponding to each value are verbs in past participle or, in some cases, adjectives. For instance, the column *MOVI_ESTADO* (table *FD_TMOVI*) is used in the check constraint's code

```
MOVI_ESTADO IN ('A','I','T','P','X')
```

in which the meaning of the value 'A' is *authorized*, 'I' is *inserted*, and 'X' is *canceled*, to name just a few values. For this type of check list constraints, the values are used to build unary fact types: *Trust Payment Movement is authorized* or *Trust Payment Movement is canceled.*

Regarding values that mean operative parameters, we found in the PP module that they are used for guiding the operations and processes in the system. For example, the column *FORM_IMP_MUNIC* (table *GE_TFORMULA*), with the concept *City Tax,* is used in the check constraint's code

```
FORM_IMP_MUNIC IN ( 'S', 'N' )
```

where there are two values: *'S'* (Yes) and *'N'* (No). The column comment indicates an operative condition that means if the formula should or should not consider *City Taxes*.

Finally, other constraints are those that involve other logical conditions that always should be satisfied. For example, the column called *MOVI_VLRMOV* (table *FD_TMOVI*), which is the *Movement Value*, is used in the check constraint's code

```
MOVI_VLRMOV >= 0
```

which means that the value cannot be negative.

The summary of the DB-BR component mappings is shown in Figure 6.

C. BR Format and Scheme

Based on SBVR Structured English and RuleSpeak, we defined a BR format, which was implemented in a database scheme. The scheme represents the basis for the development of an automated General Rulebook System (GRBS) [5], that allows to store and track all the knowledge around the BR, including concepts, fact types, rules and their relationships. The general goal of a GRBS is to provide a means for a smart governance and a corporate memory though traceability [5]. The implemented scheme pursues the same goal.

The scheme is composed of eight database tables (Figure 7). Seven of them model the BR concepts and the other one, the table *bs_t_object_concept*, models the links between the DB components and noun concepts. As a result, the BR scheme not only models and formats BR but also tracks the relationships between DB and BR components.

In the scheme, concepts and verbs are terms used to compose fact types (Figure 7). In turn, fact types are complemented

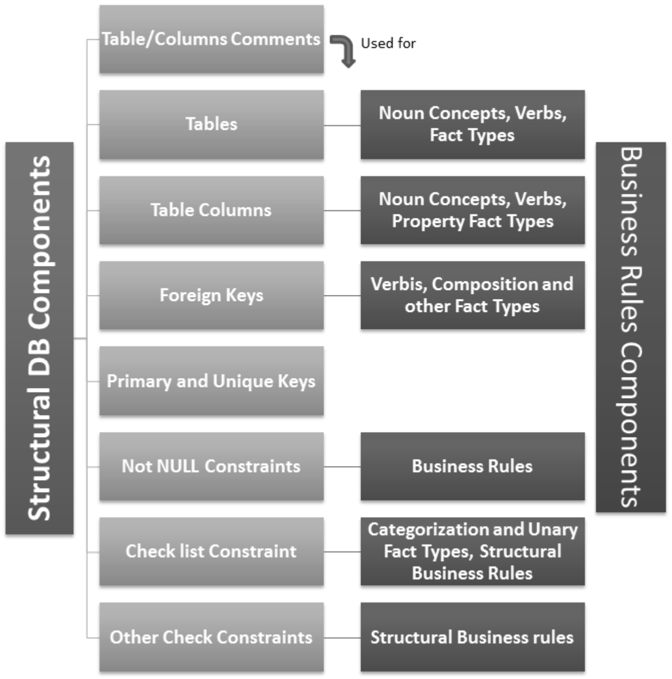
Structural DB Components
Table/Columns Comments
Used for
Tables
Table Columns
Foreign Keys
Primary and Unique Keys
Not NULL Constraints
Check list Constraint
Other Check Constraints
Noun Concepts, Verbs, Fact Types
Noun Concepts, Verbs, Property Fact Types
Verbis, Composition and other Fact Types
Business Rules
Categorization and Unary Fact Types, Structural Business Rules
Structural Business rules
Business Rules Components

Figure 6. DB - BR component mappings.

with quantifiers and keywords to compose sentences; and finally, the combination of sentences and keywords, compose BR. Examples of quantifiers are the following: *every*, *at least*, *maximum*, *exactly*, *more than one*, and *between*. Keywords are logical (*if, only if, and, or*, etc.), operative (*must, can*) or structural (*always, never*).

A sentence is composed of two quantifiers, one per fact type concept, and a keyword. For instance, the sentence *A Trust Payment Movement always has a State* has the quantifier *"a"* for both concepts and the structural keyword *"always*", which gives the sense of impossibility to the fact type *(Trust Payment Movement has State*). In turn, a BR is a sentence or a composition of two sentences joined together with another keyword. For example, a BR composed of two sentences is: *A Trust Payment Movement only is authorized if the Movement has no Rejection Causes.*

There are two important design elements of the scheme: a fact type can be composed by one or two concepts, having unary and binary fact types only; and in the same way, a business rule is exclusively composed of one or two sentences. These design decisions were based on the BR reduction principle expressed in [5]. The principle aims at producing easily-understood and granular rules that can be independently managed, re-used and modified.

IV. Results and Discussion

For extracting structural BR, an extraction tool, included in a Reverse Engineering System for Oracle Forms application, was developed. The tool is the product where the DB-BR mappings, presented in the previous section, were implemented, and is able to extract:

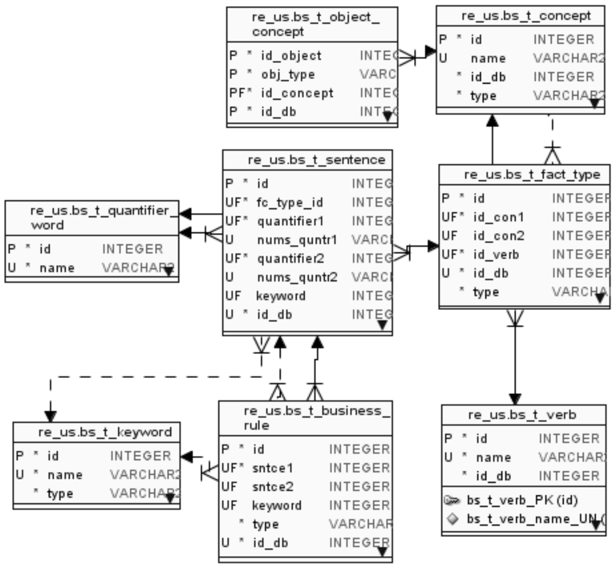

Figure 7. Implemented BR database model.

- Concepts from tables and columns, using the comments in the DB.
- Verbs and binary property fact types from the table/-column comments and concepts, and from the tables hierarchy (foreign keys).
- Categorization and unary fact types from check list constraints.
- Structural BR from the extracted fact types, using the not null constraints.

The input of the tool is a list of DB tables and the output is a set of extracted and stored concepts, verbs, fact types, sentences and business rules. The tool processes the DB components of the input tables and the tables related to them through the foreign keys (parent and children tables). Table I presents some examples of the resulting components. All the BR and the extracted components can be downloaded from http://itc.com.co/CLEI12/[6].

BR component	**Examples**
Concept	*Trust Payment Movement, Operation Check*
Unary Fact Type	*Treasury Movement Type is tax-exempt*
Binary Fact Type	*Expenditure per Contribution generates Treasury Movement*
Property Fact Type	*Credit Note has Specific Voucher Type*
Categorization Fact Type	*Fixed Investment Depreciation is a category of Investment Depreciation*
Structural Business Rule	*Every Deposit Income always has a Movement Value*

Table I
EXAMPLES OF THE EXTRACTED BR COMPONENTS FROM THE PP MODULE.

In turn, table II presents some statistics about the automatic extraction of structural BR, performed in the PP module. The 25 core tables of the module were the input to the tool, resulting in 155 processed tables and 3142 columns, for a total of 3297 processed objects. The number of extracted concepts from tables and columns was 2238, 1893 correspond to general concepts, i.e., concepts having categories, and 345 correspond to categorical concepts. In addition, the tool was able to extract 178 verb phrases and a total of 3625 fact types using the extracted concepts, verbs, and table constraints. 385 unary fact types were extracted, of which 345 correspond to categorization fact types. The number of binary fact types was 3240; those extracted from foreign keys having verbs on column comments were 114, and those with no verbs in comments were 177. At the end, 756 sentences and structural BR were generated using not null constraints and the extracted fact model. 21% of fact types were used to create the structural BR.

[6]The dataset is available only in Spanish.

Statistic	**Value**
# of processed objects (tables/columns)	3297
# of processed tables	155
# of processed columns	3142
# of concepts (tables/columns)	2238
# of general concepts	1893
# of categorical concepts	345
# of verbs	178
# of fact types (FT)	3625
# of unary FT	385
# of binary FT	3240
# of property FT	2895
# of categorization FT	345
# of FT from FK (verbs)	114
# of FT from FK (no verbs)	177
# of sentences	756
# structural BR	756
% of BR from # of FT	21

Table II
STATISTICS OF THE EXTRACTED BR COMPONENTS FROM THE PP MODULE.

We performed an informal revision of the results by manually assessing the extracted BR components, with the support of some technical and business experts that know the PP module. The conclusions were that the proposed DB-BR mappings are appropriate and around 85% of the obtained results correspond to the reality of the business. In other words, DB-BR mappings allowed us to obtain 640 true structural business rules. On the other hand, a number of problems detected led to not obtaining better results, and in around 15% of cases the extracted rules do not correspond to real rules. In any case, we followed some strategies to mitigate the problems. We explain them in detail in the next paragraphs.

Preliminary results allowed us to identify BD-BR component relationships and some technical patterns in the PP module that affect the results. In the first place, the tool was able to store the relationships between BD and BR components. In this way, it is possible to know the DB objects that correspond to concepts or BR, and vice versa, the concepts or rules that correspond to a DB object. For example, the structural BR *Every Deposit Income always has a Movement*

Value is related to the table *TE_TCNSG* and the columns *CNSG_VALOR*, *MOVI_VALOR* and *RLMV_VALOR*. Likewise, the tool allowed us to detect columns that represent operative conditions and non-structural rules. Three conditions were necessary to detect and avoid those columns: the presence of parametric values in check list constraints, the presence of the words *IF* or *WHEN* in the column comments, and the acceptance of null values in the columns. This allowed us to filter out BR from 874 to 756, thus representing a reduction rate of 13%.

Another pattern found in the module was the presence of the word *WHICH* in the column comments. We noticed that columns having this word, represent elements of business operations. In other words, they must be used to extract operative BR. The same case happened with columns that had values that mean states; generally, they represent completed actions and, therefore, they can be used to create operative BR. In any case, those concepts and fact types were extracted, and we expect to use them for proposing a method for extracting operative rules as future work.

On the other hand, we found some problems that did not allow us to obtain better results. One problem found was the precision in the extracted verbs and concept relationships, in terms of their real meaning. Some columns did not include verbs in their comments so it was not possible to obtain accurate fact types. Another issue is that some categories between concepts, that were detected manually from foreign keys, could not be extracted automatically. For example, the table called *FD_TFIDE*, which represents the concept *Trust*, had a foreign key to the table *GE_TCIAS*, which represents the concept *Company.* This foreign key represents a category of *Company*, resulting in the categorization fact type *Trust is a category of Company*. We did not find an automatic means for extracting these kind of fact types. Apart from that, we found other cases in which tables had the unique role of relating tables (many-to-many relationships), resulting in composition fact types. We neither extracted those fact types.

Another problem found was the inaccuracy of the extracted concepts, related to the poor quality of table/column comments or the inaccuracy of the POS Tagger in the identification of nouns. For example the extracted concept of the table *CP_TORPV* was *Payment Order*, instead of the real concept *Payment Order Liquidation.* In this case, the POS Tagger omitted the noun *Liquidation*.
Finally, another fact we noticed was that some nullable columns in reality may not be nullable. In this case, we assumed the constraints that check the column values are in higher layers of the information system, i.e., in presentation or application logic layers.

In summary, we detected three elements that negatively affect the accuracy of the BRE approach:

1) Poor quality in some table/column comments for the extraction of verbs, concepts, categorization and unary fact types. In the case of categorization and unary fact types, the existence of table/column comments is mandatory. If comments does not exist in the DB, the mapping of all BR would be manual or would be extracted from other sources of information.
2) Incomplete accuracy of the POS tagger, which lead to missing or wrong detected nouns and verbs in comments. In the same way, this lead to incomplete extraction of concepts, verbs and fact types.
3) Deteriorated architecture of the PP module as a result of the long evolution of the system. This is reflected in the number of nullable columns, and the presence of parametric and characteristic columns[7] in the same table. In some cases, this lead to the extraction of false structural BR. However, we try to omit DB components with these characteristics.

V. Related Work

We divided BRE in three categories: manual, heuristic and dynamic. The former refers to conduct manual examination of source code for extracting BR, and the others about performing automatic extraction of rules. Heuristic techniques have focused on static processing of source code, while dynamic techniques emphasize the processing of dynamic artifacts such as execution traces. In general, automatic BRE has only focused on the examination of source code, and on the identification of language structures or program sequences that could lead to business rules. Also, in some cases, the concept of business rules is not clear, and neither their structure and categorization.

A. Manual BRE

Earls *et al.* [10] present a method for manual BRE on legacy code. The authors mention two BRE approaches: program-centric and data-centric. According to them, the former is required, but the latter is cheaper. However, they state that the main problem with data analysis is the presence of historical data, which contributes to extract outdated or incorrect BR. In consequence, the method they propose is program-centric and focuses on locating and classifying error-processing sections in source code and the conditions that lead to those sections. The authors propose an error categorization by criticity (fatal, recoverable and database errors) in order to filter those error-processing sections that are not related to BR violations. The conditions related to the error sections are identified and recorded in a rule extraction proforma and, later, are analyzed to determine whether they refer to business or technical conditions. Those representing business conditions are translated in non-technical terms and recorded in a special repository for BR. At the end, the authors performs a review of the extracted rules with domain experts, considering the following evaluation criteria: reliability, understandability, straightforward and uncomplicated applicability, re-usability, readability and commercial viability. The most important conclusion of this work is that manual BRE requires too much time, especially in large information systems, but is more accurate compared with existing and proposed automatic tools and methods.

[7]Characteristic columns are those representing basic information of the table's entity.

We found [10] as the unique and direct work about manual BRE. However, we found related work presented in [23], in which the authors present an analysis on the impact of human factors in the BRE of legacy systems. The authors state that BRE is "heavily dependent on human interaction and steering", and conclude that the following are some consequences of ignoring human factors in BRE: increased probability of incorrect extracted business rules, less capacity to deal with emergent business needs, increased workload, lower productive output, increased probability of project delays, higher time and material costs, low quality in the extracted rules, tension and stress in people who extract rules.

B. Heuristic BRE

Heuristic BRE consists on automatic extraction techniques that take advantage some common elements of the source code that could compose BR. Elements considered in heuristic BRE are, for example, identifier names, exception raising/handling statements, and control flow structures. One method generally used in heuristic BRE is program slicing, which basically consists on extracting some portions of the code that affect some variables, according to some point of interest in the program [24].

The work presented in [12] focuses on identifying business/-domain variables and the application of generalized program slicing for those variables. The process proposed by the author is the following:

1) Identification of input and output variables/parameters of procedures.
2) Slicing criterion identification, based on three heuristics: input/output statements, dispatcher center statements (IF and SWITCH control statements), and end points of procedures.
3) Code extraction using generalized program slicing.
4) Representation of BR, through code fragments, formula views or input/output dependencies.

Although it is not clear which are the elements that composes BR and how they are extracted from code, there are some important facts to consider from this work. First, input and output variables are important because they belong to data flow interfaces in programs; second, control statements are essential for BRE since they guide the programs' execution flow; and third, the BR representation defines mostly the BR components. In addition, the authors present five requirements that BRE should fulfill: faithful representation of BR, multiple representations and hierarchical abstractions for BR, domain-specific business policies, human-assisted automation, and a maintenance tool. The work presented in [23] also applies program slicing and domain variables identification.

Shekar *et al.* [13] focus on enterprise knowledge discovery from legacy systems. The approach they propose consists of three steps:

1) The generation of detailed descriptions of entities, relationships and business rules. An algorithm performs the extraction of the DB conceptual scheme and the meaning of the entities, attributes and business rules.
2) The mapping of the legacy source schema and domain model elements, through mapping rules.
3) The creation of a wrapper that translates queries from the application domain model to the legacy source schema.

The authors consider more "semantic" source code elements for performing knowledge discovery but it is not clear enough how they extract BR from those. Examples of the elements are: output messages, semantic relationships between variables and table columns, variable usages, assignment and control statements, and variables used in database queries.

In [8], the authors suggest to use error messages, program comments, functions, and control flow structures. The authors present three steps to extract BR, based on SBVR standard: extraction of business vocabulary, creation of rules using vocabulary, and "activity interleaving".

The authors in [15] perform BRE on COBOL legacy systems. They focus on simple COBOL statements that carry business meaning, such as calculations and branching statements. In addition, they use the identifiers and conditions to extract the meaning and context of BR. The format used for BR is <*conditions*> <*actions*>, in which actions are completed if conditions are satisfied.

In [9], the following four program elements that compose business rules are proposed: results, arguments, assignments and conditions. Using program slicing, the authors track all the assignments of data results from calculations, and capture the conditions that trigger the assignments. In this work, meaningful names of variables is mandatory.

The authors in [14] propose the use of information-flow relations between input/output variables, statements and expressions to identify domain variables. The approach they propose is mostly useful on procedural code.

Finally, a loosely related topic is the summarization of code. In [25], the authors propose a technique for summarizing java methods. Through a series of heuristics related to linguistics, syntax standards, and statement categories the authors generate natural language summaries of java methods. Some elements considered in the analysis are: names of methods, variables and classes, data types, return type of methods, and method parameters. The evaluation criteria for the technique were the following: summary precision, content adequacy and conciseness. These attributes were based on the fact that a summary must be specific enough and should not omit important information and contain extraneous data and redundancy.

C. Dynamic BRE

[26] and [16] are examples of this kind of BRE. They present process mining approaches, which are intended to recover decision rules and control flow of systems from logs and execution traces. According to the authors, the main advantage of analyzing dynamic artifacts is that it is possible to detect participants, responsibilities, and concurrent activities in processes [16]. However, logs and other sources of information should comply with certain characteristics that make possible to perform process mining.

The research in dynamic BRE has not been as strong as the research performed in heuristic BRE.

VI. Conclusions and Future Work

In this paper, we presented an approach for automatically extracting structural business rules from legacy databases. Specifically, we performed a revision of the BR concepts based on SBVR standard, proposed an approach that considers DB-BR component mappings to extract structural BR from databases and, finally, applied the approach in an industrial legacy information system. We performed a preliminary assessment of the extracted components and rules, which envision promising results in terms of the number of real extracted business rules. In the same way, the assessment allowed us to detect some problems and elements to be considered for improving the extraction of rules components.

As future work, we plan to implement and refine all the proposed DB-BR mappings. The refinement will include:

- The analysis of temporary tables and check constraints with any condition.
- The detection of roles in fact types [5], verb phrases from tables that only relate other tables (many-to-many relationship), and concept synonyms.
- The extension of the BR format/scheme, in order to represent several type of rules in other forms, e.g., decision tables as a way for representing parallel business rules [5].

We also plan to perform a formal qualitative/quantitative evaluation of the approach in terms of precision and recall, and finally, we will move forward the extraction of operative business rules.

Acknowledgments

We wish to thank all technical and business people from IT Consultores that supported and contributed to this work. We also wish to thank all ColSWE research group members for their comments and suggestions in this study.

References

[1] T. Mens, S. Demeyer, J.-L. Hainaut, A. Cleve, J. Henrard, and J.-M. Hick, “Migration of Legacy Information Systems,” in *Software Evolution*, pp. 105–138, Springer Berlin Heidelberg, 2008.

[2] E. Putrycz and A. Kark, “Connecting Legacy Code, Business Rules and Documentation,” in *Rule Representation, Interchange and Reasoning on the Web*, vol. 5321 of *Lecture Notes in Computer Science*, pp. 17–30, Springer Berlin / Heidelberg, 2008.

[3] C. U. Press, “Cambridge Dictionaries Online - British English,” May 2012.

[4] K. H. Bennett and V. T. Rajlich, “Software maintenance and evolution: a roadmap,” in *ICSE ’00: Proceedings of the Conference on The Future of Software Engineering*, (New York, NY, USA), pp. 73–87, ACM, 2000.

[5] R. Ross, *Business Rule Concepts: Getting to the Point of Knowledge*. Business Rule Solutions Inc, third ed., 2009.

[6] W. M. Ulrich, “Knowledge mining: Business rule extraction and reuse.” System Transformation Portal, February 2009.

[7] S. Ali, B. Soh, and J. Lai, “Rule extraction methodology by using XML for business rules documentation,” in *Industrial Informatics, 2005. INDIN ’05. 2005 3rd IEEE International Conference on*, pp. 357 – 361, aug. 2005.

[8] I. Baxter and S. Hendryx, “A standards-based approach to extracting business rules.” OMG’s Architecture Driven Modernization Workshop, 2005.

[9] H. Sneed and K. Erdos, “Extracting business rules from source code,” in *Program Comprehension, 1996, Proceedings., Fourth Workshop on*, pp. 240 –247, mar 1996.

[10] A. Earls, S. Embury, and N. Turner, “A method for the manual extraction of business rules from legacy source code,” *BT Technology Journal*, vol. 20, pp. 127–145, 2002.

[11] X. Wang, J. Sun, X. Yang, Z. He, and S. Maddineni, “Business rules extraction from large legacy systems,” in *Software Maintenance and Reengineering, 2004. CSMR 2004. Proceedings. Eighth European Conference on*, pp. 249 – 258, 2004.

[12] H. Huang, W. Tsai, S. Bhattacharya, X. Chen, Y. Wang, and J. Sun, “Business rule extraction from legacy code,” in *Computer Software and Applications Conference, 1996. COMPSAC ’96., Proceedings of 20th International*, pp. 162 –167, Aug. 1996.

[13] S. Shekar, J. Hammer, M. Schmalz, and O. Topsakal, “Knowledge Extraction in the SEEK Project Part II: Extracting Meaning from Legacy Application Code through Pattern Matching,” tech. rep., University of Florida, 2003.

[14] X. Wang, J. Sun, X. Yang, Z. He, and S. Maddineni, “Application of information-flow relations algorithm on extracting business rules from legacy code,” in *Intelligent Control and Automation, 2004. WCICA 2004. Fifth World Congress on*, vol. 4, pp. 3055 – 3058 Vol.4, june 2004.

[15] E. Putrycz and A. Kark, “Recovering Business Rules from Legacy Source Code for System Modernization,” in *Advances in Rule Interchange and Applications*, vol. 4824 of *Lecture Notes in Computer Science*, pp. 107–118, Springer Berlin / Heidelberg, 2007.

[16] A. Kalsing, G. do Nascimento, C. Iochpe, and L. Thom, “An Incremental Process Mining Approach to Extract Knowledge from Legacy Systems,” in *Enterprise Distributed Object Computing Conference (EDOC), 2010 14th IEEE International*, pp. 79 –88, oct. 2010.

[17] I. Bajwa, B. Bordbar, and M. Lee, “SBVR vs OCL: A comparative analysis of standards,” in *Multitopic Conference (INMIC), 2011 IEEE 14th International*, pp. 261 –266, dec. 2011.

[18] OMG, “Semantics of Business Vocabulary and Business Rules (SBVR), v1.0,” 2008.

[19] B. R. Group, “Defining Business Rules   What Are They Really?,” July 2000.

[20] T. Morgan, *Business Rules and Information Systems: Aligning IT with Business Goals*. Addison-Wesley Professional, 2002.

[21] B. B. I. S. Bajwa, “SBVR Business Rules Generation from Natural Language Specification,” in *AAAI 2011 Spring Symposium - Artificial Intelligence for Business Intelligence (AI4BA)* (AAAI, ed.), vol. 2011 of *Spring Symposium*, pp. 2–8, AAAI, AAAI, March 2011.

[22] A. Agrawal, “Semantics of Business Process Vocabulary and Process Rules and a Visual Editor of SBVR,” Master’s thesis, Indian Institute of Technology Kanpur, 2009.

[23] X. Wang, J. Sun, X. Yang, Z. He, and S. Maddineni, “Human factors in extracting business rules from legacy systems,” in *Systems, Man and Cybernetics, 2004 IEEE International Conference on*, vol. 1, pp. 200 – 205 vol.1, oct. 2004.

[24] B. Xu, J. Qian, X. Zhang, Z. Wu, and L. Chen, “A brief survey of program slicing,” *SIGSOFT Softw. Eng. Notes*, vol. 30, pp. 1–36, March 2005.

[25] G. Sridhara, E. Hill, D. Muppaneni, L. Pollock, and K. Vijay-Shanker, “Towards automatically generating summary comments for Java methods,” in *Proceedings of the IEEE/ACM international conference on Automated software engineering*, ASE ’10, (New York, NY, USA), pp. 43–52, ACM, 2010.

[26] R. Crerie, F. A. Baião, and F. M. Santoro, “Identificacao de regras de negocio utilizando mineracao de processos,” in *Companion Proceedings of the XIV Brazilian Symposium on Multimedia and the Web*, WebMedia ’08, (New York, NY, USA), pp. 241–246, ACM, 2008.