PropertyField Manager – Business Rule Management System

Gessildo Fernando Bengui (77169)
IST – Instituto Superior Técnico, Lisboa, Portugal.
gessildo.bengui@tecnico.ulisboa.pt

Abstract: A Business Rules Management System (BRMS) is a software solution to define, execute, monitor and maintain the decision logic used in a wider business information systems in order to determine tactical actions. In order to provide flexibility regarding the variability and complexity of the business rules, experts have developed great efforts in the analysis, design and implementation of flexible and independents solutions for business rules management and their respective integration with systems already in production. The objective of this work was to analyze, model and implement a BRMS, named PropertyField Manager, in order to support the decision logic used by software solutions provided by Audatex Brazil to their customers and partners. Audatex Brazil has a considerable number of partners and customers in Brazilian financial industry, concretely in Auto Insurance sector, providing solutions for fifteen of the biggest Insurance Companies. In this context, the problem was that, each insurance company has its own set of business rules, it was frequent that every complex change or addition of rules a new release deployment were required. The proposed solution, PropertyField Manager, can execute and infer business rules independently of the information system logic.

Keywords: Business Rules, Business Rules Management, Business Rules Management System.

1 INTRODUCTION

In recent years, the management process of business logic used in enterprise information systems have become subject widely discussed in the community of enterprise Information Systems specialists. This discussion is related to the constant changes in the enterprise business logic and their management in Enterprise Systems already in production. This problem appears in almost all enterprise Systems, from the smallest and simplest, to the largest and most complex. Simply that these systems implement business rules in their source code routines [1].

Nowadays, the solutions to prevent these disaster is the correct Information System Planning and implementation of Business Rules Management Systems. The last one is the main topic of this paper. Actually, there are already some solutions to manage business rules variability and complexity. Each tools with specific approaches concerning to architectural and implementation details.

Considering that Audatex Brazil [8] is a large company and consequently has a considerable number of partners and customers in the financial sector and in Auto Insurance Industry, also lives issues related to management of the variability and complexity of the rules in their enterprise Systems. As a concrete example, we have the case of Drive Platform, which is a software solution for management and budgeting automobiles claims. Besides being used by insurance experts and garages, the platform is provided to fifteen of the largest auto insurance in Brazil. In this context, each insurance company has its own set of business rules which consequently affect their business process, and it is almost frequent that every change or addition of complex rules a new software deployment are required.

The main motivation for this project is the possibility and challenge of reengineering the whole process and mechanism of business rules applying and management for company that provide software solutions to more than fifteen different companies.

The general objective of this work is to design and implement a Business Rules Management System named PropertyField Manager. From this main objective, is expected that the proposed solution reach the following specific goals:

- Identify the main business rules of the client and partners of organization;
- Model and implement a business rules repository,
- Model and implement a Business Rules Engine to inference the rules requested by enterprise System in production.
As results, is expected a functional system to manage the business rules and attend the business requirements of client and partner of the organization. Also, is expected that the solution implemented helps Audatex Brazil to resolve the problems caused by variability and complexity of business rules.

2 Fundamental Concepts

For a better understanding of Business Rules Management Systems, it is important to define some basic concepts such as: Business rules, Business Rules Management Systems, business rules engine and inference algorithms.

2.1. Business Rules

A Business rules is a statement that defines or constrains some business aspects in organization. The business rules are atomics, so, they should not be broken or isolated from its business context. The Figure 1 shows an example of a rule to define damage coverage option in Auto Insurance context [2].

```
ON (damage-field entered) OR (damage-cause entered)
IF (damage-field = private third party insurance) AND
(damage-cause = damage of a car in use) AND
(third-party-insurance-type = family, single or senior)
THEN Issue error message „Damas of cars in use are not insured by this policy; please check and if necessary, pass the file to the central office.”
```

Figure 1 - Example of rules for damage coverage options

2.2. Business Rules Management System

Considering the speed with which the rules are set and changed, transcending time zones, seasons, information systems, legal limits, etc., the complexity of its management grows exponentially. For this reason, it is needed to adapt a mechanism to manage and maintain this delicate process. So, the most appropriated mechanism for this is, of course, a BRMS. A 100% functional BRMS must include the following components (Figure 2) [2]:

- **Business Rules Repository:** A Database for storing the organization’s business rules. The database for Rules repository can be implement in File System as XML or Excel files repository, or in enterprise database as Oracle or SQL Server.
- **Tools for rules management:** A front-end GUI application that allows business experts to define, edit, simulate, apply and also document business rules.
- **Business Rules Engine/Business Rules Inference Engine:** Typically a software (or software component) that has the function to inferring business rules through a specific stimulus. The rules engines are often categorized by the type of inference that implement, which can be the Forward or Backward Chaining Method.

2.3. Inference Methods/Algorithms

An inference method or algorithm is the technique used by intelligent systems to reason and evaluate a set of facts against rules stored in specific knowledge base. It is, therefore, in deriving conclusions from known premises. Inference methods or algorithms can be probabilistic (clustering analysis and data mining) or deterministic (decision tables, Differential equations and fuzzy logic).
**Forward Chaining**

This inference method execute the rule similar to if-then statement in a procedural programming language (such as C, C++, Java, C#, etc.), but nevertheless, with different application. In procedural programming languages, the if-then statement are performed in an order defined by the programmer. In the forward chaining, the then part of the statement is executed when the if part is satisfied by the facts (data provided for evaluation), which results in a less deterministic execution order than procedural based programming languages [3]. The Rules engine decides the rules execution order [3]. Forward chaining is characterized by being data-driven and reactive.

**Backward Chaining**

In a Backward chaining method, the Rules Engine is oriented to the goal and begins with a conclusion that that the Rules engine tries to satisfy. In such systems, the rules are if-then statements, but in this case, the Rules engine actively tries to reach the goals of a rule [3]. If the if part of a rule is only partially satisfied, the Rules engine may conclude that to prove the goal of another rule might cause that the first rule can be fully proved. If it finds it, it will try to reach the second rule’s goal, which may have its own goals (sub-goals). This process continues until the initial goal is proved, or there are no more sub-goals, concluding as soon as the rule is not applicable. This behavior is often referred to objective demand (goal seeking). The backward chaining is characterized by being goal-oriented [3].

![Forward Chaining vs Backward Chaining](image)

To better understand the difference between these two inference algorithms, two simple examples are shown. The Figure 3 shows an illustration of the execution of each algorithm for the proposed rules. The Forward chaining starts with the data values of a = 1 and b = 2 and uses the rules to derive d = 4. The Backward chaining starts with the aim of finding the value of d and use the both rules to reduce the problem to find a and b values (sub-goals) [3].

**3 PropertyField Manager - Analysis and Design**

This chapter presents the conceptual aspects of developed system, explaining the aspects that originated the architecture on which the PropertyField Manager was implemented, but without showing the technical details. In particular, the chapter present the architecture of the solution and its constraints; design patterns, actors and use cases; and components domain models.

**3.1 Requirements and Constraints**

The proposed solution will be designed and implemented under some important requirements and technological constraints. The solution requirements and constraints were subdivided by the solution modules.

**3.1.1. Requirements**

**PropertyField Repository Requirements**

- R1 - The repository of rules will support the registration and validation rules.
- R2 - The repository of rules will store rules in the specific domain language of business.
- R3 - The repository will store the separate business rules of the respective business processes.
- R4 - The Repository rules will store the metadata of stored business rules.
- R5 - The repository of rules will support data retrieve of business rules and the related metadata.
- R6 - The repository of rules will support referential integrity between the business rules and the respective metadata.
- R7 - The rule repository will support the rule version control.
- R8 - The repository will implement historical management for business rules execution.

**PropertyField Decision Service Requirements**

- R9 - PropertyField Decision Service will execute the business rules requested by enterprise applications into production.
- R10 - The PropertyField Decision Service will infer all enterprise policies and logical decisions used by business applications.
- R11 - The PropertyField Decision Service will execute business decisions independently of the logical form of enterprise applications.
- R12 - The PropertyField Decision Service will support mutual exclusion mechanisms, preconditioning and priority during execution of business rules.
- R13 - The PropertyField Decision Service will provide debugging and tracing functionality for the rules of inference process.
- R14 - The PropertyField Decision Service will support rule execution in simulation mode and a real mode.

**PropertyField Management Studio Requirements**

- R16 - The PropertyField Management Studio will provide features for creating rules.
- R17 - The PropertyField Management Studio will provide features for delete rules.
- R18 - The PropertyField Management Studio will provide features for change rules.
- R19 - The PropertyField Management Studio will provide features for import rules stored in files supported by the solution.
- R20 - The PropertyField Management Studio will provide features to export existing rules in the repository to excel decision tables files.
- R21 - The PropertyField Management Studio will provide features for rules simulation.
- R22 - The PropertyField Management Studio will provide features to view rules in business domain specific language.
- R23 - The PropertyField Management Studio will provide features for debugging rules.
- R24 - The PropertyField Management Studio will provide features for managing dependencies between rules.
- R26 - The PropertyField Management Studio will provide features to manage exceptions associated rules.
- R27 - The PropertyField Management Studio will provide features for managing business entities.
- R28 - The PropertyField Management Studio will provide features for managing (creating and deleting) rules versions.

3.1.2. Constraints

**PropertyField Repository Constraints**

- C1 – The rules repository should be implemented in a MS SQL Server database.
- C2 – The interaction of the others components with the business rules repository should be ensured through T-SQL language.

**PropertyField Decision Service Constraints**

- C3 – The rules engine should be implemented as Shared Library using Microsoft’s .NET Framework.
- C4 – The rules engine should be implemented using C# language belonging to the .NET Framework.

**PropertyField Management Studio**

- C5 – The business rules management tools should be published as a Windows Desktop Application.
- C6 – The business rules management tools must have all its graphical user interface implemented with the Microsoft subsystem WPF (Windows Presentation).
- C7 – The business rules management tools should have its graphical interfaces building with XAML language (eXtensible Application Markup Language).
3.2 Architecture Overview

The proposed solution was implemented with a modular architecture, resulting that its components perform as independent modules with distinct own functions and responsibilities. The architecture will consist in three main components: a Business Rules Repository named PropertyField Repository, a Business Rules Engine named PropertyField Decision Service and finally an Administrative Tool for Business Rules Management named PropertyField Management Studio. Following is described the overview of these main components of the proposed architecture.

• PropertyField Repository: The main function and responsibility of this component is to ensure the persistence and storing process of the Business Rules and their Metadata, and also to provide the mechanism for their access.

• PropertyField Decision Service: This is the main and core component of the solution architecture, its responsibility is to provide whole engine inference process requested by systems in production validating the provided data with the rules persisted in repository.

• PropertyField Management Studio: This module allows business users to manage the Business Rules Management System without strong or any technical skills. The management activities include create, change, delete, execute and debug business rules.

Figure 4 shows an overview of the PropertyField Manager logical architecture, highlighting its modules and the relationship between them.

Figure 4 - Overview of the proposed solution: PropertyField Manager

3.3 Design Patterns

A Design pattern is software engineering approach that represent a reusable solution for a common design and implementation software problem [4]. Design patterns are not necessarily related to problems found in software development, may be also for formalization of best practices that software engineers can use as a reference in the software design and developed process [4].

During the PropertyField Manager analysis and design process, were used some design patterns as reference such as: Database relational model; Unit of Work + Repository (UoW) pattern; and Model View ViewModel (MVVM) pattern.

3.3.1 Relational Database model for PF Repository

Following the Database relational model, the entities of the PropertyField Repository are represented by relations, known also as database tables. The PF Repository tables represent rules entities and their related metadata. The implementation of this patterns brought the following advantages [5]:

• Fast responses for data request, because the data is persisted in a single structure.
• Multiple access: Database management systems provide resource competition and concurrency for relational databases.
• Flexibility: Independence between applications that use the rules repository.
• Data Integrity: Unable to redundancy and inconsistency guaranteed by referential integrity.
• Management Facility: Guaranteed by centralized database information

3.3.2 Unit of Work + Repository pattern for PF Repository Data Access Layer

The access to PropertyField Repository was designed to implement the Unit of Work + Repository Patterns. This main concept of this pattern is the creation of an intermediate abstract layer between application logic and database. This approach is composed by three main components: Unit of Work (the abstract layer), Repositories (the entities CRUD manager) and DbContext or ORM (Object-Relational Mapping) (represents the virtual database) [6].

3.3.2 Model View ViewModel pattern for PF Decision Service and Management Studio logic separation

The application logic for Decision Service and their communication with Management Studio was designed to implement the MVVM Pattern (Model View ViewModel). The overview of this pattern is the separation of application logic from User Interface. The Model represents directly the domain model of the application, the view represents the User Interface and finally the ViewModel represents all implemented application logic [7].

3.4 PropertyField Manager – Main Components

The PropertyField Manager has a modular architecture, making its components independents and with own and distinct functions and responsibilities. The main components described in this sections are: PropertyField Repository - the business rules repository; and PropertyField Decision Service – the business rules engine. These components relate each other through the features that each exposes to another components.

3.4.2 PropertyField Repository

The PropertyField Repository implementation will be done in MS SQL Server relational database, this solution was accepted because this is the most powerful and used paradigm to implement a relational database for most requested business application. The main logical concept in this paradigm is the entities/tables and fields. The tables are, in this case, the rules entities and the fields are theirs properties. The Table 1 shows the list of main tables/entities of rules repository and the Figure 5 shows the resulting Entity-Relationship Diagram.

<table>
<thead>
<tr>
<th>Entity/Table</th>
<th>Importance</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rules</td>
<td>Principal</td>
<td>Main table/entity, store the rules and theirs related metadata.</td>
</tr>
<tr>
<td>Policies</td>
<td>Principal</td>
<td>Stores the rule type (Calculation, Conditional, Data Validation, etc.)</td>
</tr>
<tr>
<td>Scopes</td>
<td>Principal</td>
<td>Stores the rule business context (Insurance, Credit etc.)</td>
</tr>
<tr>
<td>Projects</td>
<td>Principal</td>
<td>Stores rule project/application.</td>
</tr>
<tr>
<td>RulesTargets</td>
<td>Principal</td>
<td>Stores the rules targets information (business entities and related fields).</td>
</tr>
<tr>
<td>RulesTriggers</td>
<td>Principal</td>
<td>Stores rule trigger information (business entities and related fields).</td>
</tr>
<tr>
<td>RulesHistorical</td>
<td>Principal</td>
<td>Store rule versions table reference</td>
</tr>
<tr>
<td>RulesErrorCondition</td>
<td>Principal</td>
<td>Store the execution rule errors conditions</td>
</tr>
</tbody>
</table>

Table 1 – List of Main Tables/Entities of PF Repository

3.4.2 PropertyField Decision Service

The PropertyField Decision Service and is domain model are intended to serve as a basis for the inference process implementation. Once deployed, it will be the coordinator of all inference and decision requested by enterprise information system in production. In general business rules management system reference, it will play the same role as business rule engine, with the responsibility to evaluate and execute the facts sent by live systems. The Figure 6 shows the PropertyField Decision Service Domain Model.
3.4.3 Actors and Use Cases

PropertyField Manager will have their use cases performed by three type of user roles, namely: Business Analyst, Business Manager and System Administrator. Each roles has different access levels and permission to perform a certain use case. However, the business analyst is the default role type, the business manager is a generalization of business analyst. The system administrator is a business manager generalization with total access and permission to perform all system use cases. Figure 7 shows the use cases diagram.
4 Results and Evaluation

This chapter presents the results of the implementation of some important features of the proposed solution. It was taken into account all design and implementation details described in system requirements and constrains.

4.1 Results

The results of the PropertyField Manager implementation are visible through the Windows Desktop application module named “PropertyField Management Studio”. This module includes two major features groups: basic rules management functions and advanced rules management functions.

Home Screen

The home screen (Figure 8) consists in three major areas: Toolbar, Rules Exploration List and Central Stage. The Quick Access Toolbar is where each user can customize the buttons by putting the features that he use most frequently. Just below is the toolbar where the main features like global application definitions, rules recording, editing rule expression, functions, exceptions, simulation and debugging rules.

On the left side is the operating list of rules and respective versions organized by type / nature of the rule where you can perform operations creating, editing and rule of elimination. In the central area is the center stage (center stage), reserved for edition and rule management created or edit mode.
Rule Creation Screen

The creation of a rule is made by rules for the operation list by clicking the add rule (button with the "+" sign) and follow the displayed screen, provide the name of the new rule to be created. Figure 9 shows the new rule creation screen.

Rule Deletion Screen

To delete a stored rule is necessary to access the rules list and click the delete button (button with red “x”). However, for a user who does not have administrators’ permissions, the rules deletion has some functional restrictions, as: It is not allowed to delete rule that is in use in one or more system in production; it is not allowed to delete a rule in use in central stage or a rule version. But, it is possible to delete a rule with versions, in this scenario, two choices are presented: delete a rule and its all versions; or delete only the latest version and promote the previous one as the active rule version.

Rule Edition and Functions Screen

A rule (or rule version) created can be changed by the rule editing functionality, clicking the edit button on the front of the rule or its versions. The rule is opened in "Center Stage" where displayed informations already stored are shows to users and can be deleted, changed or added other information and business entities. The Central Stage it can view information such as the creator of the rule, the creation date, the rule type, version, business entities and their properties (triggers and targets), rule execution results, mistakes, etc. Figure 11 shows the central stage for editing a rule.

Rule Business Entities Manager Screen

Triggers entity management is performed by adding or removing business entities and/or its properties. The added entity/property has its behavior in the rule configured through properties such as: set as input parameter; delete
propagation, ignore the null values, stop the execution and order of propagation. Figure 12 shows the rules triggers entity management screen.

![Figure 11 – Rule Edition and Functions](image1)

![Figure 12 – Rule Trigger entity management screen](image2)

### 4.2 Evaluation

The results presented in the previous section are intended to demonstrate the full meet of proposed objectives for the PropertyField Manager. However, its evaluation was done through factors that are considered important to guarantee the solution asset. The evaluation criteria for the implemented solution was:

- The quality of the results: Parameter using to compare the evaluation results of a rule execution with expected evaluation.
- The efforts of business users to manage rules: A parameter used to assess the time to manage a rule or set of rules.
- Response time in terms of create new rule or change existing rule for Information System in production. This item compares the response time in environment with and without PropertyField Manager.
- Costs of creation and change rules sets in scenarios with and without PropertyField Manager.

Table 2 shows a comparison of number of created rules between PropertyField Manager and rules implemented in information systems logic. Table 3 shows the number of business rules version managed. Table 4 shows the average time and costs for create or change a rule in scenarios with and without PropertyField Manager. Finally, Table 5 shows the results quality evolution of inference process in space of three PropertyField versions released.
<table>
<thead>
<tr>
<th>Scenario</th>
<th>Period</th>
<th>Number of Rules</th>
</tr>
</thead>
<tbody>
<tr>
<td>PropertyField Manager</td>
<td>2 Months</td>
<td>1130</td>
</tr>
<tr>
<td>All IS in production</td>
<td>24 Months</td>
<td>370</td>
</tr>
</tbody>
</table>

Table 2 – Number of created rules PF Manager vs Production Apps

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Period</th>
<th>Number of Rules</th>
</tr>
</thead>
<tbody>
<tr>
<td>PropertyField Manager</td>
<td>2 Months</td>
<td>47</td>
</tr>
<tr>
<td>All IS in production</td>
<td>24 Months</td>
<td>0 (Do not support version control)</td>
</tr>
</tbody>
</table>

Table 3 – Number of rules versions PF Manager vs Production Apps

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Intervention Time</th>
<th>Costs</th>
</tr>
</thead>
<tbody>
<tr>
<td>PropertyField Manager</td>
<td>30 a 60 minutes</td>
<td>N/A.</td>
</tr>
<tr>
<td>All IS in production</td>
<td>2 a 5 days</td>
<td>Must be priced per intervention.</td>
</tr>
</tbody>
</table>

Table 4 – Average time and costs for rules management intervention PF Manager vs Production Apps

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Nº of Inferred rules</th>
<th>Nº de Correct rules</th>
<th>% of correct inferred decision</th>
</tr>
</thead>
<tbody>
<tr>
<td>PropertyField Manager Release 1</td>
<td>240</td>
<td>100</td>
<td>43.6%</td>
</tr>
<tr>
<td>PropertyField Manager Release 2</td>
<td>512</td>
<td>270</td>
<td>52.7%</td>
</tr>
<tr>
<td>PropertyField Manager Release 3</td>
<td>906</td>
<td>713</td>
<td>78.7%</td>
</tr>
<tr>
<td>PropertyField Manager Release 4</td>
<td>1130</td>
<td>1028</td>
<td>91%</td>
</tr>
</tbody>
</table>

Table 5 – Number and quality of inferred rules vs correct rules

5 Conclusions and Future Work

To make this solution possible, was realized a detailed study of BRMS and all impacts it can have in Audatex Brazil information systems management. It was also necessary to perform a study about all organization partners, their business and modus operandi. In the other direction, was analyzed the complexity and variety of partners business rules inferred by the application in production. Also, was thoroughly analyzed all the alternatives possible solution to solve the Audatex Brazil partners rules management caused by complex and heavily legislated industries such as financial organization and insurance companies.

Among all possible solution, was reached the conclusion that the completion of a BRMS could play an important role in improving the organization’s business processes and decision management, and consequently an improvement in the organization’s performance in response to market new demands.

Being this solution restricted to financial industry, more specifically the auto insurance companies and Audatex Brazil partners, the aspects presented and discussed in this work leave interesting perspectives for future work. From the current solution state, in addition to their improvement, there are other issues that make sense for future researches, including alternatives to achieve business rules inferences not based on decision tables of matching algorithms.

Even inside the same analysis, can be developed researches to optimize this solution for solving the previously described problem related to high-level coupled between PropertyField Manager and information systems in production. The focus of this research could be changes to make the solution more generic, lowly coupled and with easy integration with other enterprise applications in production.

References