Calculation of Insurance Policy Prices Using Rules-based Systems

Mina Medić, Katarina Preradov, Miroslav Zarić, Goran Sladić, Siniša Nikolić
University of Novi Sad, Faculty of Technical Sciences, Novi Sad, Serbia

{mina.medic, katarina.preradov, miroslavzaric, sladicg, sinisa_nikolic}@uns.ac.rs

***Abstract* — The main focus of this paper is the design of a system that will enable a declarative approach, based on a set of rules, to support complex pricing calculation logic for the insurance industry. The insurance industry is highly regulated by legal and specific domain enforced rules. Those rules are frequently adjusted, therefore creating additional effort for software maintenance, as it needs to be updated regularly as well as on short notice when sudden changes are introduced. The key feature of the proposed system is its adaptability to those frequent changes in environmental parameters in real time, achieved through the use of rule-based calculation module. This approach brings necessary flexibility to the system, reducing maintenance downtime as well as reduced susceptibility to errors during source code changes. Our approach relies on rules-based systems (engines) that execute complex business computations, declared as a set of business rules, for the insurance sector.**

***Keywords* — expert systems, rule-based systems, Drools, risk management, insurance policy**

I. INTRODUCTION

The risk is commonly defined as the possibility of loss and is the part of daily life which cannot be avoided [1]. In addition, it can either represent the loss of health, life or property. Insurance is one of the most popular ways that both individuals and organizations use to minimize the effect of future potential economic or financial losses. Insurance provides protection against various risks that may endanger property and/or person's life by promising a payment of compensation, or insured sums, in case of the occurrence of the insured event. This payment is provided in exchange for upfront or payment in installments of a predetermined insurance premium.

When defining the insurance, it is necessary to identify and assess the risks, in order to define the amount of capital that the insurance company must possess in order to be able to settle the due obligations towards the insured, at any given time during the validity of insurance policy. The precise risk estimation is one of the biggest challenges in the operation of insurance companies. If calculated risks are stated too high, the company will require higher insurance premiums from the customers - and risk losing the competitive edge, or if the risks are calculated too low, the company is facing possible financial losses. Therefore, incorrect risk estimation can lead to negative financial impact for insurance company, for insured customers, or for both.

The capability to successfully provide risk assessments is the defining feature of each insurance companies. The precise algorithms, procedures, and experts performing the risk analysis are often the most valuable assets and a core business value of the insurance company. In order to fine-tune their pricing, insurance companies need to thoroughly understand the potential risks that come with certain insurance products, especially when introducing new insurance options. Moreover, development and advances in the field of risk management may decisively influence an insurer's overall profitability and competitiveness, giving the company a competitive edge and better profit margins.

Many business systems today use knowledge-based systems to automate their business processes. Rule-based systems rely on rules as the most common paradigm for defining the system behavior, specific algorithms for management and decision-making. Written in a simple form, the condition-action rules allow the execution of actions in the form in which the man would do it. As a software system that is logically separated from the application code, rule-based expert systems allow users to easily manipulate, modify, and add rules related to the business itself.

The goal of this paper is to describe how to automate the process of insurance policy price calculation using a rule-based system. In order to realize this idea, it was necessary to develop a support decision-making system in addition to the basic information system. The primary goal of this model is to use the knowledge of the experts in the given field to make all necessary calculations.

We organize this paper in the following manner. In Section II, we present the related work. In Section III, we introduce the concept of rule-based systems and their basics. In Section IV we discuss system design. We present a data model and provide a brief overview of the main functionalities of the presented solution. In Section V, we analyzed the possible ways of system integration. We discuss strategies on how Drools can be integrated into the reference architecture and discuss the pros and cons of each of them. Finally, we discuss limitations and future work in Section VI and the last section gives the conclusion of this research.

II. RELATED WORK

Throughout the literature review for this research, we concluded that a lot of authors are trying to solve the same problem of automating the process of insurance policy price calculation, and therefore a lot of different solutions have been proposed. Not all papers are from the computer science field, but there were a few which were the good starting point for the beginning of our research.

Authors in [2] suggest a case-based approach. In this approach, the insurance application is compared with earlier ones and based on the degree of overlap (confidence factor), the price of the new policy is determined. The main problem with this approach is that it is not adaptable to frequent changes in factors that affect price calculation. This limitation is a huge problem because one of the main characteristics of price factor - rapid changes. Also, this algorithm has been poorly proven at the very beginning of the insurance company's business because it is a very small set to compare new policies because there were not enough sales. There is also an open question: with what to compare the first policy that enters the system?

In paper [3] the authors introduce a solution that relies on a set of declarative rules. They have managed to calculate the premium by using Microsoft Excel as the design time tool. Additionally, this same tool has helped them to specify certain guidelines and execute the spreadsheet at runtime. The main advantage of this paper is that it eliminates the need for translation of these rules into software code. On the other hand, the major limitation of this solution is that it is not integrated with a company's information system but is solely depends on data entered through Excel.

III. Methodology

In order to determine the insurance price (insurance premium) which insurance companies use to charge for their products, some companies heavily rely on compound rules and computations. The computations of insurance premiums are well-defined by numerous organizations and specific rules defined by the insurance company. Processes, such as interpreting the standard rules, integrating them with company-specific rules, converting these declarative rules into technical specifications and implementing them as a part of software programs are immensely complicated and time-consuming. In addition, the same process requires multiple manual steps and expertise from different areas (IT staff, business, and risk analysts). Due to the difficulty of the whole process and numerous steps involved, errors can occur frequently.

In order to propose an appropriate solution for the described problem, two approaches have been analyzed in the remaining of this section.

A. "hard coded" rules

The first approach was to represent business rules as part of the program code in the form of "if-else" statements. This programming paradigm is known as Imperative Programming. Imperative Programming is defined by the control of the sequence flow of instructions, the programmer explicitly defines when each code instruction should be executed. On this way, rules are tightly coupled with application logic and therefore not flexible enough to handle rapid changes in prices and factors. A system like this will have a low level of availability and reliability, which can directly influence the success of an insurance company.

B. Basic concepts in Rule-Based Systems

The business rules are based on a programming paradigm called Declarative Programming. Declarative programming strives for the ideal of programming by wish: the user states what he or she wants, and the computer figures out how to achieve it. Thus, declarative programming splits into two separate parts: methods for humans on how to write wishes, and algorithms for computers that fulfill these wishes [4].

Rule-based programming has its foundations in the symbolic production systems of Emil Post. The basic rule-based programming approach is to decompose a computation into a set of elementary transformations. Each elementary transformation attempts to match its input against a set of templates. When some of the templates match, a rule that is corresponding to one of these templates is chosen, and the action associated with the rule is effected. If no templates match, the program halts [5].

Although many different techniques have emerged for organizing collections of rules into automated experts, all rule-based systems share certain key properties:

- practical human knowledge is incorporate in conditional if-then rules,
- the skill of reasoning increases proportional with the size of their knowledge base,
- by selecting relevant rules and then combining the results in appropriate ways, they can solve a wide range of possibly complex problems,
- the best sequence of rules is always executing, and
- conclusions are explained by retracing their actual lines of reasoning and translating the logic of each rule employed into natural language [6].

The basic form of any rule is as follows:

$$rule : \langle preconditions \rangle \rightarrow \langle conclusions \rangle$$

where *<preconditions>* is a formula which is defining when the rule can be applied, and *<conclusions>* is the definition of the effect of applying the rule; it can be a logical formula, decision or action [7].

The *<preconditions>* is also referred to as the Left Hand Side of the rule (LHS). The LHS of the rule is composed of conditional elements, which serve as the filters to define the conditions that need to be met for the rule to evaluate true. The *<conclusions>* is also referred to as the Right Hand Side of the rules (RHS).

IV. System design

The system has decentralized architecture and it consists of several subsystems that optimize the operations of the insurance company, such as the website for sale of insurance policies with online payment support, e-mail server, internal application for updating price lists and system administration. The most valuable part of the system is a knowledge module used for price calculation. Developing and optimizing this part of the system is the primary topic of our research and for that reason, most attention in this paper is focused on this module.

The insurance policy itself is described by the *Policy* class. In order for the policy to be valid, it is necessary to have the beginning and the end of the insurance, as well as the type of insurance it covers. The implemented system allows simultaneously one policy to be linked to multiple types of insurance, such as travel, vehicle or property insurance. In addition to the above-mentioned data, the policy also includes the person with whom the contract was drawn up, as well as the list of insured persons. When defining insurance, all potential risks that affect the price of the same are concluded.

Each type of insurance (travel, vehicle or property) carries with it certain risks. The risks are described by the *Risk* class and depending on the type of insurance they can refer to the age of the insured, whether it is engaged in sports and similar. On the other hand, they may be linked to the region where the insured person is traveling, or, in the case of property insurance, size or age of the property itself. *RiskItem* class represents the specific value of a particular type of risk, e.g. if the risk is travel regions, the risk items are America, Europe, Africa, etc. The price list is described by the *Pricelist* class and it is modeled on the reputation of the standard pricing model that allows tracking the history of price changes.

V. System Implementation

The first step in developing a system for defining insurance policy pricing was to find the appropriate development tool for the knowledge base (BRE or BRMS) [7]. Considering that it has been previously decided to introduce knowledge through rules, this tool must support the suggested way of presenting knowledge, as well as forward-chaining techniques. For easier integration with the rest of the system, it was also convenient for this tool to be written in the Java programming language. Since support for near real-time changes is crucial, the next step was to set up a convenient system architecture corresponding to this request. The last step was to define a set of specific rules for the tasks of risk management.

A. Drools tool

After analyzing some of the most familiar tools for presenting knowledge base, it was decided to use Drools for this purpose. Drools [8] is a free, open source Business Rule Management System (BRMS) written in the Java programming language. It uses a specific syntax for writing rules - Drools Rule Language. The technique supports forward and backward-chaining. Conflicts in executing rules can be resolved by assigning priority to rules or by ensuring that a particular rule can be executed only once during the session. Drools use Java storage facilities, so integration with other applications written in Java is very simple.

B. Integrating with the rest of an application

The first thing we need to address when designing how Drools should interact with the rest of our application components is how they will fit in the overall architecture; we will have special requirements in regards to how data will be fed into the rule engine. Also, we must decide how information should be published back to our application from the rule engine, or how it should be exposed to the rest of the applications.

Drools is a framework; as such, it can interact with other frameworks in many ways. In the next sections, we will discuss some architecture approaches to integrating Drools with the rest of our design, including some pros and cons for each approach.

Embedded Drools

The first most common step for integrating Drools is usually embedding its dependencies and code inside our own application and using it as a library. Figure 1 shows this integration as it happens in the first stage.

During this first stage in the development of rule-based projects, all the components needed to run our rules are usually included in our application, including the rules themselves. This means we will have to redeploy our application each time when we want to change the business rules running in it. So we upgrade the system to achieve an independent development life cycle for our business rules. This independence will allow for the rules to be developed, deployed, and managed as many times as needed without having to redeploy our applications.

Drools as a service

Upgrading the architecture leads us to move business rules as an outside dependency, defined as a KJAR. KJAR (knowledge artifact) represents a standard JAR file within which an additional configuration document is included. This change is not just a project rearrangement, because the actual JAR with all the rules won't be a dependency at the moment of deploying our application. Instead, the Drools runtime will read the JAR directly from our Maven repository (local or remote) and deliver the rules whenever needed.

Components in our rule runtime can dynamically load these rules from outside repositories, as Figure 2 shows.

Also, once our business rule requirements start growing — including dynamic rule reload, more and more calls, and interactions with external systems — we get to the point where multiple applications would need to replicate a lot of things in their own runtimes to reach the same behavior for what could be common business logic. The natural transition at this point is creating knowledge-based external services outside our application. Managing independent rule development life cycles becomes easier, as does the possibility of replicating the service environment for higher demands. Also, all client applications can share and improve the same knowledge base.

C. Define the knowledge base

At first, it was necessary to explore and understand the given business domain and to collect and then formalize the knowledge needed to define risk categories. Afterward, the risk categorization significantly influences the premium paid by the applicant. In other words, the greater the risk category, the greater the premium. There are several different methods of collecting knowledge. Some reliable ways of defining rules involve consulting a domain expert, finding appropriate literature, such as university textbooks, case studies, scientific journals, etc.

Part of the knowledge base was created based on rules which are used in some of the already existing insurance companies. These rules have been altered slightly so they do not represent the business rules of real companies. The main reason that the rules have been modified is that because they represent confidential information and any public disclosure of the secret of any company's business can lead to financial losses for that company. Also, some information is strictly confidential and any revelation of such types of information may be prosecuted.

Another part of the knowledge database consists of experiential (heuristic) rules that generally can't be found in the literature. This part of the knowledge database represents the weakest point in our the application and

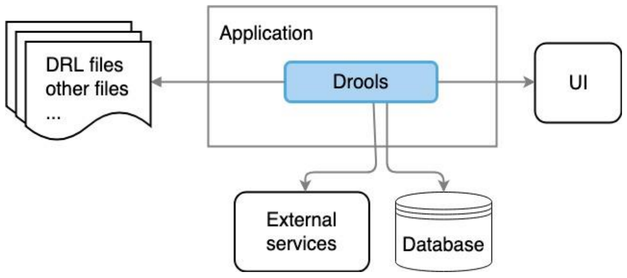


Figure 1 - Embedding Drools into our application

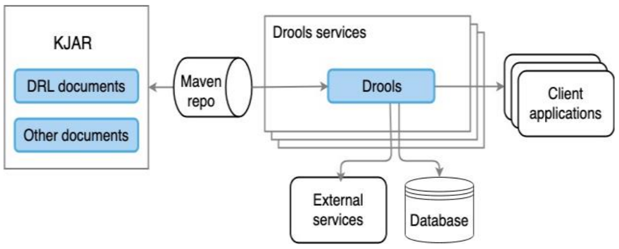


Figure 2 - Drools as a service

therefore improving or completely removing these rules are one of the main goals in future work.

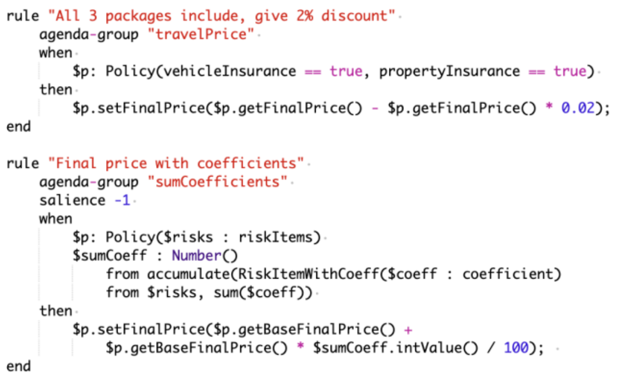
Most of the implemented rules are based on multiplying the insurance policy base price with different coefficients. This coefficient represents a factor that is the most valuable part of the system. These factors influence the final cost of the policy so that insurance company can have the most of it. In our implementation, coefficients were divided into two groups that were proposed in paper [9]. But the key problem that remains for future work is: how to define their optimal values?

Since there are three large groups of insurance packages, it was decided to define three different files with rules, each file corresponding to a particular package. According to the Drools specification, the rules are written within the files with the *.drl* extension. Accordingly, they were created:

- *property.drl* - file with rules related to property insurance;
- *travel.drl* - file with rules related to travel insurance;
- *vehicle.drl* - file with rules related to vehicle insurance.

Part of the knowledge base implemented in the Drools tool can be seen in Figure 3.

Drools syntax states that each rule must have a unique identifier written after the reserved word *"rule"*. The rules that have been presented have a different priority (*salience*) [7]. The second rule's priority is -1, which means that it will be executed after all rules with a higher priority. It is also set up that each of these two rules belongs to different groups (*agenda-group*). The beginning of the rule premise is always indicated by the reserved word *"when"*, while the consequence of the rule is written in the form of ordinary Java statements after the reserved word "*then"*. The end of each rule is indicated by the reserved word *"end"* [7].



```
rule "All 3 packages include, give 2% discount"
    agenda-group "travelPrice"
    when
        $p: Policy(vehicleInsurance == true, propertyInsurance == true)
    then
        $p.setFinalPrice($p.getFinalPrice() - $p.getFinalPrice() * 0.02);
end

rule "Final price with coefficients"
    agenda-group "sumCoefficients"
    salience -1
    when
        $p: Policy($risks : riskItems)
        $sumCoeff : Number()
            from accumulate(RiskItemWithCoeff($coeff : coefficient)
            from $risks, sum($coeff))
    then
        $p.setFinalPrice($p.getBaseFinalPrice() +
            $p.getBaseFinalPrice() * $sumCoeff.intValue() / 100);
end
```

Figure 3 - Part of the knowledge base implemented in the Drools tool

VI. Discussions

In this section, we outline the main limitations of the implemented system and propose some solutions for the problems which are still open issues in this area. Addressing those issues are left to be a part of our future research.

A. Limitations

As stated in the previous sections, most of the knowledge base consists of experiential (heuristic) rules. Burnett et al. in [9] propose a separation of parameters in two groups: demographic (age, sex, income, education, marital status, etc.) and psychographic (work ethic, fatalism, socialization preferences, etc.). Since there is no solid guarantee that these parameters are entirely optimal, there is a requirement to run a specific process that can optimize those parameters.

B. Future work

The tunable parameters have a substantial influence on the general accuracy and coverage of the decision-making process. Besides, it is of great importance to tune these parameters correctly since the quality of decision-making and maintainability of these systems depends on it. In future work, attention will be devoted to the optimization of parameters. As already stated, the knowledge base mostly consists of heuristic rules, so the improvement of these rules is certainly a part of future system improvements.

To support these statements, our future work will be dedicated to addressing the following statements:

- consult more experts to determine the best asset of rules;
- improve or completely remove heuristic rules from the knowledge database;
- consider machine learning algorithms for fine-tuning of parameters that influence the policy price.

VII. Conclusion

The research problem presented in this paper refers to the problem of automating the process of the price calculation for the insurance policy. Policy price can be influenced by various factors and therefore the automated process can help in defining the ideal price for the policy so that the effect of potential economic or financial losses is minimized for both company and insured people. The model developed for the purposes of this research aims to demonstrate that the use of expert systems, more specifically rule-based systems, can help in making introduced idea achievable. To prove this, an application for online sales of insurance was developed.

The architecture of the developed system is designed to represent a simplified version of the real system of an insurance company, but still to retain all key aspects. The paper describes the basic functionality of the system, with emphasis on the concepts of the rule-based system, defining knowledge base with domain rules and their use in insurance policy price calculation process.

References

1. M. H. Meyer, A. DeTore, S. F. Siegel, K. F. Curley, "The strategic use of expert systems for risk management in the insurance industry," *Expert Systems with Applications* 5.1-2: 15-24, 1992.

2. B. P. Patrone, et al. “System for case-based insurance underwriting suitable for use by an automated system,” *U.S. Patent No.* 7, 630, 910, 2009.
3. V. Sundar, K. Shankar, K. Ravi, “Method and system to implement complex pricing rules,” *U.S. Patent Application* No 10/329, 627, 2004.
4. M. U. Armin, W. Oskar, B. Ulrich, G. Dietmar, S. O. Takata, “Declarative Programming for Knowledge Management,” 16th International Conference on Applications of Declarative Programming and Knowledge Management, INAP 2005 Fukuoka, Japan, October, 2005.
5. T. J. Kowalski, L. S. Levy, “Rule-based programming,” *The Springer International Series in Engineering and Computer Science*, 1996th Edition
6. F. Roth, “Rule-based systems”, *Communications of the ACM*, vol. 28, issue 9, pp. 921-932, September, 1985.
7. M. Salatino, M. De Maio, E. Aliverti. “Mastering JBoss Drools 6,” *Packt Publishing* Ltd, March, 2016.
8. P. Browne, “JBoss Drools business rules,” *Packt Publishing Ltd*, April, 2009.
9. Burnett, John J., and Bruce A. Palmer. “Examining life insurance ownership through demographic and psychographic characteristics,” *Journal of risk and insurance*: 453-467, 1984.