Feb, 2014

Rules Harvesting from Source Code

By

Shantanu Paknikar
Happiest Minds, IT Services Group

Abhishek Anand
Karunesh Kumar Pandey
Kalpesh Kotkar
Debjyoti Bhol
IIM Rohtak, PGPM - 2014

happiest minds
The Mindful IT Company
Born **Digital** . Born **Agile**

Executive Summary

Over the last decade and more, enterprises across industry sectors have built several custom business applications to help them run their day-to-day business operations efficiently. Some examples of where such applications are used include customer on-boarding, order management, self-service and customer care portals, and so on.

One important aspect of the business logic in such applications is the concept of business rules. A business rule is defined as logic that defines or constrains some aspect of business and always resolves to either true or false. Business rules are intended to assert business structure or to control or influence the behavior of the business. Business rules describe the operations, definitions and constraints that apply to an organization [1].

In today's world, business agility is a critical success factor. Therefore, enterprises need to have the ability to change business rules rapidly, especially in response to the changing business landscape. However, with business rules logic embedded in source code, this becomes a major problem. Any changes to business rules require changes to the application source code, resulting in long cycle times for enhancements, testing and re-deployment.

The need therefore is to extract business rules logic from the application source code and re-implement these on an external *Rules Engine platform.* This is easier said than done, though. The complexity of business applications means that it will be a herculean task to manually search for the business rules in the source code. A semi-automated approach is therefore needed for the business rule extraction from such business applications.

The problem above applies across industry sectors. This paper focuses on the Insurance Industry, and the area of Auto Insurance Claims Processing. We attempt to identify some of the business rules for this industry and recommend approaches to extract and implement these rules.

1. Insurance Industry Overview

Insurance of any type is all about managing risk. As per Investopedia Insurance is a form of risk management in which the insured transfers the cost of potential loss to another entity in exchange for monetary compensation known as the premium[1]. For example, in life insurance, the insurance company attempts to manage mortality (death) rates among its clients. The insurance company collects premiums from policy holders, invests the money (usually in low risk investments), and then reimburses this money once the person passes away or the policy matures. A person called an actuary constantly crunches demographic data to estimate the life of a person. This is why characteristics such as age, sex, whether smoker, illnesses and so on, all affect the premium that a policy holder must pay. The greater the chance that a person will have a shorter life span than the average, the higher the premium that person will have to pay. This process is virtually the same for every type of insurance, including automobile, health and property.

1.1. The Agility Challenge

The Insurance business as such can be divided into various sub-heads.

- Marketing
- Risk Modeling
- Sales
- Policy Administration
- Claims
- Customer Service

There are different software applications available to take care of each aspect of the business above. For example, an insurance organization can have claim processing software from vendor x, risk modelling from vendor y, and in house custom built software for policy administration and other areas.

For custom business applications, especially those built a decade and more ago, business rules logic is often embedded directly in the source code. As the size and complexity of such applications grows, such embedded business rules present a huge challenge when any of the rules needs to be changed.

The challenge is that any change to the rules functionality means that the application has to go through the entire SDLC lifecycle:

1. Business rule modification in source code requires development efforts.
2. The changed application needs to be tested to ensure it is working smoothly and that the changes have not caused any impact to other components.
3. The changed application is re-deployed.

The net result of the above is cycle times of a few weeks to a few months for every change. In today’s world, this is impractical. To be agile, the business needs to be able to change business rules much more frequently with much shorter turnaround times, from a few days to sometimes a few hours as well.

1.2. Our Recommendation: Text Analytics on Source Code

Over the years, there has been a lot of work done in the area of text analytics. Natural Language Processing (NLP) algorithms and technologies are quite mature and a subject of ongoing research, providing reasonable results even for unstructured text in applications such as sentiment analysis. Source code is structured text and therefore, our hypothesis is that applying text analytics techniques to extract rules patterns from source code should be an equivalent or easier problem to solve as compared to unstructured text.

A large number of custom business applications have been implemented using J2EE (Java 2, Enterprise Edition) application server technologies with the business logic implemented in the Java programming language, and often containing several thousand to several hundred thousand lines of source code. For this paper, we have considered applications written in the Java programming language. However the approach and recommendations are applicable for business applications written in any programming language.

We begin by identifying KPIs for the Insurance Industry, and then identifying typical business rules logic which might impact these KPIs. We then list out some challenges likely to be faced during rules extraction and provide recommendations to address these. Finally, we provide the overall approach and methodology that we would recommend for a project implementation.

2. KPI for Insurance Industry

Most of the business keep tap on their performance by means of Key Performance Indicators (KPI). Some of the well-known Insurance industry KPIs are:

- Net Income Ratio
- Policy Sales Growth
- Average Cost per Claim
- Renewal Ratio Claims Ratio
- Average Time to Settle a Claim
- Sales Ratio

Why are the KPIs important in the current context? The reason is that for maximum business agility and impact, it is the business rules logic related to these KPIs that should be easy to change.

2.1 Auto Insurance

Auto insurance is a policy purchased by vehicle owners to mitigate costs associated after an auto accident [2]. Instead of paying out of their own pockets for accidents, people pay annual premiums to an auto insurance company; the company then pays all or most of the costs associated with an auto accident or other vehicle damage. In many jurisdictions it is compulsory to have vehicle insurance before using or keeping a motor vehicle on public roads. Most jurisdictions relate insurance to both the car and the driver, however the degree of each varies greatly.

2.2 A few business rules for Auto Insurance Claims

Business rules are mostly of the “if else” format. Given below are a few examples of rules for the area of claims processing.

Claim Application validity – Discounts

```
if
        No claim is made in the past 2 years
Then
        Give 20% discount
Else if
        No claim is made in the past 1 year
Then
        Give 10% discount
```

Investigation Policy

```
if
        The claim amount is < $500
Then
        Do not investigate, process claim normally
```

Renewal and No claim Bonus

```
If
        Policy is not renewed within 90 days of the expiry date of the previous
policy.
Then
        No Claim Bonus will not be given
```

Claim Application validity – Insurance Cover

```
if
        Driving within 3 hrs. Of alcohol consumption and amount of alcohol in
        urine/breath > prescribed limit
Then
        Insurance cover=0
```

Driving License validity check

If

The Owner-Driver doesn’t hold an effective driving license,

Then

Reject claim as policy is not valid;

Vandalism check

If

You make a claim for damage to your car that is a result of vandalism, which is damage caused by a malicious and deliberate act,

Then

You will not lose your No Claim Discount

Ownership validity check

If

The Owner-Driver is not the registered owner of the Private Car insured

Then

Reject claim as policy is not valid

Anti-theft device discount

If

Vehicle has anti-theft device fit to it

Then

Insurers are eligible for a discount of 2.5%

3. The challenges for rules extraction and possible resolutions

The first approach to detecting business rules logic in application source code is to do a simple text search on the code base. However this is unlikely to produce any meaningful results because of the following reasons:

3.1 Challenge #1: Differing Variable Names

The variable used in converting a business rule into programs can have different names. For example, from a business analyst point of view a rule can be something like "if policy is rejected by company return 100% of premium while if it is cancelled by policy holder return 50% premium (on pro rata basis)"

A programmer can model this rule into source code as

```
Boolean policyrejectedByComp=false, policyCancellecByUser=false
……….
………..
if (policyrejectedByComp) {return premium}
Else if (policyCancellecByUser) {return premium *0.5}
```

the variable names used here; i.e. *policyrejectedByComp* and *policyCancellecByUser* are totally dependent on the programmer.

3.1.1 Recommended Resolution

One way to address the challenge above would be the following:

1. First try to identify the business vocabulary i.e. set of terms used in business (for our case it is auto-insurance). Any business rule will typically act on one or more terms of this vocabulary.
2. One way to do this is from database tables. Often, most of the business terms are related to state of information and are stored somewhere in the database. Therefore, column names in database tables can be a good proxy for the business terms.
3. Once we have a business vocabulary we can guess the names of the variable that a programmer might select. The authors believe there will typically be a high correlation between a variable used in coding a business rule and business vocabulary keyword.
4. To check whether a variable x can be a possible substitute for business vocabulary y, we can use the **Levenshtein distance algorithm.**

For example, instead of a variable name "policyrejectedByComp", a developer can use "policyCancelledByComp". We can see that there is very high chance that one of the words like policy, or rejected company will be used in the variable name.

3.2 Challenge # 2: Different ways to represent a rule

There could be different ways of implementing business rules logic depending on programming styles. An example is given below:

If Accident results in Death
Then scale of compensation = 100 % Capital Sum Insured

If Accident results in Loss of two limbs or sight of two eyes or one limb and sight of one eye
Then scale of compensation = 100 % Capital Sum Insured

If Accident results in Loss of one limb or sight of one eye
Then scale of compensation = 50 % Capital Sum Insured

If Accident results in Permanent Total Disablement from injuries
Then scale of compensation = 100 % Capital Sum Insured

This rule can be implemented either as a nested if else clause or independent if clauses – however in both scenarios, represents the same rule logic.

3.2.1 Recommended Resolution

One way to handle this is to model this rule both as nested clause and independent clause and try to find both of them in source code.

4. Overall Approach for Rules Extraction

The image below summarizes the approach that for extraction.

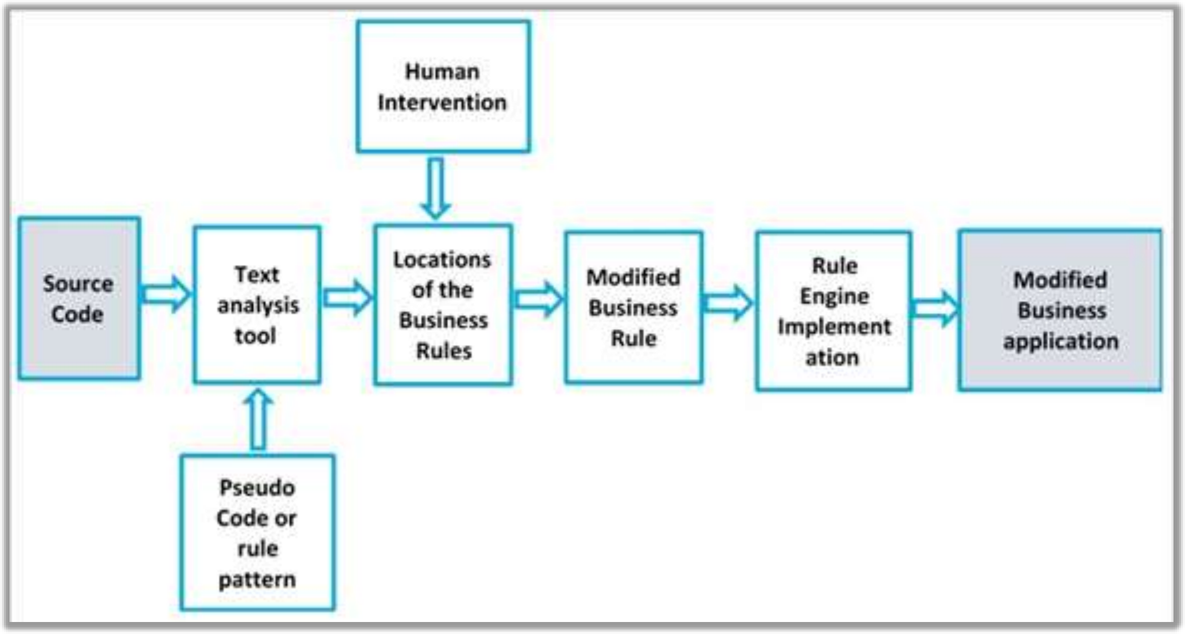
Human Intervention
Source Code
Text analysis tool
Locations of the Business Rules
Modified Business Rule
Rule Engine Implement ation
Modified Business application
Pseudo Code or rule pattern

Fig: Flow diagram of rule harvesting from source code

5. Methodology

Step1. Model the business rule as if-else block pseudo code

Step 2. Transform pseudo code into semi -regular expression

e.g. - If driving within 3 hrs. of alcohol consumption and amount of alcohol in urine/breath is greater than prescribed limit then Insurance cover does not apply. Possible pseudo code can be -

If (time of alcohol consumption<3 OR urine in blood> prescribed limit)
Then insurance_cover=0

Transformed semi-regular expression
If (a<b || c>d) then {}

Meaning we are looking for if code block having 2 comparison of type less than and greater than joined by logical OR.

Step 3. Filter output by comparing variable names inside the if block similar to keywords

1. Do cluster analysis among all the existing if else code block, this way all the if-else block having similarity in logical operand will be grouped
2. After the input is transformed into pseudo code, find the cluster which is closest to it.
3. Do a classification among all the members of the cluster, use variable names as deciding factor for matching.
   E.g. in above example alcohol and urine are keywords, so we filter down the result of step 2 by eliminating all the code blocks which do not have these keywords.

Step 4. Consult business analyst to check whether any of the found code block correctly represents the business

Step 5. If yes isolate that code block and implement in some rule engine so that future change is hassle-free

5.1 Pseudo code of algorithm

```
For (every source file in code base) {
        For (every if else block) {
                If (pattern of conditional logic is as in input) {
                        If (similarity between variable used in conditional logic > threshold) {
                                Print the if else code block
                        }
                }
        }
}
```

6. Summary

Custom software applications developed over the last ten to fifteen years have become more and more complex. As the business environment becomes very dynamic, the ability to make rapid changes to business rules logic in these applications becomes critical. However the complexity of applications results in long cycle times for changes to be implemented, since the entire Software Development Life Cycle has to be repeated. The need of the day is to extract business rules logic from existing applications and re-implement this in an external rules engine or decision platform. The most difficult part of this exercise is the extraction of the rules logic itself. This paper proposes an innovative approach to this problem. The key business benefit is enhanced agility and the ability to respond rapidly to change.

References

1. http://en.wikipedia.org/wiki/Business_rules
2. http://www.investopedia.com/university/insurance/insurance1.asp
3. http://www.investopedia.com/terms/a/auto-insurance.asp
4. Wang ,sue and he, Automatically Identifying Domain Variables based on Data Dependence Graph
5. Li & Zohu, PRMiner:Automatically Extracting Implicit Programming Rules and Detecting Violations in Large Software Code
6. Erik Putrycz (2007) : Recovering business ruled from legacy software http://2007.ruleml.org/docs/erik%20putrycz%20-%20RuleML%202007.pdf
7. Dhillon, Malela & Kumar (2003, March 3) : A Divisive Information-Theoretic Feature Clustering Algorithm for Text Classification
8. Dumas, Plat &Heckerman: Inductive Learning Algorithms and Representations for Text Categorization
9. Sneed & Ordos (1996) : Extracting Business Rules from Source Code
10. Aponte, Ortega & Marcus : Towards the Automatic Extraction of Structural Business Rules from Legacy Databases

About Happiest Minds Technologies

Happiest Minds, the Mindful IT Company, applies agile methodologies to enable digital transformation for enterprises and technology providers by delivering seamless customer experience, business efficiency and actionable insights. We leverage a spectrum of disruptive technologies such as: Big Data Analytics, AI & Cognitive Computing, Internet of Things, Cloud, Security, SDN-NFV, RPA, Blockchain, etc. Positioned as "Born Digital . Born Agile", our capabilities spans across product engineering, digital business solutions, infrastructure management and security services. We deliver these services across industry sectors such as retail, consumer packaged goods, edutech, e-commerce, banking, insurance, hi-tech, engineering R&D, manufacturing, automotive and travel/ transportation/hospitality.

Headquartered in Bangalore, India; Happiest Minds has operations in USA, UK, The Netherlands, Australia and Middle East.

To know more about our offerings. Please write to us at business@happiestminds.com

About the authors

Shantanu Paknikar (shantanu.paknikar@happiestminds.com) is General Manager, Innovative Business Solutions in the IT Services division of Happiest Minds. His interest areas include Middleware systems, SOA, BPM, Decision Management, Enterprise Integration, Cloud Computing, Social Media, Mobility, Analytics, Multimedia, and Distributed Computing Systems. He is passionate about innovation, research, and learning.

Abhishek Anand, Karunesh Kumar Pandey, Kalpesh Kotkar and **Debjyoti Bhol** are students of IIM Rohtak, PGPM – 2014.The work on this paper was done during Business Analytics course at IIM Rohtak in collaboration with Happiest Minds.

DISCLAIMER: It may be noted that the authors take full responsibility for the content. IIM Rohtak does not necessarily subscribe to the views expressed in this paper, which are those of the authors.