CONDITION ORIENTED ENRICHMENT APPROACH IN BUSINESS RULE MANAGEMENT SYSTEMS

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

MUSTAFA HALİL YILDIZ

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
COMPUTER ENGINEERING

SEPTEMBER 2013

Approval of the thesis

CONDITION ORIENTED ENRICHMENT APPROACH IN BUSINESS RULE MANAGEMENT SYSTEMS

submitted by **MUSTAFA HALİL YILDIZ** in partial fulfillment of the requirements for the degree of **Master of Science in Computer Engineering Department, Middle East Technical University** by,

Prof. Dr. Canan Özgen _____
Dean, Graduate School of **Natural and Applied Sciences**

Prof. Dr. Adnan Yazıcı _____
Head of Department, **Computer Engineering**

Prof. Dr. Ali Hikmet Doğru _____
Supervisor, **Computer Engineering Dept., METU**

Assoc. Prof. Dr. Halit Oğuztüzün _____
Co-Supervisor, **Computer Engineering Dept., METU**

Examining Committee Members:

Assoc. Prof. Dr. Ahmet Coşar _____
Computer Engineering Dept., METU

Prof. Dr. Ali Hikmet Doğru _____
Computer Engineering Dept., METU

Assoc. Prof. Dr. Pınar Karagöz _____
Computer Engineering Dept., METU

Dr. Cevat Şener _____
Computer Engineering Dept., METU

Ümit Vardar, M.Sc. _____
Owner, Vardar Software Company

Date: *4th September 2013*

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last Name: MUSTAFA HALİL YILDIZ

Signature:

ABSTRACT

CONDITION ORIENTED ENRICHMENT APPROACH IN BUSINESS RULE MANAGEMENT SYSTEMS

Yıldız, Mustafa Halil
M.S., Department of Computer Engineering
Supervisor: Assoc. Prof. Dr. Ali Hikmet Doğru
Co-Supervisor: Assoc. Prof. Dr. Halit Oğuztüzün
September 2013, 60 pages

This study is about enrichment process in Business Rule Management Systems. A new approach "condition oriented enrichment" and its benefits are discussed in this study. This approach suggests adding limitation to inquiry method at enrichment service that is corresponding to related business rule's condition and targets to decrease the number of enriched values and improve performance. In this study, because most of the rules are suitable for implementation of this approach, "Turkish National Health Data Validation System" is chosen and implemented as Business Rule Management System with a domain specific language which is named as "Health Data Validation Specific Language".

Keywords: Business Rule Management System, Enrichment in Business Rule Management System, Domain Specific Language, Software Engineering

ÖZ

İŞ KURALLARI YÖNETİM SİSTEMİNDE ŞART YÖNELİMLİ ZENGİNLEŞTİRME YAKLAŞIMI

Yıldız, Mustafa Halil
Yüksek Lisans, Bilgisayar Mühendisliği Bölümü
Tez Yöneticisi: Assoc. Prof. Dr. Ali Hikmet Doğru
Ortak Tez Yöneticisi: Assoc. Prof. Dr. Halit Oğuztüzün
Eylül 2013, 60 sayfa

Bu çalışma, iş kuralları yönetim sistemlerinin, hitap ettiği alandan ve/veya tasarımından kaynaklanan var olan verilerin zenginleştirilmesi durumunda tavsiye edilen "şart yönelimli zenginleştirme" yaklaşımı üzerine olmuştur. Bu yaklaşımda ilgili iş kuralının şart kısmının başarılı olmasını sağlayacak şekilde zenginleştirme adımındaki sorgulamaya limit koyularak zenginleştirme yapılması ve zenginleştirilen verilerin sayısını azaltarak performansın arttırılması hedeflenmiştir. Bu çalışma için, bu yaklaşımın uygulanmasına imkan veren kurallarının çokluğu sebebiyle "Türkiye Ulusal Sağlık Veri Doğrulama Sistemi" seçilerek geliştirim yapılmıştır, ayrıca bu sistemi geliştirirken "Sağlık Verisi Doğrulamasına Özgü Dil" alana özgü dil olarak kullanılmıştır.

Anahtar Kelimeler: İş Kuralları Yönetim Sistemi, İş Kuralları Yönetim Sisteminde Zenginleştirme, Alana Özgü Dil, Yazılım Mühendisliği

To My Best Friend and His Love, My Dearest Mother Sema, My Wife Merve,
My Father Hüseyin, My Mother Figen, My Brother Hilmi, My Grandfather Halil and
All My Beloved Friends.

ACKNOWLEDGMENTS

First, I would like to thank my supervisor Ali Hikmet Doğru and my co-supervisor Halit Oğuztüzün for their continuous support and guidance throughout my M.S. studies. They have pointed me in the right direction all the time.

I am grateful to the members of my advisory committee, Ahmet Coşar, Cevat Şener, Pınar Karagöz and Ümit Vardar for their support and invaluable suggestions in our meetings.

I want to especially thank to my Love, my Dearest, My Best Friend, My Teacher and Mother, Sema. She is always my biggest motivation. This thesis work would certainly have never been possible without her support and assistance.

I would like to express my gratitude to my wife, my love, my darling Merve, for all of her kindly and lovely supports and sharing my sleepless nights with me.

Furthermore, I want to thank my parents Hüseyin and Figen and brother Hilmi and my grandfather Halil who are my lovely, self-sacrificing, indispensable, indulgent family.

I would like to thank my team partner Tuğba Karakuş for her help on creating new DSL and I also would like to thank my another team partner Sertaç Demir for his help on designing orchestration of services on the service bus. I would like to thank my friends Kemal Aşık and Necip Karabulak for their helps on correcting my spelling mistakes.

Finally, I would like to thank to all my dear friends for providing their support every time when I needed

TABLE OF CONTENTS

TABLES

FIGURES

CHAPTER 1

INTRODUCTION

A Business Rule Management System (BRMS) is a software system that can be specified as a bridge between business and IT, which must have business rules and decision making parts. However, the need for separating Business Rule Management (BRM) from the rest of information system development process is not trivial. There exists some attempts to separate the decision making from classical software development, but most often, BRMS should be used at the backend and as a standalone system. This prevents typical BRMS from being used in multiple tiers such as presentation tier and content management tier. Another difficulty lies in the integration of legacy decision making sources such as existing business logic, web services, and database management systems.

By the way, Business rule management systems responded "faster reactive and proactive time to market" (we can call it 'Business Agility') request, which is the most common request from all of the clients for any kind of business, since information systems (ISs) are at the core of them, by providing the "experts define, developers use" approach. This approach also produced faster solutions to the specified problems and increased control over implemented decision logics.

Despite, developing applications without any programming effort foresighted, there has not been any solution for this yet. A developer is still needed to do the job even if it is a minor change. On the other hand, communication obstacles between the developers and business professionals are often found as the underlying cause for many IS project failures [2].

Since it is argued that business rules have a direct relation between project scalability by defining the business environment [2], business rules are found easy to manage and maintain, if represented properly.

On the other hand, business rule management and decision making are dedicated domain-specific aspects to software development, and they can be better modeled and implemented with domain-specific approaches. One such approach is abstracting domain-specific modeling with Domain-Specific Kits (DSKs). Basically, a DSK is defined as: "A DSK is a composite of a Domain-Specific Language, Domain-Specific Engine and Domain-Specific Toolset"[1]. In this study, because of combining these two (BRMS and DSK) together, Oracle Policy Automation toolkit is chosen for development of "Turkish National Health Data Validation System" with "Health Data Validation Specific Language" which are explained in Chapter 4.1.

In the execution part of business rule management system, there can be two conditions. In one situation, the data is enough for decision making and rule is executed according to coming data. In the second condition, the data is not enough for decision making and it needs to be enriched.

Nowadays, the subject of “Enrichment of Data” is not a big deal for the Vendors of BRMS (explained in Chapter 2.4). They suggest one of the methods explained in Chapter 2.4.2. They provide a lot of methods for enrichment process like web services, front-end forms, connections to databases… etc. as explained more detailed in their formal websites. On the other hand, the current approach for enrichment is delivering the data without any modification as its original status.

But in enterprise architecture that contains a BRMS, it is important to build the system architecture according to these two conditions. If there will be no need for enrichment than a standalone, BRMS can be enough, but if there will be a need for enrichment, then there must be at least one more tier (such as web services, front-end forms, database accesses) for the enrichment process.

In this study, a BRMS that needs enrichment process is implemented. “Calling a web service” method is used for enrichment. Over a service bus, enrichment results are connected to BRMS.

In the enrichment part, a new approach called "condition oriented enrichment" (In basic terms, if the rule is a binary rule, then this approach suggests adding limitation to inquiry method at enrichment process that will be corresponding to related business rule's condition, else it suggests to continue with the common enrichment process) is used for building the BRMS system faster and easier. The efficiency of this approach is also discussed in this study.

CHAPTER 2

CONCEPTS

In this chapter, Business Rule Management System (BRMS) and Domain Specific Language (DSL) will be explained. Advantages and problems will be summarized and usage of DSL with BRMS will be mentioned. While explaining BRMS, because of the Linear Inferencing Algorithm which is used in this study depends on Rete Algorithm, Rete Algorithm and Linear Inferencing Algorithm will be described. After the general concepts clarified, the concept of Enrichment will be explained.

2.1. BUSINESS RULE MANAGEMENT SYSTEM (BRMS)

2.1.1. What is a BRMS?

Business Rule Management System (BRMS) is a software system that serves as a bridge between business and IT, which must have business rules and decision making parts. According to [2], the terms for BRMS could be defined as:

Business Policy: A business policy can be defined as a statement which explains the direction a business will take.
Example:
Only a student which has fulfilled the requirements can enroll into an examination.

Business Rule Statement: A business rule statement can be defined as a statement which explains the structure or limitations in relation to the business.
Example:
In the case which a student has not successfully completed the examination more than three times, the board of examiners must hold a meeting. Moreover, the examination fee must also be paid by the student.
(The example shows that statements in comparison to policies are more detailed and explanatory)

Business Rule: A business rule can be defined as a statement which shapes or limits certain aspects in relation to the business. It has an atomic nature due to the fact that it can't be disintegrated into various business rules. In the case of attempting to break down the business rule, there will be a loss of meaningful information about the business.
Example:
In the case which a student has not successfully completed the exam

ination more than three times, the board of examiners must hold a meeting.
(The atomic nature of a business rule can be seen in the example above)

Business rules derive from business rule statements which are formed on a business rule statements basis, as can be seen in Figure 1. A formal expression style is used while stating the formal rule statements. [2]

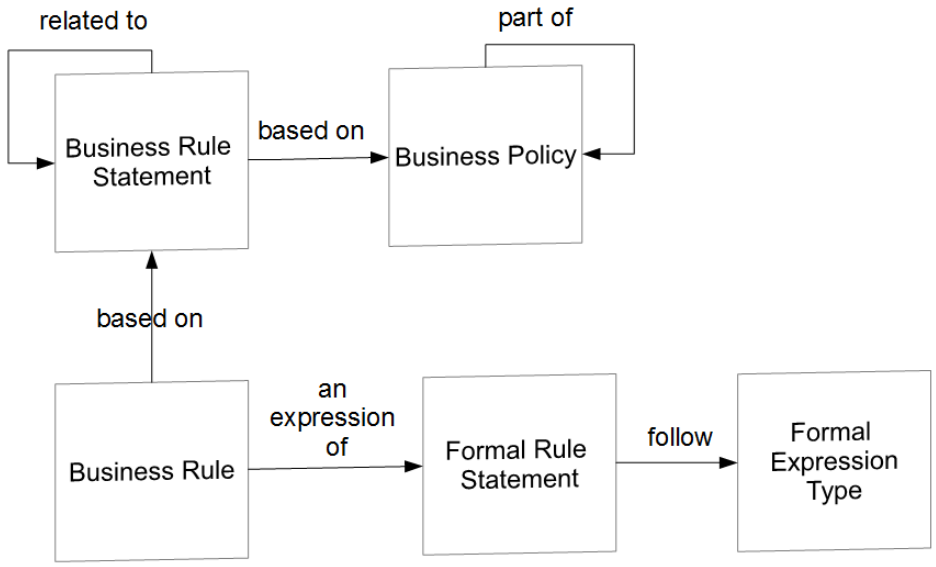

Figure 1 : The Origin of Business Rules (adapted) [2]

2.1.2. Old and New Approaches and Benefits

The old and new approaches and their benefits can be compared as following according to [14]:

Table 1: The older and more recent approaches [14]

Old Way	New Way
The system code of a simple application carries the rules deep within its structure.	Rules are not within the system code and are written in a clear and simple way.
Rules can only be used by the application and code which they were formulated for.	The ways in which rules are used are more diverse and they can be used by various applications or business processes.
Rule changes, which require reprogramming and can even involve reverse engineering, need IT help.	Rule changes are done more conveniently by the business users themselves without needing IT support.

Table 2: The older and more recent approaches [14] (continued)

Developing new application is time-consuming due to the fact that programmers must take into account all possible variables and conditions. This is not convenient for today's business environment so enhancement cycles generally follow the development cycles.	**New applications are developed rapidly due to the rules management system and the enhancements are performed by the business users themselves. Adding the new rules can be done whenever necessary, without disturbing the operation taking place at that moment, and with the rules engine solving how and when to use these rules.**
Adding rules carries risk because of the fragility of the system and difficulty of forcing them to operate functions which were not aimed for them to do as they were being designed.	Existing applications are used to call rules management services. Due to this fact, it is possible for legacy applications to extend capabilities without challenging their structure.
The extension of decisioning to new channels is possible with forming duplicate application that is transferred to the new environment.	Decisioning applications which are only written one single time can be modified across most operating environments and can be deployed and as an addition multiple channels can use this single application.
The developing and deploying codes is a hard task because it's difficult to code these business processes and requirements. It is possible for information to get lost due to the communication between the business user and the programmer. It has been revised a number of times before it is done correctly.	An IT business analyst familiar with rules language that is used in business policies applies the decisioning applications and services. Therefore, not much information is lost due to interpretation. Business users are also able to formulate rules using graphs, tables and Web forms.

2.1.3. Old and New Methods

Old Methods; can be summarized into two categories according to [2]:

i. Parameterization:

It is possible that the parameters are located in a configuration file or database which can be operated by using configuration tools. Applications can be applied to various environments and situations by the usage of parameter settings, with no need for programming requirements.

ii. Database Triggers and Database Procedures:

'Triggers' which are placed within the Relational Database Management Systems (RDBMS), after specific situations take place, are able to operate actions as a response. These specific situations are those related to when a record is changed, added or erased from the database.

The Structured Query Language (SQL) is used to write the actions which are performed by the triggers. These actions can be stored as one of the two following; the trigger body or database procedures.

The properties which have been expressed above show that the 'triggers' are an appropriate way for implementing business rules. The 'before-insert' trigger, as an example, is able to control the record (and reject it if it does not fit the business rule) prior to actually inserting it. Modifications can be performed apart from the applications without using the application logic because the implementing of rules is within the database.

The new methods are categorized into three groups:

i. **Database-independent tools:** The implementation of business rules takes place in a database by the usage of triggers and stored procedures. Although, the formulation of them takes place automatically and is controlled by development instead of a database tool, which can be seen in Figure 2.

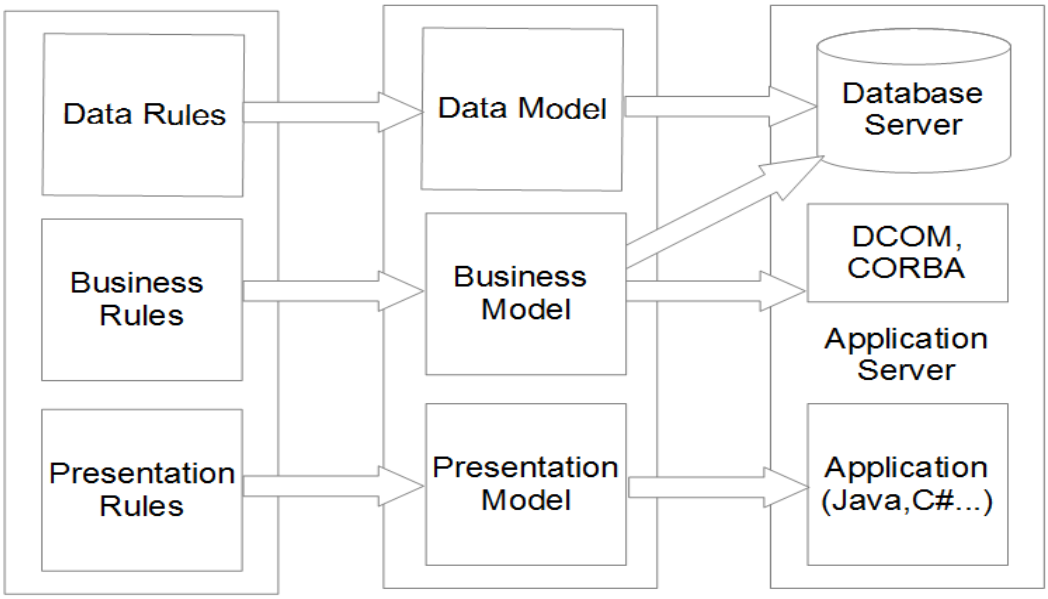

Figure 2 : From Rules to Application (adapted) [2]

ii. **Server-based tools:** Development tool generated business rules become middle-tier application services and are located within an application server, which can be seen in Figure 3.

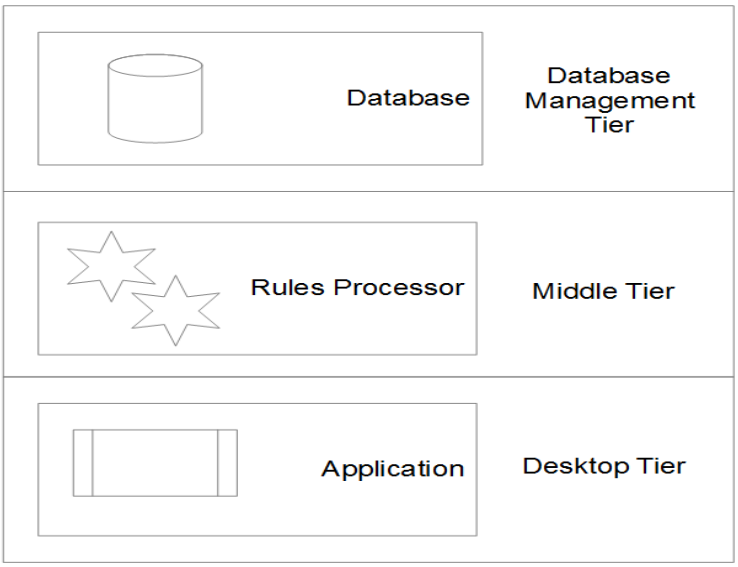

Figure 3 : Rules Processor: Middle-Tier Service (adapted) [2]

iii. **Rule-based systems:** With a logic-oriented approach the higher-level business logic and rules, which are related with various situations, are captured unlike the approach which designates limitations on data elements or tables. Special engine systems which operate the rules and form the necessary responses to them are used during the run-time which can also be seen in Figure 4.

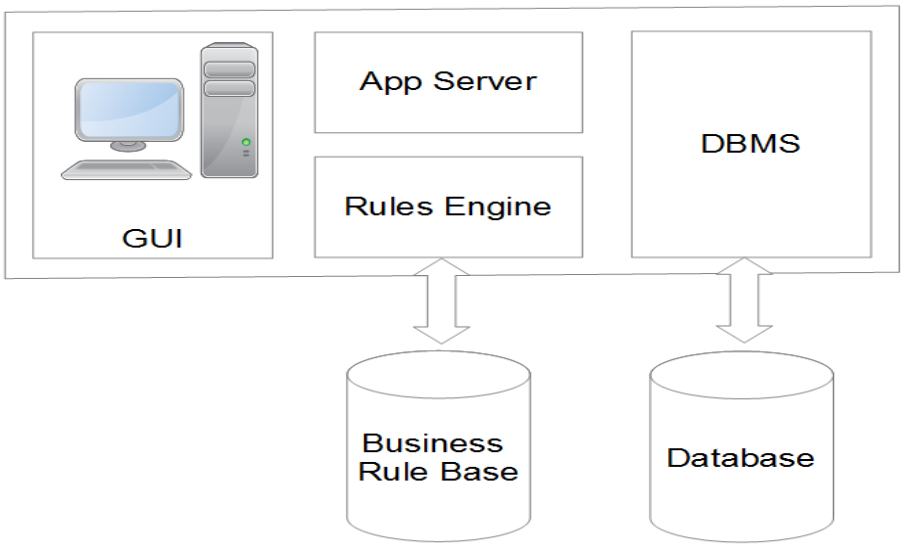

Figure 4 : Rule-based system (adapted) [2]

2.1.4. Market overview: Business Rules Platforms

This Chapter (2.1.4) is based on Forrester research [21] which contains "product and market presence data from 11 business rules platform vendors (Bosch Software Innovations, Corticon Technologies, CRIF, Experian Information Solutions, Fair Isaac Corporation (FICO), IBM, InRule Technology, Red Hat, OpenRules, Sparkling Logic, and WSO2)"[21].

i. State of the market:

- Increasing numbers are being reported by key vendors.
- There is an increase in the amount of revenues and customers.
- "Products have reached a point of perceived parity."[21] Many functionalities are shared by many products now.
- Vendors have found other categories which they can compete in. Other product categories have formed a new area in which business rules vendors have invested. Selecting a business rules vendor has become a more complicated decision due to the fact that business rules management is not a priority anymore and the product sets have capabilities which are not associated with business rules management primarily. This category includes:
  a. Rules are also given by BPM vendors. A business rules platform is not usually included within the suites of BPM vendors. The business rules are now used to integrate decisions with the aid of automated processes. The part which carries risk about the suppliers for the customer is that the BPM vendor's rule may not respond to the necessary needs of the business rule requirements. The problem with these providers is that they can do more than business rules platform, therefore, customers that are only interested in a business rules platform will benefit from purchasing a product which only does that.
  b. Business solutions are sold by a lot of business rule platform vendors, mainly depending on the technology instead of the development platforms. The major examples which mainly concentrate on financial services sectors are CRIF, Equifax, Experian Information Solutions, FICO, First Data, and MindBox. CRIF, Experian, and FICO use business rules development utilities in order to aid customers to give decision solutions which respond to the customer's needs. In order for the eligibility applications to be addressed Oracle has integrated Haley within its public sector solutions organization and IBM has generated a range of industry-specific solutions with ILOG.
- Market maturity has been observed by the signs of vendor consolidation. Between the years 2007-2011, SAP took control over Yasu Technologies, Haley was purchased by RuleBurst and then was taken over by Oracle, IBM swept up ILOG, and a famous auto parts company— Bosch Group — bought Innovations Software Technology.

ii. State of the vendors:

- After IBM purchased ILOG in 2008, they also adopted JRules which is a product of ILOG. For the past two years, the focus of IBM has been to combine ILOG JRules with IBM, business occasions and decision modeling products and also a number of Smarter Planet solutions in order to be able to make automate decisions. An updated emphasis has also been placed open JRules for z/OS by IBM.
- IBM ILOG has been provided a cheaper alternative by Red Hat JBoss. "Right on the heels of IBM's acquisition of ILOG, client inquiry spiked on Red Hat's JBoss Business Rules Management System (BRMS) and Drools (the open source project upon which JBoss BRMS is based)."[21] Developers that preferred a cheaper business rules platform

than IBM were interested in a JBoss product. Although JBoss Enterprise BRMS (and Drools) is an effective business rules platform customer, it does not have the various authoring utilities that Bosch Software Innovations, Corticon Technologies, IBM ILOG, FICO Blaze Advisor, and others have. Likes of Corticon and IBM have been working rapidly to be able to provide business experts authoring tools for developers that use a subset of the logic in Java applications but JBoss has not been able to compete with their speed. The JBoss Enterprise BRMS and serve oriented architecture suite (SOA) of Red Hat is combined.

- InRule Technology is a choice for the .NET platform. Since Microsoft does not own a business rules platform, InRule has become the selected option for .NET shops that requires this technology. Companies such as Corticon, IBM ILOG, and FICO also supply business rules platforms for .NET, but they are more common in enterprise Java architectures in comparison to the .NET architectures. The version 4 of InRule is designed in order to be used in applications which are larger and to be used in tools of particular business domains.
- Over the course of the last 7 years, Corticon Technologies has changed its clientele from independent software vendors towards enterprises. Again Corticon is focusing on its enterprise customers and has started a partnership with HP. It is also important to note that Corticon does run the risk of being owned by HP. Although a .NET environment does exist within Corticon, it is possible to say that the main products among its enterprise customers are the Java products.
- FICO's ambition is based on the company aiming at decision management solutions, not integration of business rules platforms, decision modeling, linear programming and business rules execution.

iii. Summary

The aim of business rules platform developments are optimizing and integrating. Research shows that the reasons which customers choose business rules platforms by believing that the business utilities for the business professionals they will be strengthening to their business. Authoring tools have been offered by large vendors although many of them also need a lot of help from expert application developers which results in two different ways:

a. "Demand for products that empower business experts to minimize developer involvement."[21]
b. "New opportunities for independent vendors that innovate in business-expert features."[21]. The fact that the major vendors do not push themselves to be innovative in business expert tools (with the exception of OpenRules and Sparkling Logic) has created the opportunity for independents. While OpenRules chooses to use a Devision Model method, Sparkling Logic prefers to integrate three characteristics in order to encourage business experts to author rules independently: software-as-a-service solution, social collaboration and decide-by doing tool which saves time and money.

Major vendors are changing four developments in their main business rule platforms:

a. The recently developed human-interface technologies which will strengthen business experts. Web 2.0 created a human interface technology in order to provide web-based business rules authoring for customers. Although in the past most business had to use Microsoft Windows desktop application for similar functions, Corticon, FICO and InRule rewrote their human interface layers integrating Web 2.0 in it.
b. Independent software vendors (ISVs) will be more willing when easier integration is possible. By permitting ISVs in their product to generate their authoring tools, time will show whether the partners will be more willing to use the rule engines in their own

products. Although Jess is currently offered, Drools and OpenRules are the favorites for OEMs as open source products.

c. Integrating other products in order to enrich and enhance rules. "Integration has been a major thrust for the ILOG team inside IBM for the past two years; the integration of ILOG with BPM and events platforms was readily apparent at IBM's Impact 2011 conference."[21]. While Bosch Software's integration with Java application products have given them the possibility of using Visual Rules generated Java codes in larger applications. Also, FICO has developed its integration with Blaze Advisor and other properties of the decision management suite which has resulted in the handoffs being less apparent and making it much more practical.
d. "Better performance and manageability for large-scale implementations."[21]. Two meanings come to mind when large-scale is said in terms of business rules platforms. "First, each of the big vendors has brought the expected incremental improvements in raw rules-processing performance in recent releases."[21].The performance optimizations effects differ according to the real applications. When past experiences are examined, it is possible to say that most performance issues within rules applications are caused by the data architecture, not the rules engine performances. Secondly, recent tools that deal with large numbers of rules were generated by Bosch Software Innovations which were then brought to parity with the other strong vendors.

2.1.5. Rete Algorithm

"The Rete Algorithm is an efficient method for comparing a large collection of patterns to a large collection of objects. It finds all the objects that match each pattern. The algorithm was developed for use in production system interpreters." [6] It was designed by Dr Charles L. Forgy of Carnegie Mellon University, first published in a working paper in 1974, and later elaborated in his 1979 Ph.D. thesis and a 1982 paper. [16]

In Rete Algorithm, there are Rules, Rulesets and Facts.

The rules are usually "if <condition> then <result/action>" form. Set of rules forms Rulesets.

Example of a sample rule:

If the temperature of water as Celsius > 40 then assign status= "Hot Water"

In this example "If, then, assign" words are keywords. Condition part of the rule is "temperature of water as Celsius > 40", and result part of the rule is "status= Hot Water" (assigning operation). There can be more than one condition joined by logical operators and also there can be more than one result.

In the above example, a data is needed for a decision, which is "temperature of water as Celsius". It is called that the data need for decision as "fact". Facts are unique variables or definitions of condition parts.

In order to fulfill the requirements of Rete Algorithm, rule sets must be provided to the rule engine to continue its processing. Each rule is matched with its information in the rule set in order to determine if the rule will be executed or rejected, which is called pattern matching that is a process that occurs over and over again. During each cycle, changes can be done within the information such as the addition or omission of information or facts from the lists. Unsatisfied patterns may turn into satisfied patterns after the modifications are made, additionally the rule sets which are satisfied must be kept that way and be updated during each cycle. Most of the time, only a few minor modifications are made with the rules which are within the list which is called temporary redundancy. If each rule is checked by the rule engine in order to search for all the information whether it is changed or not, this would

cause the process to experience a loss in its speed. By only computing the added or deleted facts by remembering the former modifications, this unnecessary action is avoided. This work is done by Rete algorithm efficiently. [22]

The implementation of The Rete algorithm consists of a network of nodes. In this network design, each state of the matching process from cycle to cycle is saved and re-computations are only for modified facts. If a new fact is added or some facts are removed then the state of the matching process is updated in this algorithm. As it is usual, the matching process speed is directly proportional with the number of added/removed facts. [22]

Each rule has an inference cycle. In inference cycle, rules are selected from a base called knowledgebase and then matches with contents on working memory and lastly they are executed. To summarize, there are three phases: match, select and execute. [3], [5]

During the matching process, rules and facts are matched in order to decide on whether a rule will be executed or not. The rules that have the necessary conditions are placed separately into a list which is called agenda for firing. A rule is chosen from the list to be fired or to be executed. The reasons for which the specific rule has been chosen may differ from its priority, recency, specificity or different criteria. The chosen rule is then operated by performing the actions that placed on the right of the rule. [22]

Figure 5 summarizes these phases:

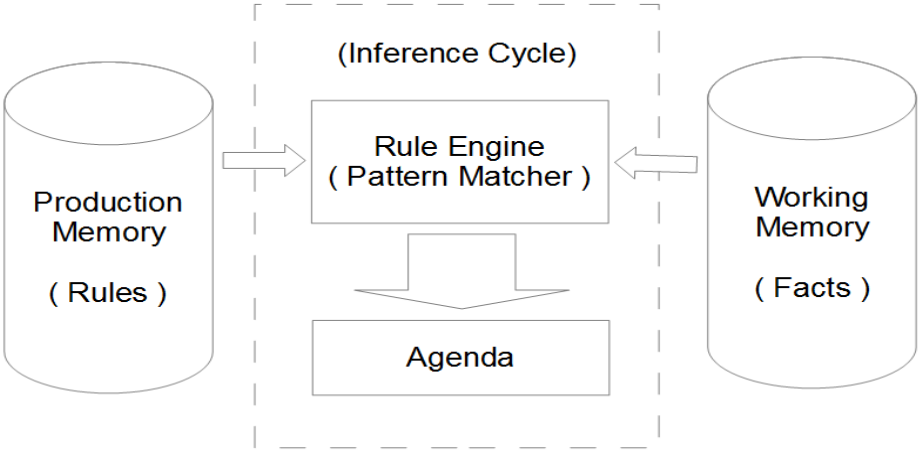

Figure 5 : Pattern matching: Rules and Facts [22] (adapted)

Rete carries actions with two types of memory which are able to change in time, production memory (PM) and working memory (WM). The working memory reflects information on the current situation the system is in, the situation of the outside world and the system's problem-solving situation. A dataflow network is what Rete makes use of to represent the situation the productions are in. [5] The network mentioned above is a direct acyclic graph which has nodes which include representing patterns that represent the situation of the rules. The nodes function is to test the tokens and determine which ones pass and send only the ones that do, in a way it functions like a filter. [22]

The rete network consists of two parts: alpha network and beta network.

The ongoing tests which test the working memory properties which can also be called inputs are done by the alpha part. Alpha memory (AM) is a storage where the working memory elements which are successful to be filtered pass the test of specific conditions are stored. Recalling the facts which have matched is also the function of this AM.

The networks beta part specifically includes join nodes and beta memories. Test that analyze the consistency among variables in specific situations are carried out by the join nodes. Partial instantiations which are WM elements that are matched with some of the rules conditions and are called tokens are stored within the beta memories. [5]

The Rete network begins with the Rete Node which is then followed by the kind nodes which exists for each fact type. An alpha node for each pattern is generated after this process and then it linked to a kind node. By asserting a fact, a token is created and then the tokens insert the root node. A new branch is divided by the network for each token type. A token's copy is given to each kind of node and then SELECT operation is performed in order to select the token which are from the same kind. The kind node passes on the copy of the token to the alpha node. After that, the alpha notes perform a PROJECT operation which includes extracting components out of the token and matching them with variables that are included within the pattern. It can be said that the alpha node functions as an evaluator. Then the beta nodes decide on a cross product for each rule and the final process is for the rule to be executed. [22]

Example:
Consider a rule like;

If height (in cm)>160 or height (in cm)<5 or weight (in kg)>30 then price = 100.

Assuming height is java variable and weight is a json variable. Rete network in Figure 6 can be created to represent this rule.

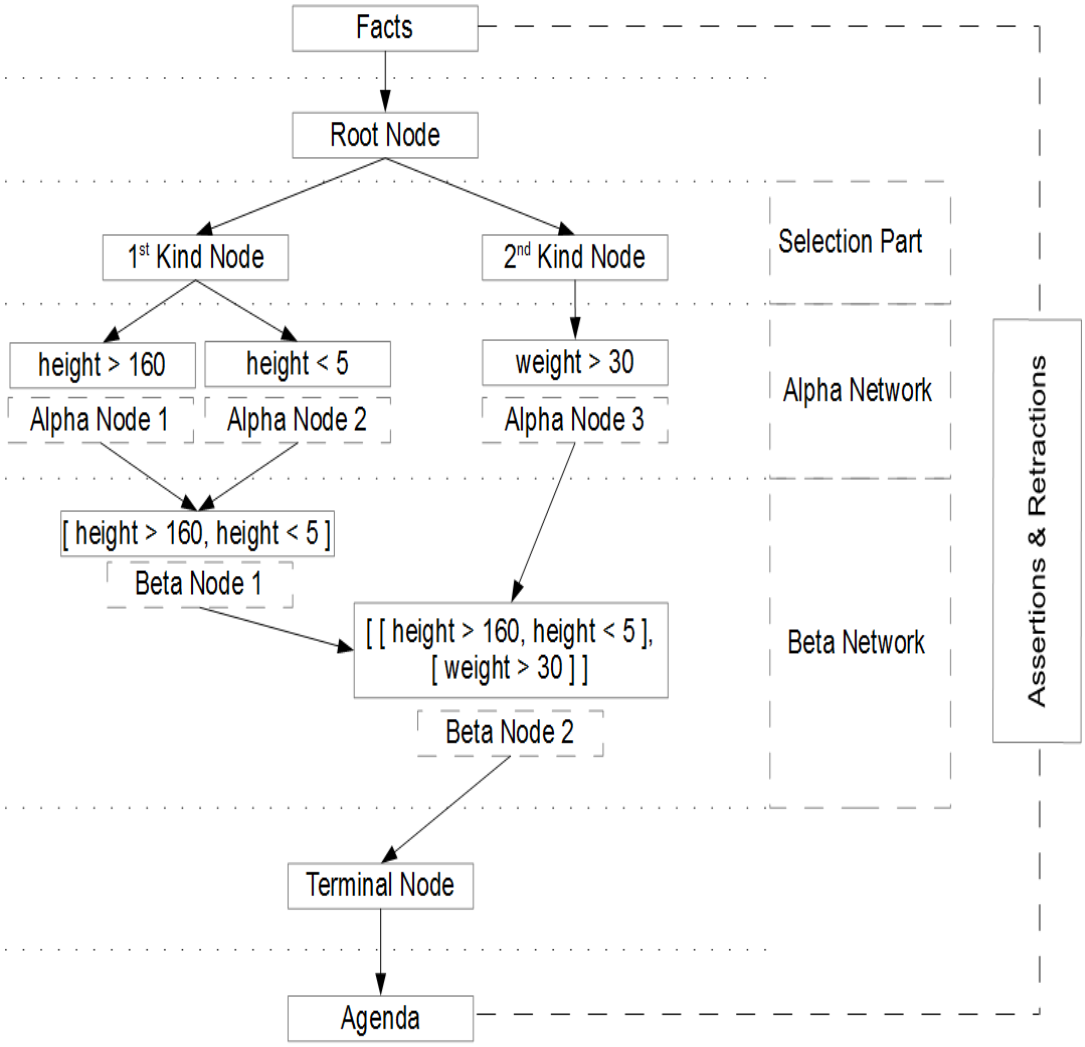

Figure 6 : Sample Rete Network

In the sample Rete network seen in Figure 6, because of two types of facts: Java and Json, there are two kinds of nodes. 1st Kind Node represents Java type and 2nd Kind Node represents Json type.

Three alpha nodes created because there are three patterns: height>160, height>5 and weight<30 . Alpha Node 1 and Alpha Node 2 is representing the first two patterns which are connected to 1st Kind Node. On the other hand, Alpha Node 3 is representing the third pattern which is connected to 2nd Kind Node. First two alpha nodes are joined by Beta Node 1. The Alpha Node 3 and Beta Node 1 is joined by Beta Node 2.

When a value for "height" is entered to the root a token will be created. Copies of this token will be passed to kind nodes. The 1st Kind Node will accept it as the fact type is Java. This token will be passed onto Alpha Node 1 and Alpha Node 2. If the value satisfies the constraint then the token will be passed onto Beta Node 1 and then will be passed to Beta Node 2. In the mean time value of "weight" is entered to the root and its token will be accepted by 2nd Kind Node. Alpha Node 3 will receive it and check if the value satisfies the constraint, "weight > 30". If yes then it allows the token passing onto Beta Node 2. If the

fact, that is the values, match with the condition in the Beta Node 2 then the rule will be added to agenda for firing.

Rete has a couple of significant features which make it work rapidly in comparison to native match algorithms [5]:

a. It uses state-saving. After the alterations are done on the working memory, the results from the matching process are saved within the alpha and beta memories. Subsequently to the next alterations, most of the results are not changed, so Rete ignores recomputation by leaving the state within the successive working memory changes.
b. It shares nodes between productions that have conditions which are alike, which forms its second significant feature. The sharing differs according to the part of the network, sharing can also take place within the alpha network. In the alpha network which multiple productions have conditions which are alike, Rete uses only one alpha memory for each independent condition avoiding duplicating memory for each and every production. Also in the beta network, when multiple productions are alike within the first conditions, nodes which are identical match the conditions which avoid duplicating match efforts for the productions. [5]

The main disadvantage of Rete is its memory usage as can be specified from (a). However (b) helps for decreasing the memory, it still needs considerable amount of memory for saving state of the system.

And by the time Dr Charles Forgy (designer of Rete[6]) developed new and more successful algorithms depends on Rete, which are named Rete II and Rete NT.

According to [17] Rete-NT algorithm is at least 500 times faster than the original Rete and 10 times faster than Rete-2.

Test Conditions and Platform are explained in [17] as:

"The Waltz, WaltzDB-16, and WaltzDB-200 tests use a series of lines as 2-D end points to build a 3-D box. DB-16 and DB-200 build 16 and 200 boxes respectively, with each box constructed of 20 lines and obeying a number of constraints."

As an example, when the lines have been matched up, the lines are used to construct the boxes. 35 complex rules are used in this process.

While in the early 2000s, deciding on the most suitable way to construct 16 boxes took a number of minutes by a machine, not only it takes less than a second to do the same action but also takes over 17 seconds to construct up to 200 boxes. The 200 boxes are constructed in about 2 seconds by the Rete-NT engine. The table below shows approximately how performance differs for each rules execution engines. Dell PC with Core i7 CPU including a 12GB RAM was used as a testing platform; and because of not having enough time and vendor support Blaze Advisor, JRules and Drools could not be tested.

Table 3: Rules execution benchmark results[17]

	WaltzDB-16	WaltzDB-200
JBoss Drools (Rete)	0.994	Not tested
IBM/ILOG JRules (Rete)	0.640	Not tested
Fair Isaac Blaze Advisor (Rete-2)	0.305	Not tested
OPSJ Rete-2	0.262	17.466
OPSJ Rete-NT	0.433	2.362

2.1.6. Linear Inferencing Algorithm

Linear Inferencing algorithm was built over Rete. The Oracle's BRMS "Oracle Policy Automation" uses this algorithm for forward-chain inferencing. According to the claim in a paper of Oracle "linear inferencing significantly outperforms the Rete algorithm"[11]. The performance benchmarks in [10] are very impressive.

Actually there are no fundamental algorithmic advantages of one engine over the other. This comparison can be more meaningful if it is over implementation efficiency.

The Basic Approach of the Linear Inferencing Algorithm:

The procedure which recent computer processors operate is exploited by the linear inferencing algorithm. The algorithm especially functions by serializing the inferencing process while maximizing the usage of large processor memory caches.. The algorithm functions by ordering rules to be processed during every forward inference cycle by doing left-to-right sweep. The amount of data which is altered during each cycle does not change the fact that only one sweep of rules is necessary. [11]

The decreasing of the processor cache misses, the access to rules by the memory efficiently is increased. The path which rules are usually accessed from is the processor's high-speed onboard memory which forms the most significant feature of the algorithm due to the fact that mostly legislative and policy rule sets do not usually fit with such caches. [11]

The subsequent access to rules also indicates data structures which are more space-efficient which enhances the usage of onboard processor cache. In addition to the information above, the rule accesses are read-only, so it's possible to share a cache copy across more than one inferencing sessions. [11]

A Comparison with the Rete Algorithm:

The highly popular Rete algorithm has a data flows network which works with the goal of decreasing the number of operations during every inference. In order to succeed in this it is necessary to have data structures which are large and complex. One of the primary things that should be executed by the rule engine is to copy the logical structure of the rules to be used for every inference session. Every single modification within the data means that a 'walk' should be done within the network due to the fact that too many changes cannot be executed at the same time. Every walk of the network needs nonsequential, mutable access to memory. As a result, an increase in memory requirements and poor memory access locality leasing to an increased missed memory caches. Therefore, it is difficult to handle the complex logic within legislation and policies. [11]

Extensions to the Basic Approach

Oracle's basic linear inferencing approach with the following extensions makes further performance with Oracle Policy Automation. This chapter is based on the reference [11].

i. Incremental Inferencing

The mentioned approach has been developed into mechanism responding to incremental inferencing. The rules which are relevant for that particular session are examined whereas the rules that do not have any effect on the cycle are ignored. The extension has a read-only table which is shared that contains a map from data items to dependent rules and a bitmap which has the relevanci situation of every single rule.

ii. Cyclic Dependencies

Loops are within almost all rule sets. As an example,

- If the person is dead, then they are not alive
- If the person is alive, then they are not dead

These cycles order rules in a way that avoids their execution to be sequential. Although cyclical rule sets are generally not large, they may include multiple rules.

"Oracle handles cyclic dependencies by treating each loop as a composite rule consisting of the minimal set of rules that fully contain the loop."[11]

The set of rules are then ordered in a sequence with the other rule sets. During the process of ordering, composite rules are taken as indivisible, which leads to the sets being insignificant; the only relevant thing is the composite dependencies of the cycle upon the rule set.

After that, the inferencing takes place like before by a left-to-right sweep. The combining rules of a composite rule are processed by the usage of a brute force approach. A brute force approach follows a localized manner in order not to affect the memory access performance of the algorithm.

iii. Conditional Branching

Deep instruction pipelining is a significant approach used by modern processors to enhance their performance. In this strategy each part of a processor is working at the same time because the instructions for execution are overlapped.

A risk which is carried by instruction pipelining is that conditional branching may result in stalling of the pipeline when the processor is unable to understand the following instruction during the execution. In order to achieve high performance from the processor, the frequency of the unpredictable conditional branches must be reduced.

If conditional branching in the rule evaluation is ignored during linear inferencing, it can result in dividing of the performance. The linear inferencing that Oracle does involves the evaluation of rules into a sequential bitwise logical operation and table lookup. By doing so, conditional logic is eliminated which leads to the decreasing of pipeline stalls and boosts Oracle Policy Automations' performance.

Advantages of Linear Inferencing

There are a number of benefits of linear inferencing in the Rete algorithm which is widely used by rule engines.

- Numerous modifications made to the input data are smoothly handled. This provides a more appropriate solution to real-life processing, such as transactional processing and batch data which has been uploaded via a database or interactive applications.
- The use of memory for each session is reduced. The amount of memory needed to execute heavy applications decreases.
- It makes more use of modern processor architectures. For a better performance, onboard processor cache is raised by the space-efficiency of linear inferencing data structures.

2.1.7. Advantages of BRMS

Some of the advantages of BRMS can be summarized as following from [13]:

- The need for IT usage decreases. The main advantage is that it allows rule developers that are not software engineers to write rules without relying on IT technicians. This is a major benefit to the organization because deployment of the rule is faster than traditional software environment. Before every rule is renewed or updated, Testing and Quality Assurance must be a priority to be certain of the integrity of the system.
- The way rules are written and controlled is up to the business. Now that IT is not a necessity, editing the rules and the rule lifecycles are controlled by business managers. The complex process is simplified because one department is not involved within the process.
- "High correlation of business vernacular to rule development."[13] Anyone who is accustomed to the business is able to read the rules without being in need for help from IT.
- For the deploying of rules automated processes can be installed. This is inestimable for responding to changes and competition within the market.

2.1.8. Problems with BRMS

Some of the disadvantages of BRMS can be summarized as following from [13]:

- Specialty is necessary for enhancement and applying. Employing such employees with the specific qualifications is expensive for the business.
- "Extensive development cycle is the norm"[13] – It is a hard task to form a BRMS since it needs an "Object Model, rule harvesting, organization and template rule creation and system design and development with a good security model."[13].These processes are not all the same to all vendor BRMS implementations and change according to every development environment.
- IT departments could always be required even in basic tasks such as implementing a new rule conditions. The same goes for when a new rule is implemented. It is possible that the necessity for IT department may always exist.
- The rule engines reduced effectiveness while handling computations and resulting in a failure while trying to templatize codes are two of the main reasons which BRMS is not a dream environment to write complex algorithms. If the IT department struggles to implement, it's obvious that it will be difficult for the rule developers, too.
- "Perturbation of the Object Model may have system wide implications"[13] – after several BRMS training experiences it can be said that something as basic as an enum can harm the software development which will then be extremely time-consuming trying to fix. Although this is relevant for a number of general software processes, it's particularly correct when it comes to BRMS because data is sent using an object model to the rule engine. Since the circumstances and actions depend on the model, if there is any change done to the model, this indicates that the rules will also change.

- "Strong coupling to a particular BRMS vendor"[13] – whether or not the BRMS used by the system will function successfully is related to the BRMS vendors functionality and its characteristics. The cost of deleting this product in terms of finance, development, efficiency and time is almost non-existent.

2.2. DOMAIN SPECIFIC LANGUAGE (DSL)

2.2.1. What is DSL?

Martin Fowler explains DSL as following: "Domain-specific language (noun): a computer programming language of limited expressiveness focused on a particular domain."[7]

And key elements of DSL and definition of DSM are explained as following in [8]:
The definition consists of four main components:
Computer programming language: DSL is what is used by people to make a computer do a task. Like most modern programming languages, it's understandable by humans and is also constructed in a way that enables the computer to execute it.

Language nature: The DSL must be fluent in terms of both separate expressions and the collection of these expressions.

Limited expressiveness: Varied data, control and abstraction structures are some of the many utilities that a general-purpose programming language supports. Although this is beneficial, it is difficult to understand and to be able to use it. A DSL can only provide for a small number of features necessary for the domain. It can't build a system as a whole, instead it provides for only a specific aspect of the system.

Domain focus: The only benefit of a limited language is when it concentrates on a smaller domain. A limited language only becomes valuable with the domain focus.

Domain Specific Modeling uses a language which uses a particular problem domains' concepts and rules. The final products are created within a selected programming language or a different high level specification. The reason the automation can be done is that the language and generators needs are compatible with a single company or domains requirements. While experts define them, they are being used by developers.

As seen from definitions, Domain Specific Languages cannot be thought separate from domain specific modeling. Actually it can be thought as DSLs are results of abstractions or a way to implement Domain Specific Modeling. If the domain is specific and models are specified, a language can easily be build with these models.

2.2.2. Architecture of DSL Processing

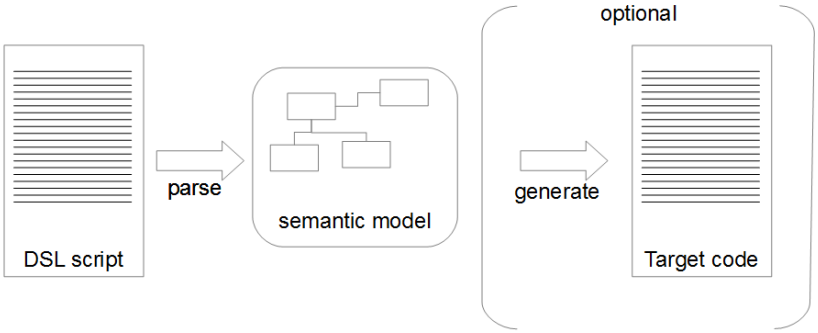

Figure 7 : The overall architecture of DSL processing (adapted) [7]

DSL will be explained according to [7] in this chapter.

DSL can be described as a thin layer over a model which in this context is the Semantic Model pattern. The pattern aim is to catch the significant semantic behavior into the model and the DSL functions by populating the model by using a parsing step. This indicates that the Semantic Model has a central role.

The Semantic Model can be seen as a model which integrates data and processing or can be used simply as a data structure. Since DSL cannot handle each and every part of the Domain Model in the most fitting way, the Semantic Model of the DSL functions as a subset of the Domain Model. DSLs may have other functions aside from populating Domain Models. Just like any other object model, the Semantic Model can also be altered or manipulated.

One of the main reasons for the wide usage of Domain Model is to capture a software system's main behavior. A large part of a Domain Model consists of the DSL. Domain Model and Semantic Model can be viewed differently. The Semantic Model generally functions as a subset of the Domain Model because the DSL is not always able to deal with the Domain Model. Also, it's important to note that DSLs are useful for other reasons than populating a Domain Model.

Just like all object models the Semantic Model can be manipulated as the user wishes. Separating the Semantic Model and the DSL has its benefits. One of the main advantages is that you are able to process the domain semantics without needing to struggle with DSL syntax or parser. DSL is usually used by people which are representing something which is complicated. Since it has own model, it provides you with the opportunity to test the Semantic Model with objects which can be manipulated within the model. Numerous states and transitions can be formed to see whether or not the events or commands execute, with no need for struggling with parsing. With no need to have any knowledge about the parsing process, in any case of problems, the problem can be isolated from the model.

Basically, Semantic Model enables you to evolve the model and language independently. In the case that we feel like modifying the model, without altering the DSL just by the addition of the required constructs, this can be accomplished. As another solution new DSL syntaxes can be tested and see whether they create the same objects for the model. A comparison can

be made between the two syntaxes in order to have an understanding of how the Semantic Model is populated by them.

After obtaining a Semantic Model we have to be able to control it to do the tasks we want it to do, which can be done in two ways:

1. Executing the Semantic Model (which is code and can be runned) is the easiest and most beneficial way.
2. Code generation, which is when a code is compiled and, can be used. According to a few circles, code generation is one of the main necessities of DSLs.

Code generation is most useful in situations in which the place where the model will be run and the place that the DSL will be parsed are not the same. To execute codes with a language-limited environment like limited hardware or a relational database can be given as examples. The parser and Semantic Model are implemented in a language that is fitting and C or SQL are generated. A similar situation occurs when unwanted library dependencies are within the parser. This situation takes place often if a complex tool for DSL is used which is the reason that language workbenches are used to generate codes.

Semantic Models enables people to test executing the DSL without needing to fully understand how generating codes actually function at the same time. Parsing and semantic can be tested more rapidly and can isolate problems because code generation is not required. It is possible to run a search for errors before code generation actually takes place.

One of the main concerns about the code generation is that although it is in an environment that can be implemented to Semantic Model easily, it is difficult to be understood by most developers. Code generation from Semantic Model is much more explicit which may cause a problem with teams that do not have capable developers. Code generation is not mandatory rather than being an optional part of the DSL. It is a feature that is incredibly significant if we are in need of it, but in general it's not a necessity.

By the use of code generation another advantage of using a Semantic Model is discovered; the fact that it is able to couple code generators from a parser. With no knowledge about the parsing process, a code generator can be written and tested which is one of the factors that makes the Semantic Model appealing.

2.2.3. Advantages of DSL

Advantages of a DSL can be summarized in five items:

i. Productivity[8]:

The quality of productivity is linked to the level of abstraction. This not only means the time and resources required to create a product but it also means its protection. Companies prefer not to share their DSM solution which highly increases the productivity because it is beneficial for them to stand out within the competition. It has been observed that in comparison to general-purpose modeling languages or manual coding practices, domain-specific approaches are more productive between 300-1000% more productive. [8]

ii. Communication with Domain Experts[7]:

One of the struggles in software projects and one of the main reasons of being unsuccessful in these projects is the difficulty of user-customer interaction and communication. DSL comes in useful in this communication barrier by using a clear and simple language in relation to the domains. [7]

iii. Change in Execution Context[7]:

Compile time could be when definition is evaluated instead of runtime. To run a code in a separate environment is one of the key reasons of DSL usage. [7]

iv. Test During Development[8]:

Applications which are qualified are possible with automation with DSM by prohibiting errors from occurring rather than adding tests to increase the quality while the specification is taking place.[8]

v. Documentation:

Almost no documentation is required when DSLs are selected. Documentation with DSL is not difficult because it is similar to natural language.

2.2.4. Problems with DSL

Problems of a DSL can be summarized in four items according to [7]:

i. Language Cacophony:

The major struggle about DSL is called the language cacophony problem which indicates the idea that learning multiple languages is a more complicated and hard task in comparison to learning one single language. Requiring knowledge of more than one language also complicates working on the system and hiring people to work on the project.

ii. Cost of Building:

With the writing and its general maintenance a DSL still is costly. Although it has its advantages, it doesn't mean that each and every library will benefit from DSL. If an API can fulfill and complete the necessary jobs, an additional API is not needed. It may be unnecessary and will not benefit although it has its advantages.

iii. Ghetto Language:

"The ghetto language problem is a contrast to the language cacophony problem. Here, we have a company that's built a lot of its systems on an in-house language which is not used anywhere else." [7]. This is challenging in terms of hiring staff and to be constantly updated with recent technologies.

iv. Blinkered Abstraction:

One of the benefits of using DSL is the fact that it ensures an abstraction which can be used while dealing with one specific subject matter. It is a significant factor because the behavior of a domain can be stated more clearly than lower-level constructs.

Both DSL and models have the risks of placing blinkers on the thinking process. A blinkered abstraction forces more time and effort to be spent on placing the world into the abstraction rather than fitting the abstraction into the world. This can be observed when something which does not fit within the abstraction is faced and wastes time, rather than modifying the abstraction in order to adapt the new behavior within it.

2.3. BRMS with DSL

The idea "The more business specific logic you can abstract out of an application, the less programmer involvement you will need to alter the business logic."[15], points a solution for a better business rule management system: More abstraction in BRMS.

Separating business rule management from the rest of the information system model is not so easy however; it becomes a need for developing more effective business rule management systems with less cost.

However, Domain Specific Modeling and Domain Specific Languages come with lots of similarity and ease to this separation subject:

DSM is a kind of software engineering methodology aims raising the levels of abstraction and improvement of productivity. This is basically done by hiding today's programming languages more and more, getting rid of its complexities at the first glance of production. On the other hand, "DSL is a programming language or executable specification language that offers, through appropriate notations and abstractions, expressive power focused on, and usually restricted to, a particular problem domain" [4].

On the other hand in Table 3, it is seen that standalone BRMS covers standalone DSL in most cases:

Table 4 : DSL and BRMS [23]

Needs For DSL:	How BRMS handles:
DSLs must be directly executable on a specific platform.	Business rules management system handles this.
DSLs must be extensible by existing GPLs [General Purpose Languages]	Most BRMS products do this inherently
DSLs must be understandable by both developers and domain experts.	This, of course, is a key value of using a business rules approach and a BRMS
DSLs need models	BRMSs have business rules and can be specified as domain models

So instead of comparing DSL with BRMS, using a DSM approach with a DSL over BRMS can separate business rule management from the rest of the information system and by the way abstraction will be done with this methodology.

This solution is preferred for implementation in Chapter 4.2.

2.4. ENRICHMENT

2.4.1. What is enrichment?

In the execution part of business rule management system, there can be two conditions. In one situation the data is sufficient for decision making and a rule is executed with the coming data. In the second condition data is not enough for decision making and it needs to be enriched. This enrichment process can be done via web services, front-end forms, databases …etc and the current approach for enrichment is delivering the data without any modification to its original status.

If a software system will use a BRMS, it is very important to build its architecture according to these two conditions. If there is no need for enrichment then standalone BRMS can be enough, but if there is need for enrichment then there must be at least one more tier (such as web services, front-end forms, database accesses) for the enrichment process.

Actually, "Enrichment" is a known process mostly used in BRMS and Event Driven Architectures. If there is a need to know some not existent data in a BRMS or Event Driven System, the process of accessing that external data (or other constituents) is called "Enrichment of data/fact/message/value …etc". In one of the Oracle's Product's White Paper [22], Enrichment on Event Collection is also explained as "finding additional data":

> "The event collection, tracking, and enrichment process occurs prior to the event rating process. This process ensures all customer activities, collected from any external source, are tracked accurately in the system and enriched to include additional data required for the rating process."[22]

Let's explain enrichment with a simple example. Consider there is a rule like:

> If age of person < legal adult age limit
>
> then person is child

In here, the value of the fact "age of person" will be known in runtime, but the legal adult age limit is not a known value even for the BRMS. This value must be extracted from some another system. This system can be a database, a web service, a form on screen… etc. and this finding operation is called the enrichment process.

One more detailed example is shown in figure 8:

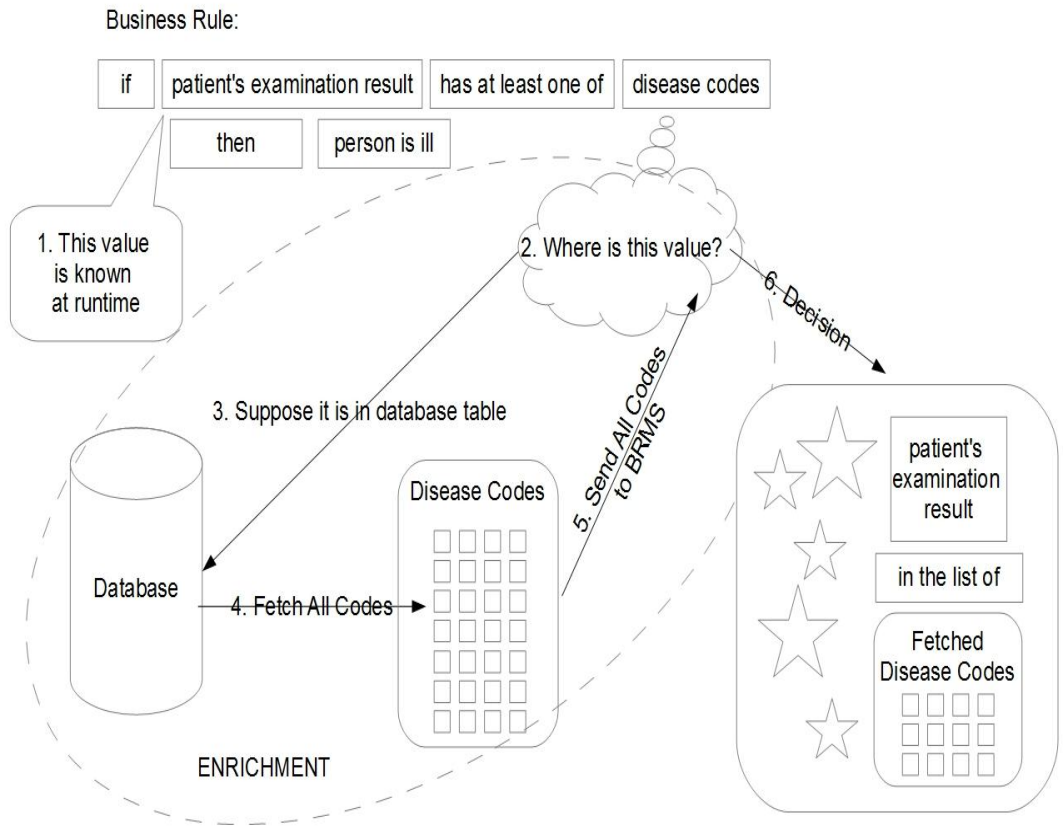

Figure 8 : Example of Enrichment Process

In this example, "disease codes" are not known and must be enriched from a database, so there must be an interaction between the BRMS and the database for fetching disease codes. After this fetching operation and sending the codes to the BRMS, the rule can be executed to look up if "patient's examination result" is in the list of "disease codes". This enrichment operation can be called Common Enrichment Approach.

This enrichment process can be done with web services, front-end forms, database accesses etc. Because of the complexity, it can be conducted as a service. For communication of an enrichment service and the BRMS, there are two methods which are explained briefly in one of the papers by IBM and there can be one another method that uses these two methods together. These three methods are explained at Chapter 2.4.2.

2.4.2. Enrichment methods

There are three main methods for implementing enrichment process in enterprise architectures. First and Second methods are also explained in [20] with more detailed examples.

First approach is completely separating the BRMS and the Enrichment Service. In this approach, because of knowing what the rule needs, when the message (or the data on which the rules will be executed) comes, it is sent to the enrichment process. All necessary

enrichments are done in a completely different service and the result is sent to the BRMS after enrichment. Because of the enrichment data can be anywhere according to the need (for example enrichment data can be in a database or there can be need to take it online from a screen), this service can be anything like a web service, a form on screen or some business logic over some data… etc. An example architecture targeting this approach is depicted in Figure 9. In this figure, enrichment needs existing data in the database and is requested through a web service. The connection between enrichment process and BRMS is made by a service bus.

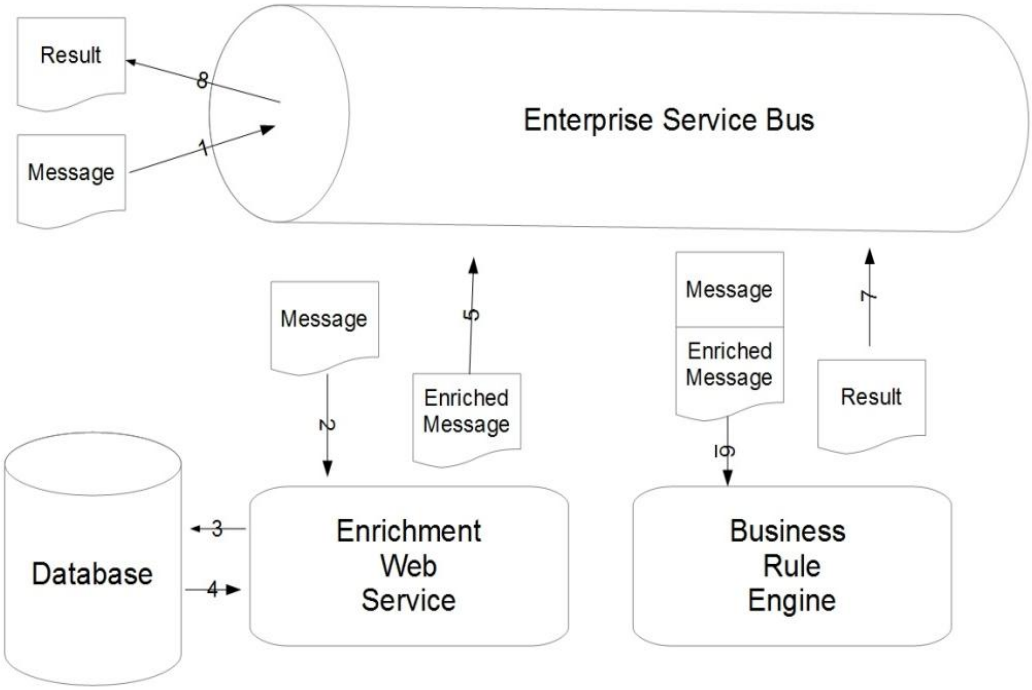

Figure 9 : An Example Architecture for the First Approach

Second approach, is completely combining Enrichment Service into BRMS. In this approach, when the message (or the data on which the rules will be executed) comes to a BRMS, parts of it are sent to enrichment processes when enrichment is needed while the execution of the rules is still in progress. All necessary enrichments are done via a service or services one by one and the result is sent to BRMS sequentially. As in the first approach, the enrichment data can be anywhere. An example architecture targeting this approach is shown in Figure 10. In this figure, enrichment needs existing data in the database and requested trough a web service. The connection between the enrichment process and the BRMS is realized through a service bus. The difference between the first approach and the second approach is the time when this operation is conducted.

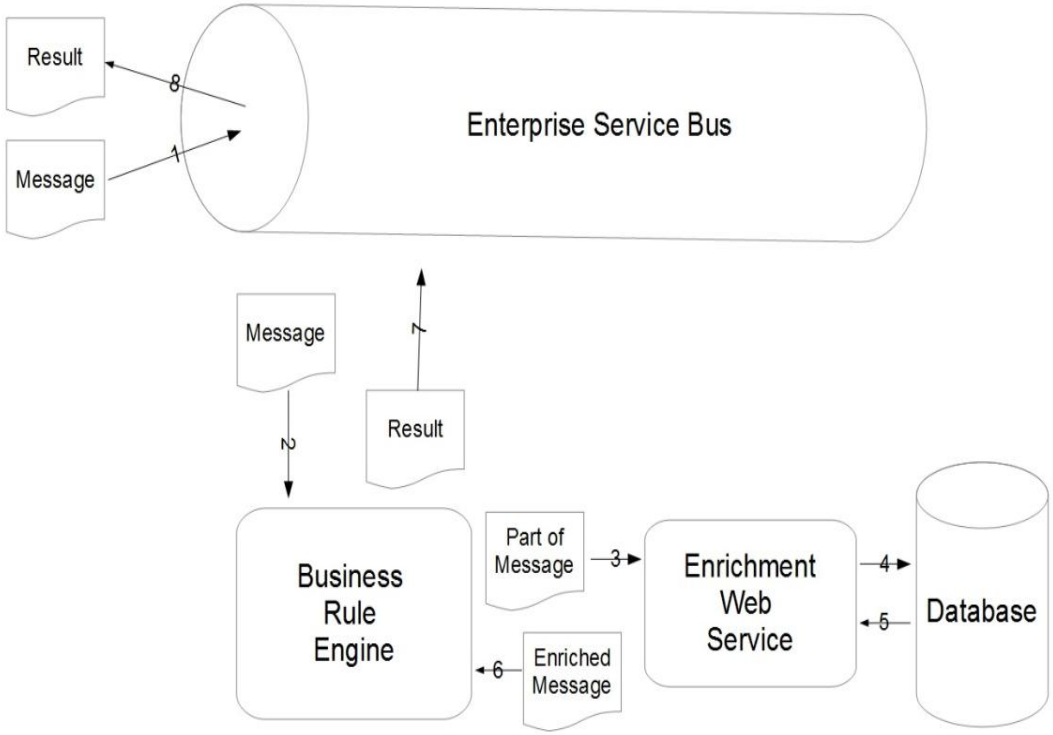

Figure 10 : An Example Architecture for the Second Approach

Third approach is a mixing of the first and the second approaches. This approach can be chosen for efficiency in some cases. For example, if some of the rules that need enrichment are executed according to some of the other rules' results and the numbers of rules that are in this situation are not so many, this approach can be efficient. In this approach, the enrichment-requiring and the static part of the message are sent to the enrichment process as in the first approach and other parts are sent to enrichment processes when they are triggered by another rule that causes the need for enrichment while the execution of the rules is still in progress as in the second approach. An example architecture targeting this approach is shown in Figure 11.

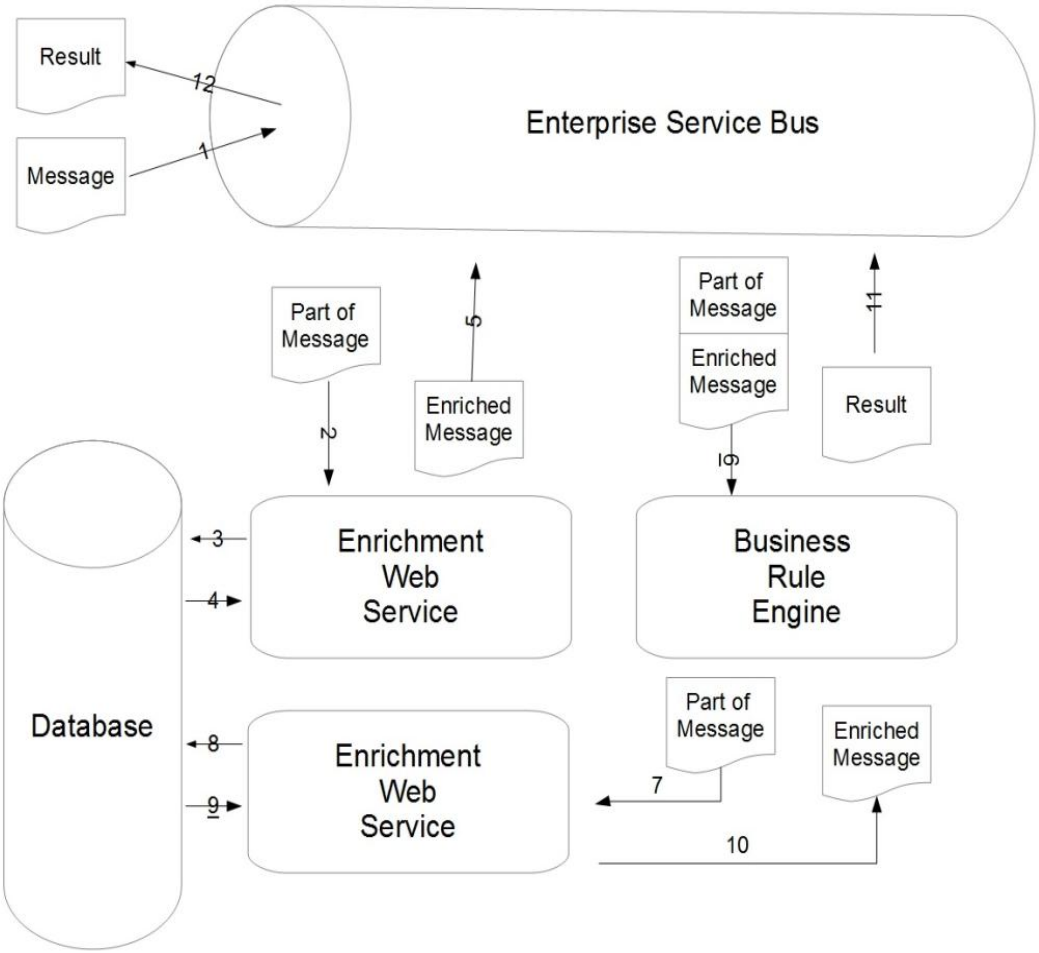

Figure 11 : An Example Architecture for the Third Approach

CHAPTER 3

PROPOSED APPROACH: CONDITION ORIENTED ENRICHMENT (COE)

In BRMS literature, the communication methods with the services for enrichment are known and explained in Chapter 2.4.2. But answer to how to do enrichment includes only one common approach. In this approach fetching unknown values for enrichment is done for all of the values and in some cases cost of this operation can cause high performance problems.

With a new approach called Condition Oriented Enrichment (COE), it is targeted to find another way / approach for enrichment to increase the efficiency of the BRMS.

In this chapter, a new approach "Condition Oriented Enrichment Approach" will be described and explained with examples. The comparison with the existing approach will be done more detailed at Chapter 3.3 and the results will be discussed in Chapter 4.2.5 and 5

3.1. CONCEPTS

Before explaining COE, explaining the concepts that COE uses will be useful. These concepts are not general concepts that BRMS uses but they are newly produced by using and extending the fundamental concepts of BRMS to understand COE more clearly. They are defined in this study.

There are three concepts:

i. Binary Rule,
ii. Simple Rule, and
iii. Complex Rule

And the definitions of these concepts are below:

i. Binary Rule: If in the result of a rule there exists an assignment operation from a set of some different two values and "uncertain" and "unknown" values, then COE calls this rule as "binary rule". These "uncertain" and "unknown" elements can be ignored in practical implementation.

Example of a Binary Rule:

if height < 50 cm then "small object"=true

In this rule, if condition part is true (height is smaller than 50) then value of "small object" will be true. If height is bigger than "50" then it is known that the rule can continue as following:

else "small object"=false

This else part can be deleted, and if it is deleted then there will be another value in "small object" which is "uncertain". If there is no value in "height" variable, then the value for "small object" will be "unknown".

In here, the set of results has two different values. If the result is not uncertain or unknown, it is known that one of two values will be chosen.

Example of a Non-Binary Rule:

if height < 50 cm then object is small

In this rule, it seems a binary rule. But because of the unknown values of the result set, it is not a binary rule. This rule can continue with the followings:

else if height < 100 cm then object is medium

else if height < 150 cm then object is long

.. and so on.

But if it is known that it ends with only one "else" as following:

else object is long

then it can be a binary rule.

ii. Simple Rule: If a rule has only one condition that cannot be divided into smaller ones except its subsets, COE calls this rule as a "simple rule".

Example:

if height < 50 cm then object is small

In this rule the condition part is "height < 50 cm" have subsets and can be written as following:

if (height < 20 cm) or (height < 50 cm) and not (height >100 cm)

then object is small

But, this rule is still a simple rule.

iii. Complex Rule: If a rule has more than one condition that cannot be divided into smaller ones except its subsets, COE calls this rule as "complex rule".

Example of a Non-Complex Rule:

if (height < 20 cm) or (height < 50 cm) and not (height >100 cm)

then object is small

This rule is a simple rule; it is not a complex rule because it can be written as following:

if height < 50 cm then object is small

Example of a Complex Rule:

if (height < 20 cm) or (weight < 50 kg) and not (width >100 cm)

then object is small

There are three different conditions in this complex rule.

3.2. PROPOSED APPROACH

Enrichment of every rule is different from each other and all of them must be enriched separately. COE offers three main steps for reorganizing the rule and developing the enrichment service.

By applying these steps for each rule in the BRMS, it is possible to develop an efficient enrichment service for the whole BRMS. Some or all of these steps can be used on a rule according to the related rule's situation. These steps are applied on the rules and related parts of the enrichment service at development time:

1. If the business rule is a binary rule and has a simple condition, then enrichment for this rule is done by providing a result that is corresponding to the rule's condition part. The rule is reorganized according to the response of enrichment service. While adding a limitation to inquiry method at enrichment service provides this correspondence, it is very important to be aware of the difference between "embedding the business into the source code of inquiry method at enrichment service" and "providing a result that is corresponding to the rule's condition part" for the sake of not losing BRMS' flexibility.
2. If the business rule is a binary rule and has a complex condition, then the rule is divided into sub-rules until all of them are simple rules. For each sub-rule, 1st step is applied and the main-rule is reorganized in order to contain the results of these sub-rules.
3. Conversion of a rule is possible if the result part of the rule contains only assignment operation(s). If the business rule is not a binary rule;
    a. If conversion of the rule is not possible, make enrichment without any modification (apply "Common Enrichment" approach.).
    b. If conversion of a rule is possible, then convert the rule into a binary rule.

"Conversion of the rule" is done by behaving as it has two elements in result set. One of them is selected as first value and the whole set of the remaining values is selected as second value. The rule is reorganized according to these two results and enrichment is done by applying 1st or 2nd methods according to its condition.

While adding limitation to inquiry method at enrichment service, "Binary Rule" concept at COE approach avoids embedding the business into the source code of inquiry method at enrichment service. As in the definition of "Binary Rule", there can be only two values in the result set of the rule. A way to add limitation to enrichment inquiry is mapping these two results to "Boolean" values (A "Boolean" value consists of two values "true" and "false"). One result can be mapped to "true" and other can be mapped to "false". If the rule is reorganized according to ask a "yes/no" question the answer will be "true" or "false". In the enrichment process, asking a question corresponding to the condition's question will provide a "Boolean" result. This Boolean result can be sent to the rule for decision and rule can decide on this boolean value. In this way, the business is not embedded into the source code

of enrichment service because if the “binary rule” wants to be changed later, then there is only one possibility: Choosing another value as condition result. And we do not need to change enrichment process, taking its complementary (if first result is "true", then choose "false" or if first result is "false", then choose "true") in result of the rule will be enough. More detailed examples will be given at Chapter 3.3.

These steps can be figured as below:

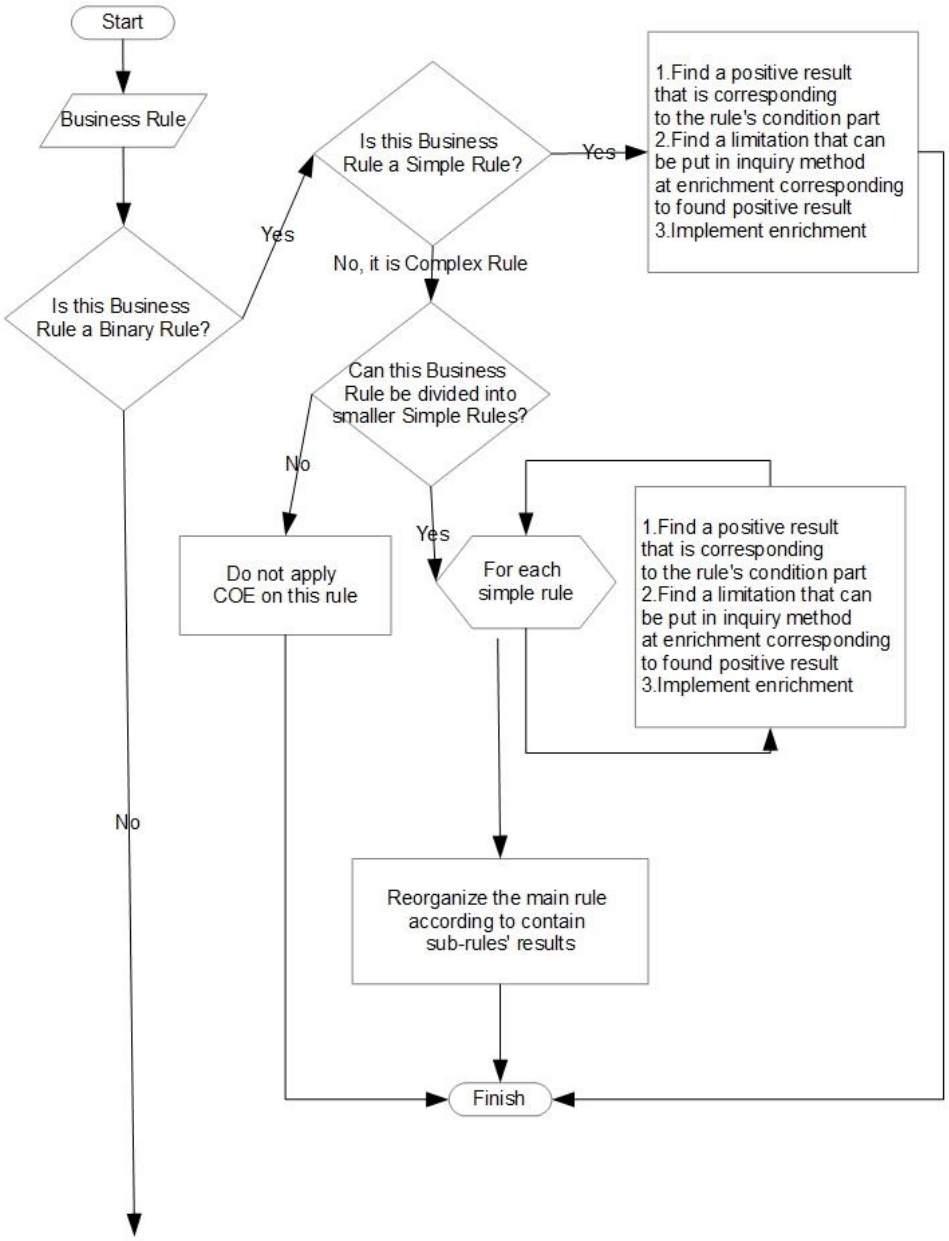

Figure 12 : COE steps at development time

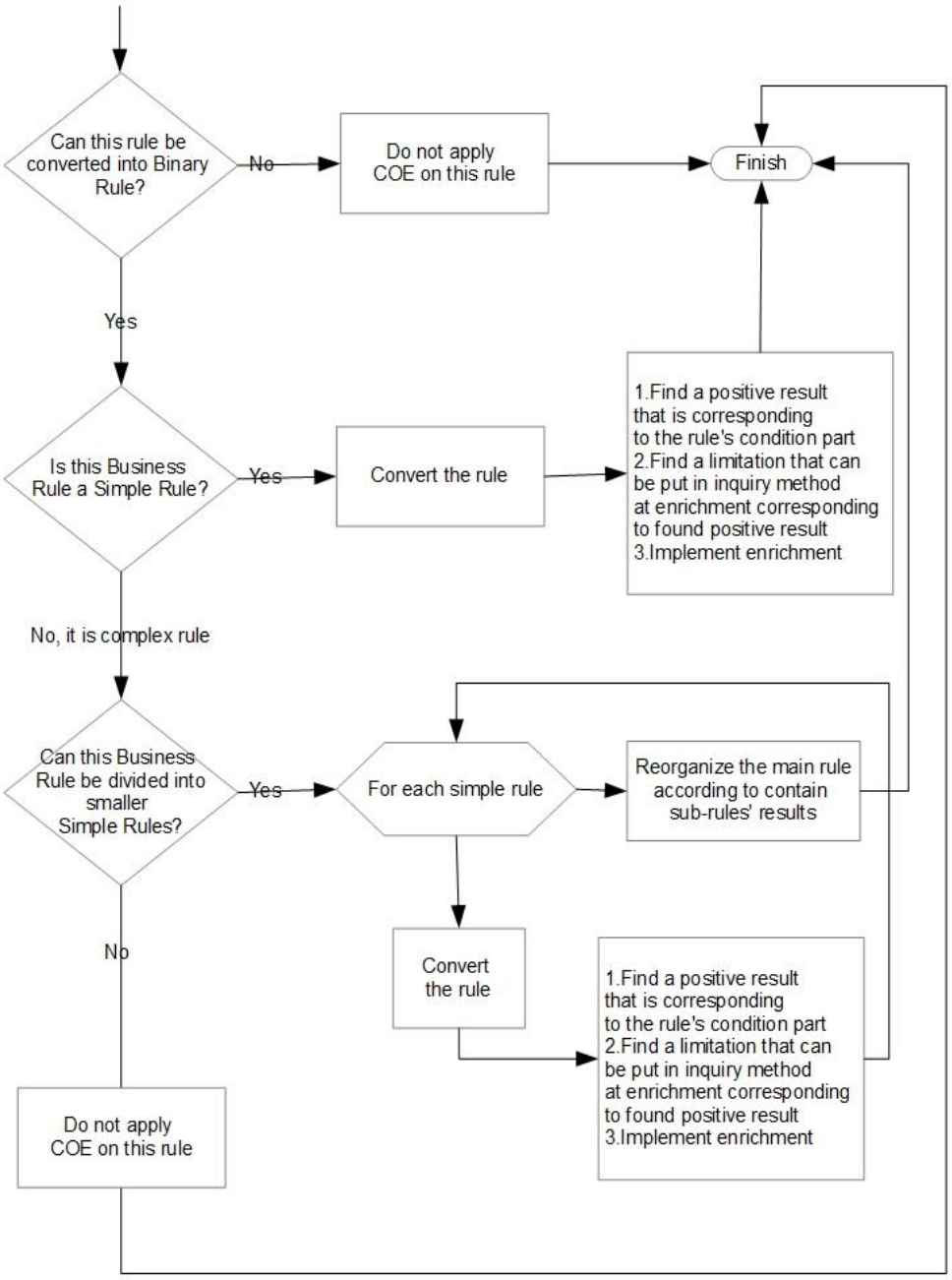

Figure 12 : COE steps at development time (continued)

All these steps are implemented to enrichment service by the developers corresponding to the rules at BRMS. More detailed steps for implementing 1st, 2nd and 3rd methods of COE on a business rule at development time are below:

1. Is this rule a binary rule?
    1.1. Yes, this rule is binary rule. Is this a simple rule?
        1.1.1. Yes, the rule is a simple rule.
            1.1.1.1. Find a positive result for the condition of the rule.
            1.1.1.2. Find a limitation that can be put in inquiry method at enrichment corresponding to found positive result in 1.1.1.1.
            1.1.1.3. Avoid embedding the business into the source code of enrichment service.
        1.1.2. No, the rule is a complex rule. Can this rule be divided into smaller simple rules?
            1.1.2.1. Yes, this rule can be divided into smaller simple rules.
                1.1.2.1.1. For each simple rule
                    1.1.2.1.1.1. Find a positive result for the condition of the rule.
                    1.1.2.1.1.2. Find a limitation that can be put in inquiry method at enrichment corresponding to found positive result in 1.1.2.1.1.1.
                    1.1.2.1.1.3. Avoid embedding the business into the source code of enrichment service
                1.1.2.1.2. Reorganize the main rule according to containing sub-rules' results
            1.1.2.2. No, this rule cannot be divided into smaller simple rules. The COE approach cannot be applied to this rule. Apply CE Approach on this rule.
    1.2. No, this rule is not a binary rule.
        1.2.1. Can this rule be converted into Binary Rule?
            1.2.1.1. No, this rule cannot be converted; COE approach cannot be applied to this rule. Apply CE Approach on this rule.
            1.2.1.2. Yes, this rule can be converted.
                1.2.1.2.1. Is this a simple rule?
                    1.2.1.2.1.1. Yes, the rule is a simple rule.
                        1.2.1.2.1.1.1. Convert the rule
                        1.2.1.2.1.1.2. Find a positive result for the condition of the rule.
                        1.2.1.2.1.1.3. Find a limitation that can be put in inquiry method at enrichment corresponding to found positive result in 1.2.1.2.1.1.2.
                        1.2.1.2.1.1.4. Avoid embedding the business into the source code of enrichment service.
                    1.2.1.2.1.2. No, the rule is a complex rule. Can this rule be divided into smaller simple rules?
                        1.2.1.2.1.2.1. Yes, this rule can be divided into smaller simple rules.
                        1.2.1.2.1.2.2. For each simple rule
                            1.2.1.2.1.2.2.1. Convert the rule
                            1.2.1.2.1.2.2.2. Find a positive result for the condition of the rule.
                            1.2.1.2.1.2.2.3. Find a limitation that can be put in inquiry method at enrichment corresponding to found positive result in 1.2.1.2.1.2.2.2.
                            1.2.1.2.1.2.2.4. Avoid embedding the business into the source code of enrichment service

1.2.1.2.1.2.3. Reorganize the main rule according to contain sub-rules' results
1.2.1.2.1.2.4. No, this rule cannot be divided into smaller simple rules. The COE approach cannot be applied to this rule. Apply CE Approach on this rule.

Before applying these steps on a rule CE algorithm is as below:

Enrichment Algorithm with CE Approach:

Input: *Conditions of the rule*
Output: *Enriched values*
for each *condition of the rule*
fetch *all values from database*
send *all values to BRMS*

These values send to BRMS from enrichment and the rule is executed in BRMS with all enriched values.

After the COE steps are applied on the rule, CE algorithm is changed to COE algorithm as below:

Enrichment Algorithm with COE Approach:

Input: *Conditions of the rule*
Output: *Enriched value*
for each *condition of the rule*
fetch *one value from database*
send *one value to BRMS*

This one value sends to BRMS from enrichment and the rule is executed in BRMS with this one enriched value.

In this algorithm because of the limitation that is added in inquiry method fetching is done for only "one value". The efficiency of COE approach can easily be observed in the following comparison: the set of enriched values of CE approach contains N enriched elements (where N>=1), but the set of enriched values of COE approach contains only 1 element.

3.3. EXAMPLES

Example for 1st Method:

Business Rule 1:

Let's consider, a BRMS accepts an input message in xml format and there is a field that contains "vaccination code" in it and business rule looks like this:

if the code of vaccination is in vaccination codes lookup table in database

then the message is valid

else

the message is invalid

Common Enrichment Approach (CE) on Business Rule 1 at Development Time:

Enrichment service must be designed to fetch all of the codes of vaccination from the vaccination lookup table in the database and it must have the ability to send them to BRMS at runtime.

Also, the rule must be reorganized according to the enrichment approach. In this example the rule must be reorganized to look up all values for a match among the enrichment values and can be logically written like this (bold written):

if the code of vaccination ***is in the enriched data list***

then message is valid

else

message is invalid

Cost of the CE Approach for the Business Rule 1 at Runtime:

Algorithm of CE at runtime:
Input: *Conditions of the rule*
Output: *Enriched values*
for each condition of the rule (1)
fetch all values from database (2)
send all values to BRMS (3)

**The output of this algorithm is used by BRMS to execute the rule.(4)*

Step 1 is ignored because there is only one condition.

At step 2, It is assumed that database uses linear search algorithm (this method is looking for a match in a collection by sequentially moving through it [9]) for fetch operation.

At step 3, the communication time is assumed to be a constant value and is ignored.

At 4, it is assumed that BRMS uses a linear search algorithm during execution.

The reason of these assumptions is to provide a clear comparing between related parts of the approaches. After these assumptions, the cost of these steps can be formulated as:

Execution Time = O(n) for 1^{st} step + O(n) for 3^{rd} step

Data Cost of Common Approach = O(n) Data Cost for Fetched Values in 1^{st} step

COE Approach on Business Rule 1 at Development Time:

According to the COE Approach, these five main steps must be applied to a rule during development:

1. Is this rule a binary rule?
   Yes, it is a binary rule because there is an assignment operation in the result part using values from a set of four elements which are "valid", "invalid", "uncertain", and "unknown".
2. Is this rule simple?
   Yes, it is simple because it has only one condition that cannot be divided into smaller ones except its subsets.
3. What is the positive result for this condition?
   If "the code of vaccination" is found in the lookup table in the database, then this condition result will be "true/positive".
4. How can a limitation be put in inquiry method at enrichment corresponding to this positive result?
   In this rule, the criterion "existency of a code in a table" is examining the condition part of the rule, so "the code of vaccination" can be inquired from database. If it is found in the lookup table then "true" value, else "false" value can be sent as result value. The reorganized rule can be written like this (bold written parts indicate modifications):

   *if **the result of the inquiry for the condition** "the code of vaccination is in the vaccination codes lookup table in database**" is "true"***

   then message is valid

   else

   message is invalid

5. Does this limitation embed the business into the source code of inquiry method at enrichment?

   No: if this is a binary rule, the result can take two values. The changed rule can replace the condition's ("if part"s result) or complementary-condition's result ("else part"s result) by only one of those values.

   If this rule is changed, then it can look like as below :

   if the result of the inquiry for the condition "the code of vaccination is in the vaccination codes lookup table in database" is "true"

   *then message is **invalid***

   else

   *message is **valid***

   Any other changes must be considered as writing a new rule and if so the approach must be implemented starting from the first step.

Cost of the COE Approach for the Business Rule 1 at Runtime:

Algorithm of COE at runtime:
***Input:** Conditions of the rule*
***Output:** Enriched value*
for each condition of the rule (1)
fetch one value from database (2)
send one value to BRMS (3)

**The output of this algorithm is used by BRMS to execute the rule. (4)*

Step 1 is ignored because there is only one condition.

At step 2, It is assumed that database uses a linear search algorithm (this method is looking for a match in a collection by sequentially moving through it [9]) for fetch operation.

At step 3, the communication time is assumed to be a constant value and is ignored.

At 4, BRMS decides on only one value at execution time.

The reason of these assumptions is to provide a clear comparison between related parts of the approaches.

After these assumptions, the cost of these steps can be formulated as:

Execution Time Cost = O(n) for 1st step + O(1) for 3rd step

Data Cost of the COE Approach = O(1) Data Cost for Inquiry Result in the 1st step.

Comparison of CE and COE Approaches for Business Rule 1:

In the above example, it can be observed that for the COE Approach is more time-efficient than the CE Approach if n>2 (where n is the number of values returned from the enrichment). The cost of data for the COE Approach is also less than that of the CE Approach if n>1 (where n is the number of values returned from the enrichment).

Although, it seems that COE is more efficient than CE at runtime; the cost for "Reorganization of a Rule" can change according to the complexity of the rule at development time. But for most cases, it can be thought that it is easier to implement the COE approach than the CE Approach because Binary Rules can have only two values in their results.

It is important to emphasize that there is no need to choose COE if the number of values returned from the enrichment is less than 2.

Example for the 2nd Method:

Business Rule 2:

A BRMS that accepts xml messages as input is considered for this example.

The below rule is desired to be executed to decide if the message is valid**:**

If patient's last menstrual period time in previous pregnancy declaration messages < patient's menstrual period time in pregnancy declaration at incoming message and patient's menstrual period time in the pregnancy declaration incoming message > patient's birth time

then message is valid

else

message is invalid

Common Enrichment Approach (CE) on Business Rule 2 at Development Time:

Enrichment service must be designed to fetch all of the "*patient's last menstrual period time"* values in patient's previous pregnancy declaration messages which are at the patient's pregnancy declaration messages' table in the database for the first condition part and "patient's birth time" value must be also fetched from the patient's information table in the database for the second condition part. The enrichment service must also have the ability to send them to the BRMS at runtime.

The rule must be reorganized according to the enrichment approach. In this example rule must be reorganized to look up all values for a match among the enrichment values. It can be logically written as follows:

> *If patient's last menstrual period time in previous pregnancy declaration messages* ***in enriched list*** *< patient's menstrual period time in pregnancy declaration at incoming message and patient's menstrual period time in the pregnancy declaration incoming message > patient's* ***enriched*** *birth time*
>
> *then message is valid*
>
> *else*
>
> *message is invalid*

Cost of the CE Approach for Business Rule 2 at Runtime:

> *Algorithm of CE at runtime:*
> ***Input:*** *Conditions of the rule*
> ***Output:*** *Enriched values*
> *for each condition of the rule (1)*
> *fetch all values from database (2)*
> *send all values to BRMS (3)*
>
> **The output of this algorithm is used by BRMS to execute the rule.(4)*

At step 2, It is assumed that database uses a linear search algorithm (this method is looking for a match in a collection by sequentially moving through it [9]) for fetch operation.

At step 3, the communication time is assumed as a constant value and ignored.

At 4, it is assumed that BRMS uses linear search algorithm for searching values during execution.

The reason for these assumptions is to provide a clear comparison between the related parts of the approaches.

After these assumptions, the cost of these steps can be formulated as:

Execution Time = k times (O(n) for 1^{st} step + O(n) for 3^{rd} step)

Data Cost for the Common Approach = k times (O(n) for Fetched Values in 1^{st} step)

(Where k is the number of conditions.)

COE Approach on Business Rule 2 at Development Time:

According to the COE Approach, these steps must be applied to a rule:

1. Is this rule a binary rule?
   Yes, it is a binary rule because in the result part there exists an assignment operation from values of a set of two elements (which are "valid" and "invalid") plus "uncertain" and "unknown" elements
2. Is this rule simple?
   No, it is not simple because it has two conditions, which are not subsets of each other.
3. Can this rule be divided into smaller simple rules?
   3.1. Yes, it can be divided into two simple rules
   3.2. For the first rule
      3.2.1. What is the positive result for this condition?
      If all of the "patient's last menstrual period time" values in patient's pregnancy declaration messages' table in the database are smaller than the value in "patient's menstrual period time in pregnancy declaration at incoming message", then this condition result will be "true/positive".
      3.2.2. How can a limitation be put in inquiry method at enrichment corresponding to this positive result?
      In this rule, the criterion "comparison of a date value" is examining the condition part of the rule, so "*patient's last menstrual period time"* values that are bigger than the value "*patient's last menstrual period time"* in message can be searched from the database. If it is found in that lookup table then "true" value, else "false" value can be sent as inquiry result. The reorganized rule can be written like this:

      *If **the result of the inquiry for the condition** "patient's last menstrual period time in previous pregnancy declaration messages < patient's menstrual period time in pregnancy declaration at incoming message" **is "true"***

      then result1 is valid

      else

      then result1is invalid

      3.2.3. Does this limitation embed the business into the source code of inquiry method at enrichment service?

      No: this is a binary rule, therefore the result can take only two values. The change of the rule can only be the replacement of these two values in condition result ("if part"s result) or in complementary-condition result ("else part"s result).

      If this rule is changed, then it can look like this:

      If the result of the inquiry for the condition "patient's last menstrual period time in previous pregnancy declaration messages < patient's menstrual period time in pregnancy declaration at incoming message" is "true"
      *then result1 is **valid***
      else
      *then result1 is **invalid***
   3.3. For the second rule
      3.3.1. What is the positive result for this condition?
      If "patient's birth time" value (which is unique for every patient) in patient's information table in the database is smaller than the value in "patient's menstrual period time in pregnancy declaration at incoming message", then this condition result will be "true/positive".

3.3.2. How can a limitation be put in inquiry method at enrichment corresponding to this positive result?

In this rule, the criterion "comparison of a date value" is examining the condition part of the rule, so "*patient's birth time"* value that is bigger than the value "*patient's last menstrual period time"* in message can be searched from the database. If it is found in that lookup table then "true" value, else "false" value can be sent as inquiry result. The reorganized rule can be written like this (bold written):

*If **the result of the inquiry for the condition** "patient's menstrual period time in the pregnancy declaration at incoming message > patient's birth time" **is "true"***

then result2 is valid

else

result2 is invalid

3.3.3. Does this limitation embed the business into the source code of inquiry method at enrichment?

No: because this is a binary rule, there can be only two values that result can have. The rule can only be changed through the replacement of these two values in the condition result ("if part"s result) or in the complementary-condition result ("else part"s result).

If this rule is changed, then it can look like this:

If the result of the inquiry for the condition "patient's menstrual period time in the pregnancy declaration at incoming message > patient's birth time" is "true"

*then result2 is **invalid***

else

*result2 is **valid***

3.4. Reorganize the main rule to contain these two rules:

This reorganized rule can be written as below:

If the result of the inquiry for the condition "patient's last menstrual period time in previous pregnancy declaration messages < patient's menstrual period time in pregnancy declaration at incoming message" is "true"

then result1 is valid

else

then result1is invalid

If the result of the inquiry for the condition "patient's menstrual period time in the pregnancy declaration at incoming message > patient's birth time" is "true"

then result2 is valid

else

result2 is invalid

if result1 is valid and result2 is valid

then message is valid

else

message is invalid

Cost of the COE Approach for Business Rule 2 at Runtime:

Algorithm of COE at runtime:
Input: *Conditions of the rule*
Output: *Enriched value*
for each condition of the rule (1)
fetch one value from database (2)
send one value to BRMS (3)

**The output of this algorithm is used by BRMS to execute the rule. (4)*

At step 2, It is assumed that database uses linear search algorithm (this method is looking for a match in a collection with sequentially moving through it [9]) for fetch operation.

At step 3, the communication time is assumed as a constant value and ignored.

At 4, BRMS decides on only one value in execution time.

The reason of these assumptions is to provide a clear comparing between related parts of the approaches.

After these assumptions and explanations, the cost of these steps can be formulated as:

Execution Time Cost = k times (O(n) for 1st step + O(1) for 3rd step)

Data Cost of COE Approach = k times (O(1) for Inquiry Result in 1st step)

(Where k is the number of the conditions.)

Comparison of CE and COE for the Business Rule 2:

In the above example, the cost formulas are same with the 1st example results with a prefix cost of "k" where k is the number of conditions. It again shows that COE is more efficient than CE.

Example for the 3rd Method:

Business Rule 3: (a Non-Binary Rule)

If diagnostic code of examination of the patient exists in diphtheria codes

then patient is diphtheria

else

If diagnostic code of examination of the patient exists in typhus codes

then patient is typhus

else

If diagnostic code of examination of the patient exists in hydrophobia codes

then patient is hydrophobia

else

If diagnostic code of examination of the patient exists in measles codes

then patient is measles

else

patient is healthy

In the above example, there is a business rule chain in which the rules are connected to each other with "else" keyword. Because the patient's status can have five different values, neither any one of the rules nor the combined rule are binary.

Considering it is known that there is only patient information at coming message, "*diphtheria codes*", *"typhus codes", "hydrophobia codes"* and *"measles codes"* values must be enriched.

The first rule is chosen for explanation in below approaches for its simplicity.

Common Enrichment Approach (CE) on Business Rule 3 at Development Time:

Enrichment service must be designed to fetch all of the "*diphtheria codes*" values from the related table in the database. Enrichment service must also have the ability to send them to the BRMS at runtime.

Also, the rule must be reorganized according to the enrichment approach. In this example, the rule must be reorganized to look up all the values for a match among the enrichment values. It can be logically written as follows:

If diagnostic code of examination of the patient exists in ***the enriched diphtheria codes list***

then patient is diphtheria

Cost of the CE Approach for Business Rule 3 at Runtime:

Algorithm of CE at runtime for 1st Rule ("diagnostic code of examination of the patient exists in enriched diphtheria codes list"):
Input: *Conditions of the rule*
Output: *Enriched values*
for each condition of the rule (1)
fetch all values from database (2)
send all values to BRMS (3)

**The output of this algorithm is used by BRMS to execute the rule.(4)*

Step 1 is ignored because there is only one condition.

At step 2, It is assumed that database uses linear search algorithm (this method is looking for a match in a collection with sequentially moving through it [9]) for fetch operation.

At step 3, the communication time is assumed as a constant value and ignored.

At 4, it is assumed that BRMS uses linear search algorithm for searching values in execution.

The reason of these assumptions is to provide a clear comparing between related parts of the approaches.

After these assumptions, the cost of these steps can be formulated as:

Execution Time = O(n) for 1st step + O(n) for 3rd step

Data Cost of Common Approach = O(n) for Fetched Values in 1st step

<u>COE Approach for Business Rule 3 at Development Time:</u>

According to the COE Approach, these steps must be applied to the rule:

1. Is this a binary rule?
   No, it is not a binary rule because an assignment operation in the result part uses a value from a set of five elements (which are "*diphtheria, typhus, hydrophobia, measles and healthy*") plus "uncertain" and "unknown" elements
2. Can this rule be converted?
   2.1. Yes, it can be converted into "Binary Rule". (Because there is an assignment operation in the result of the condition. If there are other operations except assignment, then it cannot be converted into "Binary Rule".)
   2.1.1.Is this a simple rule?
   2.1.1.1. Yes, it is a simple rule.
   2.1.1.1.1. How can it be converted into "Binary Rule"

   If the value "*diphtheria*" is selected and the rest of the values "*typhus, hydrophobia, measles and healthy*" are thought and called as "anti-*diphtheria"* this rule can be written as follows:

   If diagnostic code of examination of the patient exists in diphtheria codes

   then patient is diphtheria

   else

   patient is anti-diphtheria

   2.1.1.1.2. What is the positive result for this condition?

   If "*diagnostic code of examination of patient"* value is in "*diphtheria codes" lookup* table in database, then this condition result will be "true/positive".

   2.1.1.1.3. How can a limitation be put in inquiry method at enrichment corresponding to this positive result?

   In this rule, the criterion "existency of a value in a table" is examining the condition part of the rule, so "*diagnostic code of examination of patient*" can

be searched from the database. If it is found in that lookup table then "true" value, else "false" value can be sent as the result. The reorganized rule can be written like this:

*if **the result of the inquiry for the condition** "diagnostic code of examination of patient in diphtheria lookup table" **is "true"***

then patient is diphtheria

else a

then patient is anti-diphtheria

2.1.1.1.4. Does this embed the business into the source code of enrichment service?

No: because this is a binary rule, there can be two values that result can have. The rule can only be changed through the replacement of these two values in the condition result ("if part"s result) or in the complementary-condition result ("else part"s result).

If this rule is changed, then it can look like this :

if the result of the inquiry for the condition "diagnostic code of examination of patient in diphtheria lookup table" is "true"

*then patient is **anti-diphtheria***

else

*then patient is **diphtheria***

Cost of COE Approach for Business Rule 3 at Runtime:

Algorithm of COE at runtime for 1st Rule ("diagnostic code of examination of the patient exists in enriched diphtheria codes list"):
***Input:** Conditions of the rule*
***Output:** Enriched value*
for each condition of the rule (1)
fetch one value from database (2)
send one value to BRMS (3)

**The output of this algorithm is used by BRMS to execute the rule. (4)*

Step 1 is ignored because there is only one condition.

At step 2, It is assumed that database uses linear search algorithm (this method is looking for a match in a collection with sequentially moving through it [9]) for fetch operation.

At step 3, the communication time is assumed as a constant value and ignored.

At 4, BRMS decides on only one value in execution time.

The reason of these assumptions is to provide a clear comparing between related parts of the approaches.

After these assumptions and explanations, the cost of these steps can be formulated as:

Execution Time Cost = O(n) for 1st step + O(1) for 3rd step

Data Cost of COE Approach = O(1) for Inquiry Result in 1st step

Comparison of CE and COE for Business Rule 3:

Because the rule in this example can be converted to a binary rule, the conditions are the same as in the 1st example. COE for this rule is more efficient than CE.

CHAPTER 4

IMPLEMENTATION

In this chapter, a BRMS with DSL implementation which used "Turkish National Health Data Dictionary" as its domain will be explained and efficiency of "Condition Oriented Enrichment Approach" at enrichment part will be discussed.

4.1. DOMAIN: "TURKISH NATIONAL HEALTH DATA DICTIONARY"

As the Turkish Ministry of Health wanted to collect data that are produced at health agencies all over the country simultaneously and save them for raising the health services' quality and performance and also salary calculations for health workers, they developed projects in Turkish e-Health System [24]. Their public "National Health Data Dictionary"[25] is used to build an example BRMS system. The first version of this dictionary (1.1) was published in 2007. Now current version is 2.0 and is published on 14th March 2012. In this thesis, version 2.0 is used.

According to this dictionary, there are two types:

i. Data Elements: simple and atomic data that can only assume primitive types like integer, string, date...etc.
   For example, "Name" is a data element and its type is string and the maximum length is 50. Another example is "Patient's National Security Number". Its type is long and maximum length is 11. There are more than 450 data elements in this dictionary.
ii. Minimum Health Data Sets (MHDS), consists of data elements. Their existency knowledge (required, not required, conditionally required) and repetition knowledge (only one, more than one) is written in the dictionary.
   For example, Prescription MHDS has 14 data elements:
   Their existency situations are;
      9 of them is required,
      1 of them is required according to the value of another data element.(Conditional)
      4 of them is not required but there is no limitations to put them in Prescription MHDS.
   Their plurality situations are;
      6 of them can be only one time in one Prescription MHDS
      8 of them can be more than once in one Prescription MHDS
   There exists 65 MHDS.

The sets of MHDS are called Packages. There are 7 Packages. All of the packages have their own validation rules which are published publicly by the Ministry [26].There are minimum 5, and a maximum of 50 business rules for each MHDS.

A subset of this dictionary is chosen and a small BRMS system (called "Turkish National Health Data Validation System" in this thesis) on this subset is implemented inspired by the rules in [26]. The rules are written with "Turkish Health Data Validation Specific Language" which is a health specific language created during this thesis work. This BRMS can be grown for covering all MHDSs later and this DSL can be used in this new BRMS with a little change and adding new MHDSs to the DSL.

This subset contains One Package (Simple) and these MHDSs are:

- General Sent Package MHDS
- Pregnancy Declaration MHDS
- Vaccination MHDS

This implementation has been easy thanks to Oracle Policy Automation Tool's user-friendly interface and Turkish language support.

4.2. IMPLEMENTATION: "BUSINESS RULE MANAGEMENT SYSTEM WITH DOMAIN SPECIFIC LANGUAGE"

4.2.1. Method and Architecture

In this study, the first method introduced in Section 2.4.2 is chosen and implemented. One of these methods in Section 2.4.2 can be chosen to explain COE, because COE is independent of the relation between enrichment and BRMS. It suggests an improvement in efficiency to both Enrichment and BRMS together.

4.2.2. Tools and Technologies

BRMS Engine Selection: Oracle Policy Automation

There are many BRMS vendors that support DSL. Some of them are; IBM's ILOG, Oracle's Oracle Policy Automation (OPA), DTRules (Open Source) and JBoss's Drools (Open Source).

Using one of those tools would have been sufficient because of the fact that this thesis required the use of a DSL with a BRMS,. Among those tools, OPA (which is "a suite of software products for modeling and deploying business rules within enterprise applications" [18]) was more suitable for this research, because of ease of usage, supporting of the Turkish Language, and effective performance test results as described in [10].

Enterprise Service Bus Selection: Oracle Service Bus

Because of choosing web service method for enrichment, there was a need to connect the response of enrichment service to BRMS service. For solving this problem (orchestrating services), "Enterprise Service Bus" approach is a preferred and because of know-how of the

author on “Oracle Service Bus,” Oracle’s product “Oracle Service Bus” was used in the development.

Server and Database Selection:

Weblogic Server is used because of Oracle Service Bus and Oracle Policy Automation products running on it. Oracle Express Database is used as the database management system.

Java Technologies Used in This Project:

For development, these are used:

Java EE (Enterprise Java Platform), Jax-WS (Java API for XML Web Services), EJB 3.0 (Java EE Specification), EclipseLink (Object-Relational mapping Framework)

4.2.3. Development

The usage of OPA tool was easy. To develop a BRMS with OPA, first a project is created. “Turkish Language” was chosen for rule development. Then “Domain Models”, which were used in writing rules, were created one by one. If a domain model had attributes with a type of boolean, text, number, or date, all of them were created for each model. A number attribute is shown in Figure 12.

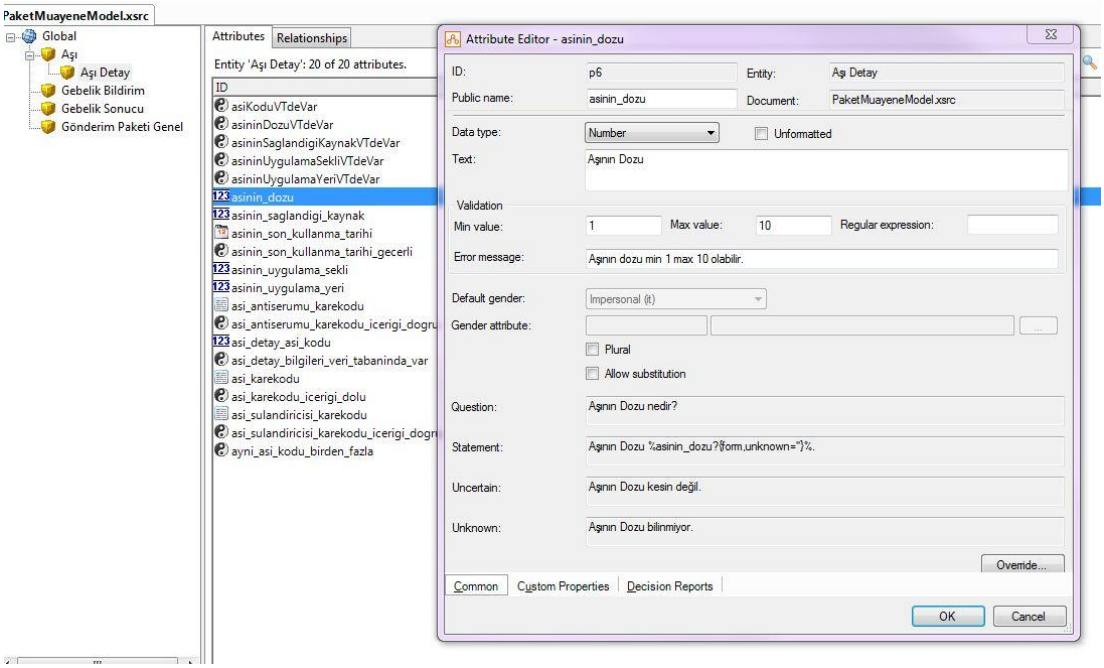

Figure 13 : A Number Attribute

In Figure 13, it is also shown that there are some facilities that OPA provides like automatic validation. If it is entered, then there is no need to code them again in BRMS, OPA provides the automatically generated rules related with it. In Figure 14, a boolean attribute can be seen:

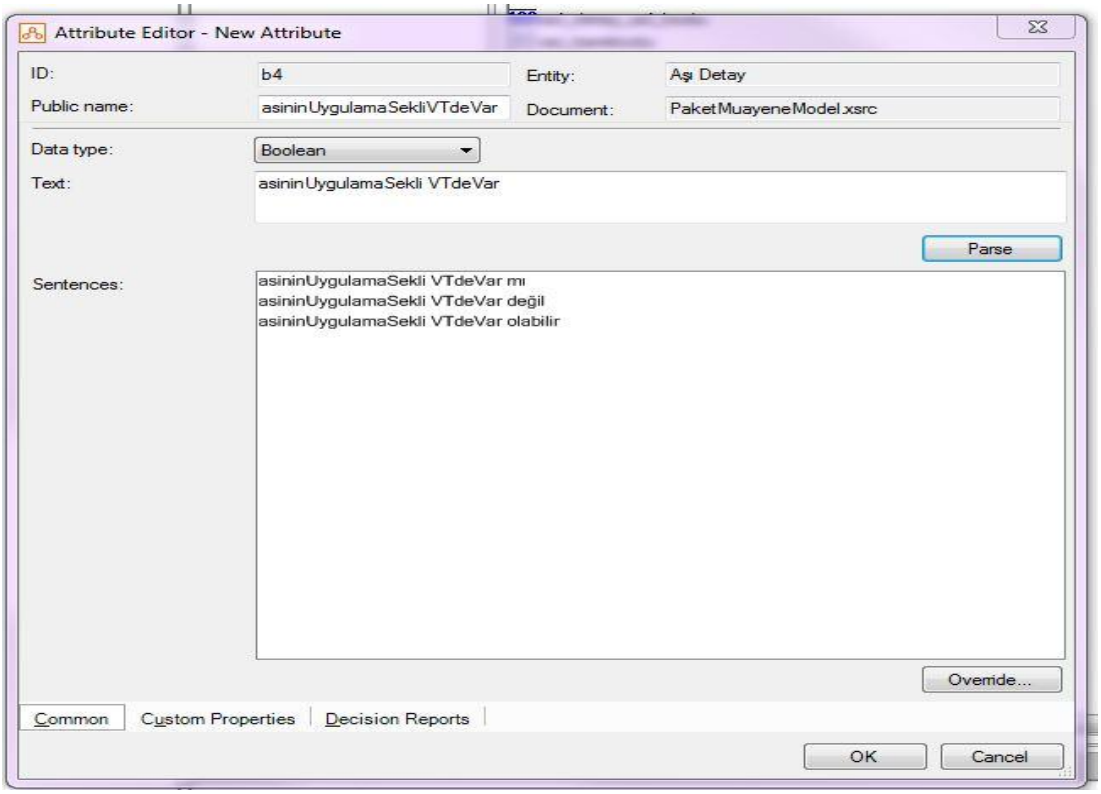

Figure 14 : A Boolean Attribute

This is also one of the other facilities of OPA. When a boolean attribute is written in OPA, it parses the boolean attribute according to the chosen language and makes sentences that are used in rules automatically. It is very helpful for creating a DSL.

For writing rules, OPA uses MS Word or MS Excel. The developers of OPA had built plugins for each of them. By the help of these plugins, rules are written in word documents and compiled in it. Some rules written in Microsoft Word are shown in Figure 15:

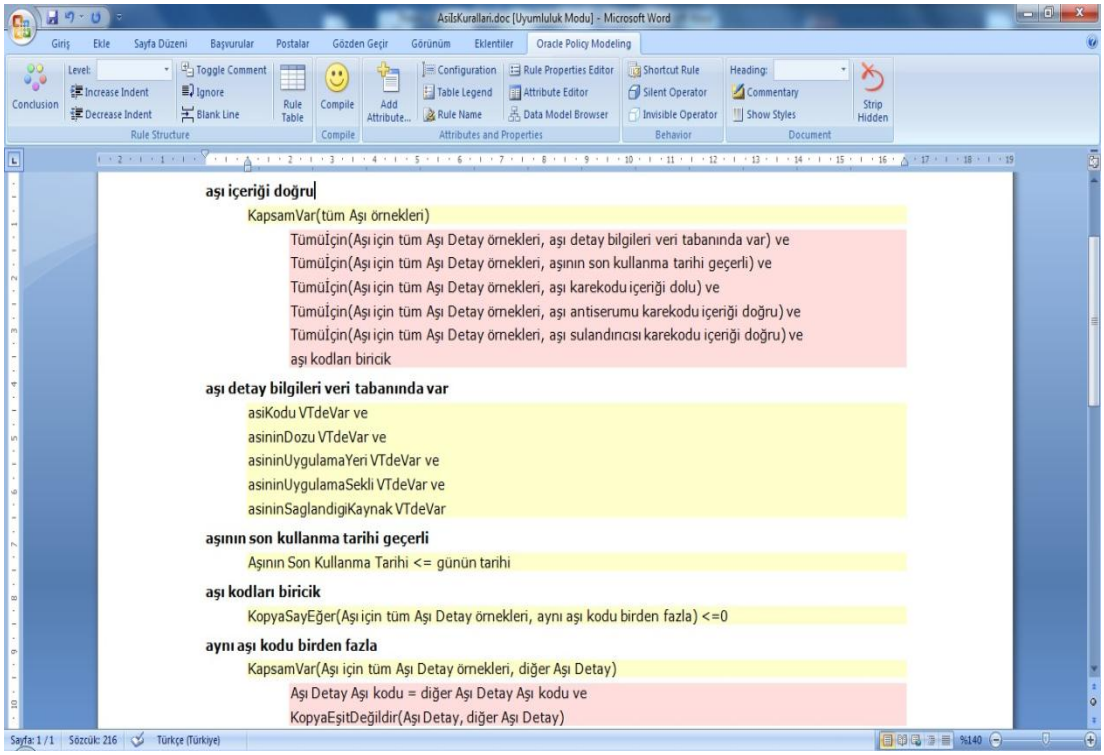

Figure 15 : Some rules written in Microsoft Word

In this example, it can be seen that some of the rules were written with COE approach. For example the boolean attribute "asiKodu VTdeVar" indicates if a code exists in the database or not. Instead of fetching all of the codes and search over them for looking if the value is valid in BRMS, one boolean attribute is enough in the COE approach. "Example 1" in Chapter 3.3 is explaining this in more detail.

After defining all necessary models with their attributes and writing all necessary rules, the whole project looks like as depicted in Figure 16.

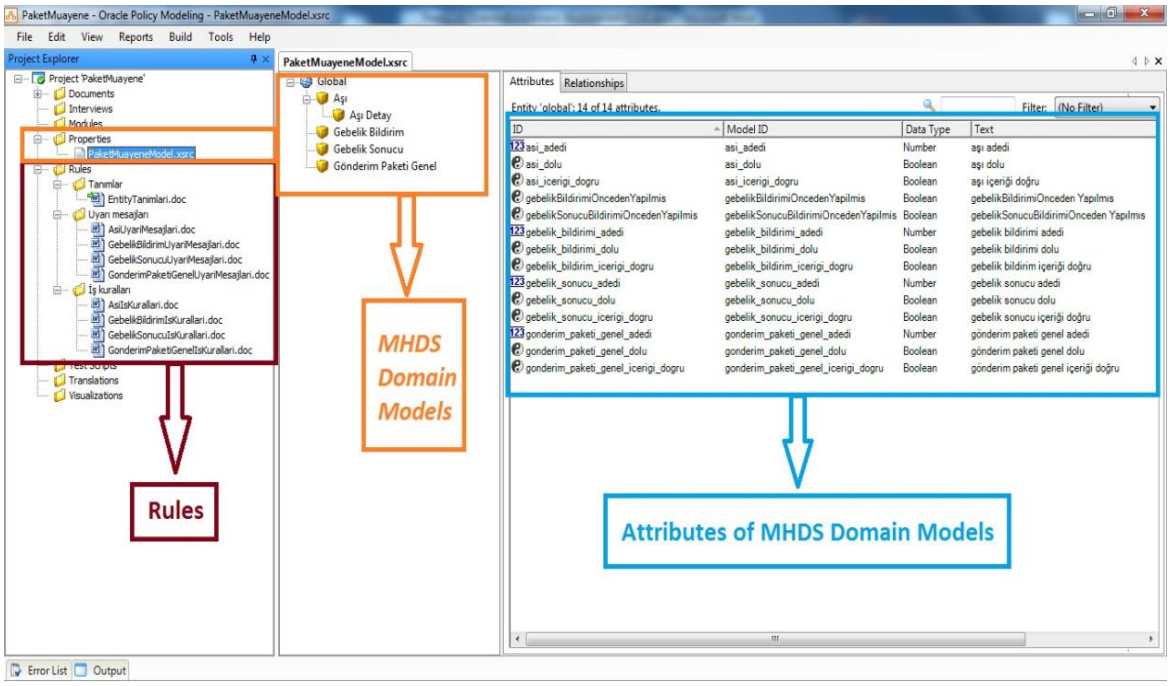

Figure 16 : A general view of a project in OPA tool of a project on OPA tool

4.2.4. Running

To run the written rules, compilation of rules is necessary first. After compilation, if there is no error, then OPA builds the executable rules. It has a simple server for debugging. While in the progress of designing of the rules, users can debug their codes and see their correctness. When the project is selected to "run", OPA creates a war file (that contains the implementation of BRMS as a web service) and runs it in its server. This war file can be deployed to any Java EE application server (some of them can need some configurations). In this development, it is deployed to Weblogic Server as a web service.

4.2.5. Testing and Results

In this chapter, COE approach and CE approach are compared according to their performance. In this work, instead of testing this system as a black box over the service bus, each service (enrichment and BRMS) is measured separately to fit the given architecture and to discuss the results with the given formulas in Chapter 3.2.

To provide more clear result, a new BRMS which has only one rule is created for benchmark. In the scenario of this benchmark, this one rule needs enrichment. Messages were sent for enrichment and their responses were sent back to the BRMS over web services. And then, average response times for each web services were measured separately. The count of enrichment values (lookup values) in the database was chosen 25, 100 and 250

respectively. The test messages were sent to each web service with 5 threads and no delay for 60 seconds.

Test platform was a Lenova PC with a Core i5 2.27GHz CPU and 8GB of RAM. Other used tools and technologies are same with the ones in Chapter 4.2.2. The results of CE Approach are in Table 4 below:

Table 5 : Performance Test Results of CE Approach

Count of Rows in Lookup table in Database	**Enrichment Service**		**BRMS**	
	Message Count (with 5 Threads and no-delay for 60seconds)	**Average Response Time for Enrichment (Selects All Enrichment Values From Database) (in ms)**	Message Count (with 5 Threads and no-delay for 60seconds)	**Average Response Time of BRMS (in ms)**
25	6094	**48.34**	3661	**81.05**
100	3205	**92.72**	1282	**233.13**
250	1517	**196.83**	550	**543.47**

In Table 4, it is seen that when the number of data fetched by enrichment service and executed by BRMS is increasing the response time of these two services are increasing relatively. This is in agreement with the algorithm costs of CE as described in Chapter 3.2.

The results of COE Approach are in Table 5:

Table 6 : Performance Test Results of the COE Approach

Count of Rows in Lookup table in Database	Enrichment Service		BRMS	
	Message Count (with 5 Threads and no-delay for 60seconds)	**Average Response Time of Enrichment (Selects All Enrichment Values From Database) (in ms)**	Message Count (with 5 Threads and no-delay for 60seconds)	**Average Response Time of BRMS (in ms)**
25	10253	**28.55**	8319	**35.17**
100	10072	28.90	8370	34.92
250	9845	29.76	8352	35.09

In Table 5, it is again seen that when the number of data is increasing, fetch time of the enrichment service is also increasing. However, because the number of fetched data is 1 (one) according to the COE approach, the selection time results are smaller than the ones in Table 4 , as expected in the CE algorithm costs explained in Chapter 3.2. The reason for slowly increasing response times is related to the response times of inquiries in enrichment.

The other difference is at BRMS's response times, in this table BRMS's response time changes can be ignored because only one input is taken through enrichment. The difference is probably because of other processes running on CPU. The unordered response time supports this probability.

In Figure 17, the comparison between COE and CE over average response times of enrichment services is figured.

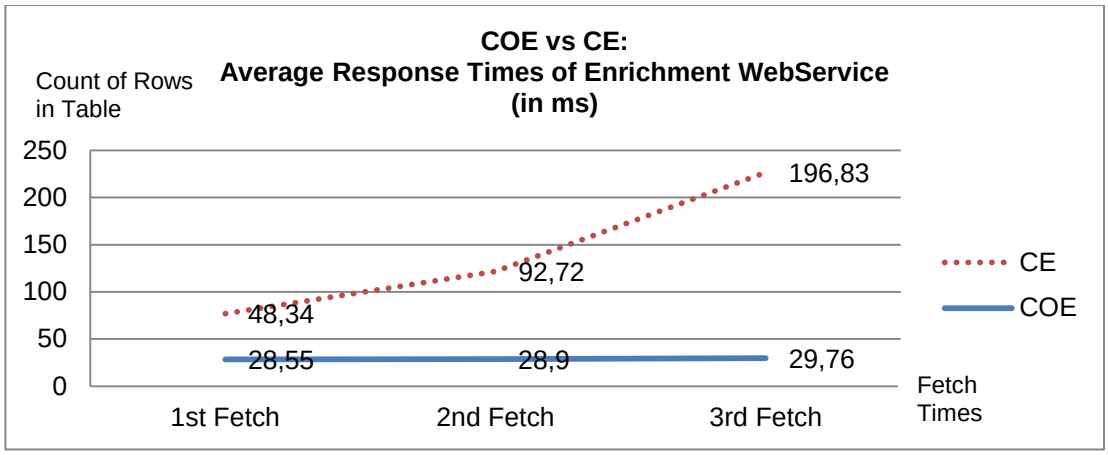

Figure 17 : Comparison of COE with CE; Average Response Times of the Enrichment Web Service

In Figure 18, a comparison between COE and CE over average response times of BRMS services is depicted.

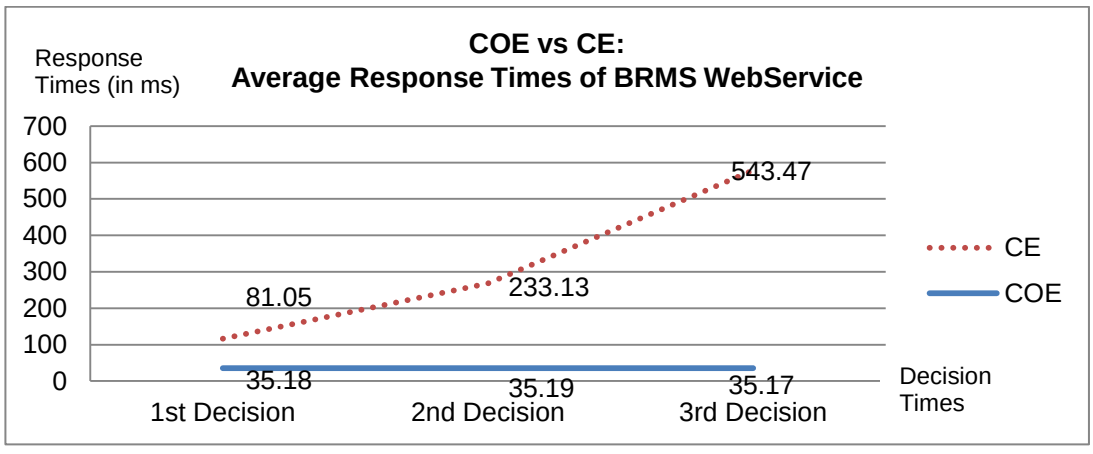

Figure 18 : Comparison of COE with CE; Average Response Times of the BRMS Web Service

CHAPTER 5

RESULTS and CONCLUSION

In this thesis, general information about BRMS and DSL were given and the old and new ways of implementing BRMS and Vendors of BRMS were mentioned. Because of the similarities and ease of usage, DSM/DSL with BRMS was used together in the same implementation.

If the data is not enough for decision making in a business rule, the need for enrichment of data in BRMS was emphasized and current approach for enrichment was explained. A new approach called "condition oriented enrichment (COE)" and its methods were explained and its efficiency is discussed at the end. It was pointed that, if the rule is a binary rule (explained in the COE approach section) then COE is meaningful and results of the benchmark indicate efficiency.

A subset of the domain "National Health Data Dictionary" [25] was chosen and a small BRMS system (called "Turkish National Health Data Validation System" inspired by the rules in [26]) on this subset was implemented to see if the development with the COE approach is possible and efficient for a real domain.

During the development of this subset, rules were written with a domain specific language (with the help of BRMS tool) named "Turkish Health Data Validation Specific Language" which is a health specific language on health domain objects.

It was observed that the development of the COE approach is possible after a small orientation time, and developing with COE Approach is easier than developing with the CE Approach. This result is qualitative and has been demonstrated during the work, supporting the opinion of the author.

The quantitative results of the benchmark were given in Chapter 4.2.5 where the efficiency of COE can be clearly observed. Enrichment was carried out by a web service in this research, however, this fact does not affect the efficiency results because the performance of the enrichment is related to the number of enriched data according to the CE, so is the performance of the BRMS. Implementing Enrichment and BRMS together was an architectural decision due to the motivation to support one of the methods described in Chapter 2.4.2. As a result, because of minimizing the enrichment data count, COE is a more efficient way to implement the enrichment process and it increases the BRMS's performance. If the number of the rules (that needs enrichment) implemented with COE increases then the whole performance of BRMS will be increased relatively.

On the other hand, using COE approach will not be an efficient solution if a non-binary rule that needs all values in its result set, is converted into a binary rule. Because, in this situation, according to COE, enrichment will be done for every result after conversion. If we take "n" as the number of the values in the result set, it can be clearly calculated that making

enrichment with COE approach “n” times will have a poor efficiency. Moreover, the communication between enrichment service and BRMS will be an additional cost to the overall performance. The CE Approach will be a better solution for this situation.

While implementing the architecture of the BRMS, it must be remembered that COE approach depends on the rule’s fact value which can only be known at runtime. However in most cases CE approach is also dependent on runtime values.

Communication speed between the enrichment service and the BRMS is another important point. If the communication latency time between the enrichment and the BRMS services is higher than the decision making time, the COE approach cannot be an efficient solution. (But in most cases, enrichment service and BRMS service have very fast communication channels between each other.)

Also, if there is a need for enrichment and the enriched data is static (will never change in runtime), using the CE Approach and caching the enriched data can be a better solution.

As a result, although in some cases using the CE approach can be a better solution; these cases are not seen so often. In most cases using the COE approach will provide high performance results. This approach and DSL were also used in a big enterprise utilizing SOA architectures and efficiency of the approach was also approved. Due to the classification of the related information, the name of the project could not be published.

REFERENCES

[1] Altıntas, N.I., Cetin, S., Dogru, A.H., Oguztuzun, H. (2012). Modeling Product Line Software Assets Using Domain-Specific Kits. IEEE Transactions on Software Engineering, vol.38, no.6, pp.1376-1402. DOI: 10.1109/TSE.2011.109

[2] Bajec, M., Krisper, M., Rupnik, R. (2000). Using business rules technologies to bridge the gap between business and business applications. COBISS.SI-ID:1955924

[3] Bourbakis, N. G. (1993). Knowledge Engineering Shells: Systems and Techniques. World Scientific, chap. 5, pp. 123-127. ISBN-13: 978-9810210564

[4] Deursen, A. V., Klint, P., Visser, J. (2000). Domain-specific languages: An annotated bibliography. ACM SIGPLAN Notices, vol. 35, no. 6, pp. 26-36. DOI: 10.1145/352029.352035

[5] Doorenbos, R. B. (1995). Production Matching for Large Learning Systems. Ph.D. Dissertation, Carnegie Mellon University.

[6] Forgy, C. L. (1982). Rete: A fast algorithm for the many pattern/many object pattern match problem. Artificial Intelligence, vol. 19, no. 1, pp. 17− 37. BIBTEX: Forgy82aij

[7] Fowler, M. (2010). Domain Specific Languages. Addison Wesley Professional. ISBN-13: 978-0-321-71294-3

[8] Kelly, S., Tolvanen, J. P. (2008). Domain-Specific Modeling Enabling Full Code Generation. A Wiley-Interscience Publication, JOHN WILEY & SONS, INC. ISBN-13: 978-0470036662

[9] Kumari, A., Tripathi, R., Pal, M., Chakraborty, S. (2012). Linear Search Versus Binary Search: A Statistical Comparison For Binomial Inputs. International Journal of Computer Science, Engineering and Applications (IJCSEA), vol.2, no.2, pp29-39 DOI: 10.5121/ijcsea.2012.2203

[10] An Oracle Whitepaper - Leveraging the Power of Oracle Engineered Systems for Enterprise Policy Automation (2012).http://www.oracle.com/technetwork/apps-tech/policy-automation/learnmore/opaonengineeredsystemswhitepaper-1713414.pdf, last visited on December 15, 2013

[11] An Oracle Whitepaper - Linear Inferencing (2009). http://www.oracle.com/us/industries/public-sector/029743.pdf?ssSourceSiteId=otnen, last visited on December 15, 2013

[12] An Oracle Whitepaper - Oracle Communications Billing and Revenue Management Product Overview (2012). http://www.oracle.com/us/industries/communications/comm-brm-management-wp-1637483.pdf, last visited on December 15, 2013

[13] Business Rule Management System. http://www.hartmannsoftware.com/pub/Enterprise-Rule-Applications/BRMS, last visited on December 15, 2013

[14] Business Rules Management - How it works. http://www.fico.com/en/Products/DMTools/Pages/Business-Rules-Management.aspx, last visited on December 15, 2013

[15] Fields, J. (2006) Business Natural Language. http://blog.jayfields.com/2006/10/bnl-introduction-update.html, last visited on December 15, 2013

[16] Forgy, C. L. http://en.wikipedia.org/wiki/Charles_Forgy, last visited on December 15, 2013

[17] InfoWorld - World's fastest rules engine | Business rule management systems. http://www.infoworld.com/t/business-rule-management-systems/worlds-fastest-rules-engine-822, last visited on December 15, 2013

[18] Oracle Policy Automation. http://en.wikipedia.org/wiki/Oracle_Policy_Automation, last visited on December 15, 2013

[19] Oracle Policy Automation. http://www.oracle.com/technetwork/apps-tech/policy-automation/overview/index.html, last visited on December 15, 2013

[20] Rao, R. (2012). Accessing external data in a rules application. http://www.ibm.com/developerworks/websphere/bpmjournal/1202_rao2/1202_rao2.html?ca=drs-, last visited on December 15, 2013

[21] Rymer, J. R., Gualtieri, M. (2011). Market Overview: Business Rules Platforms 2011. http://www.i-sol.com.ar/web/esp/brm/papers/ForresterMarket%20Overview%20Business%202011.pdf, last visited on December 15, 2013

[22] Selvamony, R. (2012). Introduction To The Rete Algorithm. http://www.sdn.sap.com/irj/scn/go/portal/prtroot/docs/library/uuid/10dea1d3-fbef-2d10-0e89-a7447f95bc0e?overridelayout=true&49868865442396, last visited on December 15, 2013

[23] Taylor, J. (2008). Business Rules, Domain-Specific Languages and Models Everything Decision Management. http://jtonedm.com/2008/08/12/business-rules-domain-specific-languages-and-models/, last visited on December 15, 2013

[24] Turkish e-Health System. http://e-saglik.gov.tr, last visited on December 15, 2013

[25] Turkish National Health Data Dictionary. http://e-saglik.gov.tr/USVS.aspx, last visited on December 15, 2013

[26] Turkish SaglikNet Business Rules. http://e-saglik.gov.tr/SaglikNet/SaglikNetDokumanlari.aspx, last visited on December 15, 2013