Improving Agility in Business Applications using Ontology Based Multilingual Understanding of Natural Business Rules

Ammar Joukhadar
Hala Al-Maghout
Information Technology Faculty, Damascus University
ammarj@scs-net.org, hala237@hotmail.com

Abstract:

Business applications need to be agile i.e. easily and quickly modified in order to respond to the frequent changes of business policies due to regulatory and market changes. Thus, agility in addition to reduction in cost of maintenance and development are key motivations for the adoption of business rules methodology. Domain experts are responsible for specifying business rules which are input to business applications usually by programmers. In order to increase agility, domain experts must be able to specify business rules to business applications directly in natural language without programmers' intervention. In addition, understanding business rules in many languages is highly required, because business rules management systems are universal applications that need to support different languages. In this paper, we present a cost and time effective multilingual solution that improves agility in business application by enabling the domain expert to specify business rules to the business application directly in many natural languages using a novel approach to natural business rules understanding based on the business models and enriched metadata provided by Elixir MDA Framework

1 Introduction:

Business Rules are increasingly being used in the development of business applications. The key motivation for the adoption of business rules is to use them to adapt the business application easily and quickly in the face of frequent and rapid changes in the market. Business rules define business policies which are specified by domain experts and input to the business application by programmers. The need for programmers to transform business rules specified by domain experts into a programming language poses cost, time and accuracy problems. First of all, much time and cost is spent educating programmers on the details of the business as they find it difficult to fully understand the business logic. This difficulty arises from the fact that programmers do not possess the business model of domain experts which forms the context in which business rules are understood. As a result, programmers may misunderstand the domain experts and this causes errors in the implementation of business rules. In addition, business rules are frequently changed because of competitive and regulatory pressures. The need for programmers to implement those changes every time they are needed takes time and raises costs, which directly impacts system agility. Because of the previously mentioned problems, a solution that enables the domain expert to specify business rules in natural language directly without programmers' intervention will have many precious benefits. First, it will make the business application more easily and quickly modified by the domain experts themselves, which allows the IT system to stay up-to-date with the current business policies without the need for a long change implementation process. Second, it will eliminate cost and time needed to educate the programmers on the details of the business. Third, it will ensure business rules accuracy and consistency with business policies. Last, it will help to document business rules in natural language and thus make them understandable and modifiable by other domain experts.

Building a multilingual system for understanding natural business using traditional methods is not practical. Traditional methods in natural language processing divide the analysis of natural language text into number of levels [1]. Many language resources are required to perform this analysis from dictionaries and morphological analyzers to syntactic analyzers and experienced linguists. Those resources need time and cost to be available and they are language dependent i.e. each language requires its own resources. Besides, business rules management systems need to be multilingual, because they spread worldwide and must be adaptable to the requirements such as regulations and language of each country.

Solutions previously followed to enable domain experts to specify business rules in natural language are mainly divided into two kinds of methods: methods that use

natural language templates and methods that use controlled natural language. The first kind, which is used by many commercial business rules management systems such as Drools [2], uses natural language templates which are predefined natural language phrases that represent conditions or actions and each has an equivalent in a programming language defined by programmers. Although this solution is easy to implement, it does not solve the problem completely; it still needs programmers to define the templates and implement any change in them. In addition, the domain expert is restricted to use those templates in expressing his rules and thus he has to learn and memorize them, which is not an easy task especially as the number of the templates increases. The second kind, which is used in commercial systems such as Haley Authority [3], uses controlled natural languages [4] .Applying language analysis on a controlled natural language is simpler as much ambiguity and complexity is reduced, which reduces cost and time needed to build such systems. Although using controlled natural language gives domain experts a considerable flexibility, it still requires language resources and designing a controlled natural language for each supported language which increases cost and time required as the number of languages to be supported increases. In addition the domain expert has to learn the grammars and structures of the controlled natural language in order to be able to use it properly to express natural business rules, which limits his freedom.

In order to understand a natural language sentence, we must have a world model (ontology) which represents a particular context in which the sentence is to be evaluated [1]. The domain expert specifies business rules in the context of business model or domain ontology. Thus, in order to enable the computer to understand natural business rules, the business model must be transferred first to the computer. MDA (Model driven architecture) provides us with the means of this transfer. MDA is an approach to system development, which increases the power of models in that work [5]. Systems based on MDA build the business models before writing the programs that use them, which enables to use those business models as ontology required to understand business rules. Elixir Framework [6] is MDA based and provides us with the required business models along with enriched metadata about them which made it a suitable environment to develop our solution.

In this paper we propose a solution that improves agility in business applications by enabling the domain expert to input business rules to the business application in many languages without programmer intervention. Our

solution uses a novel approach to natural business rules understanding based on business models provided by Elixir MDA Framework [6] in the form of UML class diagrams along with their Elixir enriched metadata. Our solution provides many features. First of all, it is multilingual; it enables the domain expert to write business rules in many natural languages. It is also cost and time effective because it needs no language resources, requires no syntactic or morphological analyzers and learns morphological rules of the language through interaction with the domain expert. In addition, it gives the domain expert the freedom to express rules using his own language and does not limit him to the use of specific grammatical structures or predefined phrases.

The next section describes our approach. In sections three through six we describe the details of our method. Section seven presents our algorithm. Section eight describes system configuration and rule authoring through examples in Arabic and English. In the last section we conclude with future work.

2 Approach:

We base our approach on the idea that each business rule is understood in the context of a business model. Thus, each business rule should reflect the business model and conform to it. This enables us to use the business model to guide the natural business rules understanding process. Elixir Framework provides us with rich information about the business model in the form of model metadata enriched with Elixir stereotypes and tagged values required to guide the understanding process. In addition, Elixir Framework provides the domain expert with the ability to specify the natural language representation of the business model which makes the business model a dictionary of business terminology. Natural business rule understanding starts by first trying to recognize business concepts and their properties and relations referenced in the rule by mapping their natural language representation specified in the business model with natural language expressions in the natural business rule. Then it tries to construct the logical expressions that constitute the conditions of the rule based on heuristics that use metadata about the business model in addition to logical rules of the conditions of the business rule. Our approach dependence on rich information about the business model enables us to understand the natural business rule without the need for syntactic analysis, which makes our approach language independent and provides the domain expert with the flexibility to use different grammatical structures. We overcome the problem of mapping

different morphological forms of the words in the business rule by enabling the system to learn the morphology of the language through interaction with the domain expert and thus eliminating the tedious work of adding every morphological form of every word to the dictionary.

3 Ontology:

Elixir Framework takes into account several UML views. In our work we used the UML class diagram view as the business model which represents the context in which the business rules will be understood. UML class diagram view in Elixir Framework consists of the business concepts and their properties and relations in a specific business domain. Each business concept has a number of properties and operations. Each property takes a value of a type which is simple (numeric, string, date, boolean...etc.). Some properties have a set of predefined possible values from which they take their values. An operation is the same as a property but it has a number of parameters each of which takes a value of simple type. Each business concept has relations with other concepts. A relation relates a pair of concepts and it is of two kinds: one to one and one to many. One to one relation relates an instance of the concept to only one instance of the other concept. One to many relation relates an instance of the concept to a number of instances of the other concept. Elixir Framework provides rich metadata which is information about different constituents of the business model. This metadata plays an essential role in our approach to understanding natural business rules.

4 Representing Ontology with Natural Language:

Knowledge representation is a key issue in Artificial Intelligence. The way the knowledge is represented entails the ways the knowledge can be manipulated [7]. We found that representing our business model with semi structured natural language fragments defined by the domain expert is the most suitable representation, because words and expressions used to represent the concepts and their properties and relations in the business model automatically inherit their meaning from the way they used by the domain expert. In addition, this representation enables us to directly map natural language expressions used in natural business rules to natural words and expressions of the business model. Elixir Framework enables the domain expert to define words and natural language expressions that represent business concepts, their properties and operations, and

their relations in many natural languages. Those natural language expressions are mapped to the natural language expressions used in natural business rule as the first step of the natural business rule understanding process. Thus, the natural language representation of the business model provided by Elixir Framework plays the role of a dictionary that contains the business terminology defined by domain experts in many languages.

5 Business Rules:

Business rules are abstractions of the policies and practices of a business organization [8].There are two fundamental categories business rules: structural rules and operative rules [9]. Structural rules are rules about how the business chooses to organize the things it deals with. Operative rules are rules that govern the conduct of business activity. In our approach we handle operative business rules; we suppose that structural business rules that define the business model have already been defined by the domain expert and entered to the system in the form of UML class diagram. Each business rule has a business concept from the business model on which it will be applied. We call this concept the main concept of the rule. The domain expert specifies the main concept for each business rule he writes. Each business rule consists logically of one or more conditions that has logical "and" between them. Each condition has a boolean value, either true or false. The result of the rule is a boolean value that is the result of applying the boolean "and" on the boolean values of all the conditions of the rule. The rule conditions are applied on the instances of the main concept of the rule. Each rule condition is logically either an access to a boolean operation or property in a concept in the business model, or a comparative phrase that contains a comparative operation between the values of two expressions. To access a property of a concept in the business model in the business rule, all related concepts from the main concept to the owner concept of this property must be mentioned. Accessing an operation is similar, but the values of its parameters must be specified. The comparative condition consists of a comparative operation (>, <, >=, <=, =, !=) and two expressions that represent the left and right sides of the comparative operation. The left side of the comparative operation is a property or operation access expression that returns a comparable value. The right side of the comparative condition is either a constant or a property or operation access expression. There are number of logical operators can be used in rules such as the negation operator to invert the value of a condition and quantifiers which

have two kinds: existential quantifiers and universal quantifiers.

To author business rules in natural language the domain expert uses natural language expressions, which he has already defined to represent the business model, in addition to natural language expressions that represent the logical operations (comparative, negation, quantification) which have also been defined by him.

6 Morphology Learning Using String Edit Distance:

Although the domain expert uses natural language expressions from business model to express business rules, he might not use it in its exact form; he might use different morphological forms of some words. In addition, some morphemes in some languages carry the meaning of some logical operations like negation and comparative operations. To avoid the need to add every possible morphological form of every word to the dictionary, which is very tedious work for the domain expert and increases the size of the dictionary and makes it difficult to maintain, we enabled the system to learn different morphological forms of the words of any language through its interaction with the domain expert. In order to enable the system to learn morphology, we need a metric to discover the morphemes. We used string edit distance (SED) metric [10] to morphologically segment words and identify language affixes. String edit distance determines the distance between two strings measured by the minimum cost sequence of "edit operations" needed to change the one string into the other. The edit operations allow changing one symbol of a string into another single symbol, deleting one symbol from a string, or inserting a single symbol into a string. A solution based on dynamic programming computes the distance between strings in time proportional to the product of the lengths of the two strings [10]. Consequent edit operations in a specific place form an affix. A recent work [11] used string edit distance as a bootstrapping heuristic in unsupervised learning of morphology. In our approach, we used string edit distance in supervised learning of morphology. When the system fails to match a word from the natural business rule with a word from natural expressions of the business model, it tries to hypothesize the words from the expressions of business model which are most similar to the given word based on string edit distance and then it suggests those words to the domain expert. The domain expert is asked to choose the word he means from those suggested words. Then affixes discovered using string edit distance is added to the learnt affixes of the

language and the system will be able to generalize this case to other cases and thus it learns the affixes of a language through its interaction with the domain expert.

7 Algorithm:

Our algorithm follows the following steps to understand natural business rule:
1. Extract the conditions of the natural business rule.
2. Apply the following on each extracted condition:
   2.1. Consider the main concept as the current concept.
   2.2. Match natural language expressions that represent properties, operations and relations of the current concept in the business models with the natural language expressions used in the condition using string edit distance and the learnt language affixes.
   2.3. For each concept matched in the previous step,
       if the concept is complex then
            make the found concept the current concept and then go to step 2.2
   2.4. For each operation matched in 2.2 find the values of the parameters of the operation based on their types.
   2.5. Find quantifiers and negation operation and comparative operations in the condition along with their operands.
   2.6. Check the validity of the interpretation found based on the business model metadata.
3. When more than one valid interpretation is found, choose the one that matches longest natural language expressions from natural language expressions in the business model.
4. Convert the interpretation of the condition to a logical format.

When there is an error in the rule, no valid interpretation will be found; in this case the system tells the domain expert of the place and cause of error and offers suggestions to correct the error according to writing context. Also, the system uses metadata about the business model to offer context sensitive help that guides the domain expert during writing rules and helps to ensure rules correctness and compliance with the business model.

8 Examples:

In this section we describe the initial language configuration done by domain expert along with a scenario of interaction between the domain expert and

the system during rule authoring through two examples; the first in Arabic and the second in English.

We suppose we have the business model illustrated in the UML class diagram below, which represents a simplified sub diagram of a class diagram that represents a bank system. This business model is specified by the domain expert and input to the application.
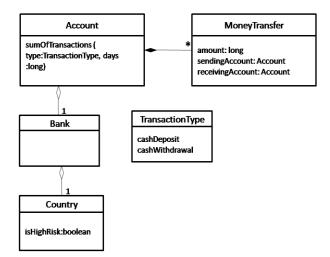

Figure1: UML class diagram

The Elixir MDA Framework enables the domain expert to specify the natural language representation of the business model above in many languages. For example the Arabic natural expression of the concept "Account" is "الحساب" and the English natural expression of the property "sumOfTransactions" from the "Account" concept is "sum of transactions".

Before the domain expert can write business rules, language configuration is required. Language configuration consists of the following steps for each language:

- Specifying the natural language expressions that represent the logical operators (>, <, == ...etc.) used in writing rules.
- Specifying the type of each logical operator which is either unary or binary. Binary operators take two operands, whereas unary operators take only one.
- Specifying the order of the operands of the logical operators relative to each other and to the operator, which may vary from language to language.

After this simple configuration which can be easily done by the domain expert, he will be able to write his business rules according to the following steps:

- Determining the main concept of the rule.
- Specifying the rule name which reflects the purpose of the rule. The rule name is common to all rules which have the same purpose.
- Writing the rule.

8.1 Arabic Example:

We suppose that language configuration of the Arabic language has been done according to the aforementioned steps. The domain expert can then write a business rule which identifies suspicious activities according to the following steps:

- Main concept: "التحويل المالي".
- Rule name: "تحويل مالي مشبوه".
- rule:
  "الحساب المصدر للتحويل المالي هو حساب في بنك من دولة خطيرة و قيمته أكبر من أو تساوي 100000"

The rule consists of two conditions. The system is unable to understand the second condition because it failed to recognize the word "قيمته" which is equivalent to the word "القيمة" with a prefix "ال" and a suffix "ه" which causes the letter "ة" at the end of the word to be replaced by the letter "ت". In this case the system searches for the natural language expressions of the Account concept which is similar to a natural language expression in the condition according to the string edit distance. The system then suggests the words "القيمة" and "قيمته" as equivalents. The domain expert accepts the suggestion in this case and the system learns that "ال" comes as a prefix and "ه" as suffix and in this case it causes "ة" at the end of the word to be substituted by "ت". When words that carry those affixes occur again, the system will recognize them immediately using the knowledge it acquired by learning through interaction with the domain expert. The system is now able to understand the rule and convert it to the following logical format:

sendingAccount.bank.country.isHighRisk && amount
>= 100000

8.2 English Example:

The domain expert inputs the following English rule:

- Main concept: account.
- Rule name: suspicious activity.
- Rule: sum of transactions of cash deposits during 30 days is greater than 100000.

The rule consists of two conditions. The system is unable to understand the second condition because it failed to recognize the expression "cash deposits" because of the "s" suffix at the end of the word "deposit". In this case the system knows using the metadata about the business

model that the operation "sum of transactions" takes a parameter which is the transactions type which is either "cash deposit" or "cash withdrawal". Thus the system searches for a natural language expression in the second condition which is most similar to the expressions that represent transaction type and suggests "cash deposits" to be equivalent to "cash deposit". The domain expert approves on the suggestion and the system learns that "s" is added as a suffix. Now the system is able to understand the rule and as a result it converts it to the following logical format:

account.sumOfTransactions ("cashDeposits", 30) >
100000

9 Conclusion and Future Work:

We presented a solution that improves agility in business application through enabling the domain expert to specify business rules in many natural languages without programmer assistance. Our solution has the following features:

- It uses a novel approach to natural business rules understanding based on business models in the form of UML class diagrams provided by Elixir Framework.
- It is multilingual. It is easily configured by the domain expert to support a new natural language.
- It is time and cost effective. It needs no language resources thanks to the novel approach that understands the natural business rule without syntactic analysis and learns the morphology of the language through interaction with the domain expert.
- It provides context sensitive assistance that guides the domain expert during rules authoring and ensures rules correctness.
- It gives the domain expert the freedom to use flexible natural language to express his rules because it does not limit him to the use of specific grammatical structures or predefined phrases.

In future work we plan to enable the domain expert to test the result of applying business rules by providing him the ability to query the business rules in natural language and display the results of the query in natural language. We also plan to add improvements to the technique used in learning the morphology of the language in order to increase the confidence of correct affixes learnt and reduce invalid affixes.

References:

[1] Allen, James F.(1995) Natural Language Understanding, The Benjamin/Cummings Publishing Company, Menlo Park, California,(Addison-Wesley Publishing Company, Reading, Massachusetts).

[2] Proctor,M. ,Neale, M., Lin, P.,and Frandsen, M., Drools Documentation, <http://labs.jboss.com/file-access/default/members/jbossrules/freezone/docs/3.0.4/html_single/index.html#preface>.

[3] Haley Systems, Inc. Haley's Natural Language Interface, 2006.

[4] Controlled Natural Languages,< http://www.ics.mq.edu.au/~rolfs/controlled-natural-languages>

[5] OMG, MDA Guide Version 1.0.1(2003), omg/2003-06-01, 12th June 2003.

[6] Elixir for intelligent software (2006) Elixir MDA Framework, www.al-ixir.com.

[7] Davis, R., Shrobe, H., and Szolovits, P.What is a knowledge representation? AI Magazine 14, 1 (1993), 17.

[8] Business rules, <http://en.wikipedia.org/wiki/Business_rules>.

[9] OMG, Semantics of Business Vocabulary and Business Rules (SBVR), Second SBVR Interim Specification without change bars dtc/06-08-05.

[10] Robert A. Wagner and Michael J. Fischer. 1974. The string-to-string correction problem. Journal of the Association for Computing Machinery, 21(1):168–173.

[11] John Goldsmith, Yu Hu, Irina Matveeva, and Colin Sprague.2005. A heuristic for morpheme discovery based on string edit distance. Technical Report TR-2205-04, Department of Computer Science, University of Chicago.