I J C T A, 9(19) 2016, pp. 9129-9139
© International Science Press

A Tool for Converting Business Rules From SBVR to Jess

Neha Kajal*

ABSTRACT

The relationship of IT personnel and the business persons is very paradoxical, on one side joining hands with IT provides automated system for the business requirements and on another side it limits the dynamic nature of business to accommodate flexibility in its demand with the changing market conditions and requirements. To the business people the intervention of IT is a very long drawn out process as once the translation is being done for the business policies, it requires a whole complex procedure to modify anything later if required. In order to overcome this time elapse and provide higher degree of flexibility, Business Rule Management System (BRMS), bridged the gap between IT group and business persons by providing control to the business persons for their rule formation and this is where our proposed approach becomes relevant.We have proposed a tool which can convert business rules from CIM layer to PSM layer of MDA which has not been done before and the respective representations chosen as the source and destination platform overcomes the various accuracy and efficiency misleads of previously proposed approaches in interlayer conversions of MDA.

Keywords: Business rules, Business Rule Management System, Model Driven Architecture, SBVR, JESS

I. INTRODUCTION

A business rule specifies what should be done and when in order to undergo a systematic procedure for completion of any business task. It lays down directive guidance for a set of predefined actions which need to be performed on some constraints to govern the business according to the business policies defined for it. A business rule is a declarative atomic entity which cannot be further decomposed [1] and can take up various representations like Natural Language Representation [2], Production Rule Representation [3], SBVR Representation [4], XML Representation [5] and JESS Representation [6]. The various representation system for the rules defined above are organized in MDA (Model Driven Architecture) framework [7],which is a standard developed by OMG group. MDA as shown in Figure 1 is composed of 3 layers:

CIM: *Computation Independent Model* views system in an encapsulated form i.e. the primary user is not aware about the details of the system or the knowledge required for the proper functioning of the system and hence it acts as a connecting point between the experts of domain and the experts of modeling and design that together completes the whole requirement of the system.

PIM: *Platform Independent Model* views system as a system which is independent of platform such that it is compatible with similar type of different platforms.

PSM: *Platform specific Model* views system as a platform specific or platform dependent model which is compatible with a particular platform specified by the details of PIM and the specifications how that particular platform is to be used.

The OMG MDA defined the above levels of model with the idea of automated or guided transformations between the layers which is the area of ongoing research. Various transformations from CIM layer to PIM Layer have been developed where Natural Language is used as the source to input the business rules and

* Department of Computer Science and Engineering, The NorthCap University, Gurgaon, Haryana, India, *E-mail: nehakajal92@gmail.com*

are converted to UML Class model in PIM layer [8, 9, 10, 11]. The major reason identified for the accuracy drawback for these transformations is the informal and ambiguous nature of the Natural Language used as a source platform which has been overcome by the more formal specification of Natural Language i.e. SBVR [12]. SBVR forms the background for our tool and is therefore explained in the section II.

The rest of the paper is organized as follows. The major platforms chosen are explained in section –II and section-III. Problem statement and its relevance is explained in section IV and Section V respectively. Proposed methodology and results are presented in section VI and section VII followed by limitations and future work in section VIII and section IX.

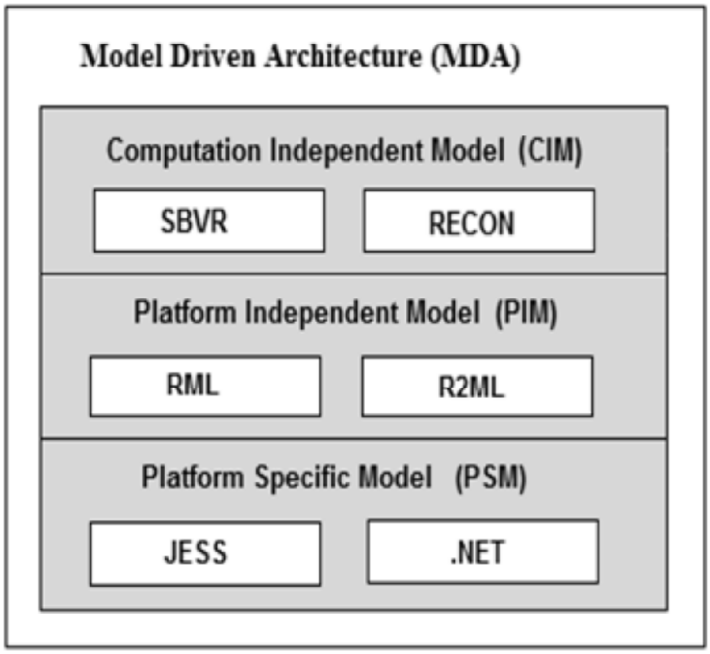

Figure 1: OMG Modelling Layers

II. SEMANTICS OF BUSINESS VOCABULARY AND RULES

Semantic Business Vocabulary and Rules (SBVR) [12] is a standard deployed by OMG in 2008 to specify the formal structure of rules for IT and Business people. It is based on natural language representation of business rules related to business policy so that business people can document the requirements and changes in their business through business rules in their own language which is easy to understand. In software modeling, SBVR is a modern and an unambiguous way of specifying business requirement not just because of its easy development and modification by non IT people but also due to its efficient representation which can be easily processed by machine because of its higher order logic.

2.1. SBVR Vocabulary

A business vocabulary [12] includes all basic elements which can describe the business entity. A business rule is composed of business concepts shown in Figure 2 which are linked with each other to add up the meaning to other concepts and formally specify the business rule. Following types of SBVR vocabulary are analyzed and used in our proposed approach and are explained below:

- *Noun Concept*: A general concept that includes all the common nouns and possesses a set of characteristics which distinguish it from all other object types. E.g. Library, student, etc
- *Individual Noun*: A individual concept which is represented by proper noun or quantified noun and corresponds to a specific object. E.g. Bangalore is the tech hub of India. Here, *Bangalore* is the individual noun for the noun concept *India.*

- *Verb Concept*: A verb concept specifies the relationship of noun concept with other object types and is represented by auxiliary and action verbs. E.g. Library has books.Has defines the relation between *library* and *books* and forms the verb concept.
- *Characteristic*: It is a property of an object which is extracted to be included in the fact type and is represented as "a-property-of" fact type. Characteristic is essential to specify the feature of the concept as shown in Figure 2. E.g. John is not allowed to vote as his age is not above 18. Here, *age* is the property of a *person.*
- *Fact Type*: Fact type is the combination of noun concept and verb concept and forms the template for the facts.
- *Keywords*: Fact types forms the rule by adding specific terms known as keywords which constraints the violation parameters of the rule. E.g. It is necessary that visitor has a ticket for entry. Here *it is necessary* is a keyword that defines the constraint level for the rule i.e. the entry will not be allowed to the visitor if ticket is not there and hence imposing strict validation rule.

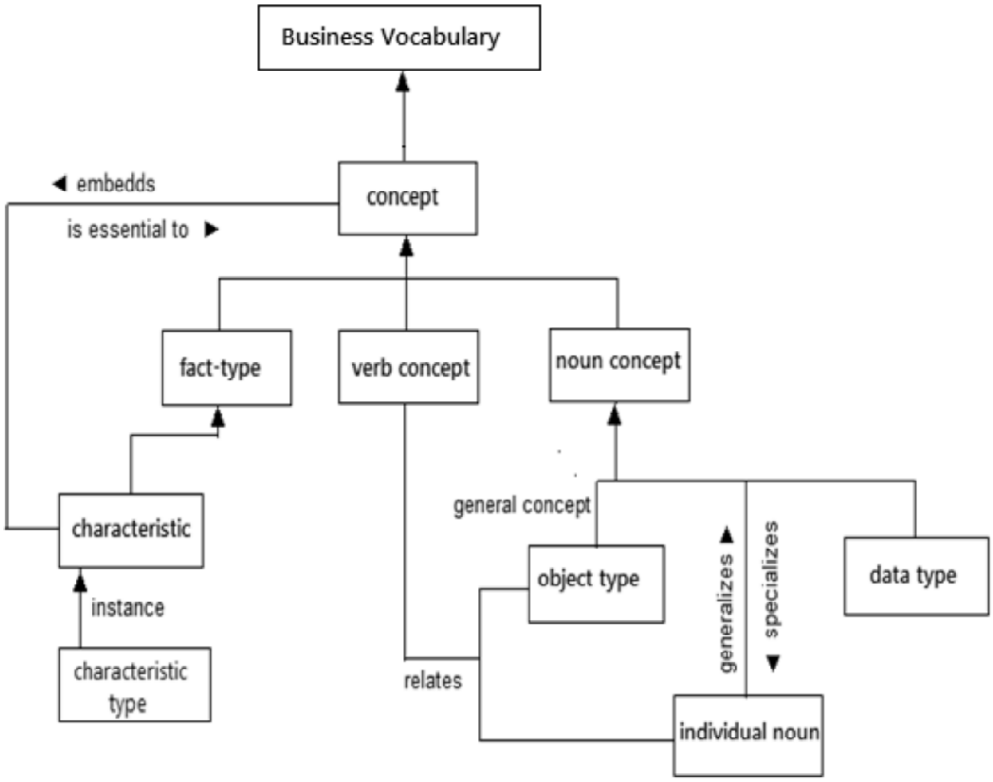

Figure 2: SBVR Vocabularies

2.2. Types of Business Rules in SBVR

The structure of a particular business and its code of conduct can be expressed in SBVR [12]in the form of a business rule which can be categorized in one of the following types and are shown in Figure 3:

- *Structural or Definitional Rule*: These are the necessity claims used to define an business policy of an organization and forms the structure which cannot be violated in order to conduct the defined operational setup. They basically use two operators: *it is necessary that…., it is possible that…*
- *Operative or Behavioral Rule*: These are the obligation claims imposed on business for its conduct that can be violated and uses the operators: it is obligatory that…, it is permitted that…

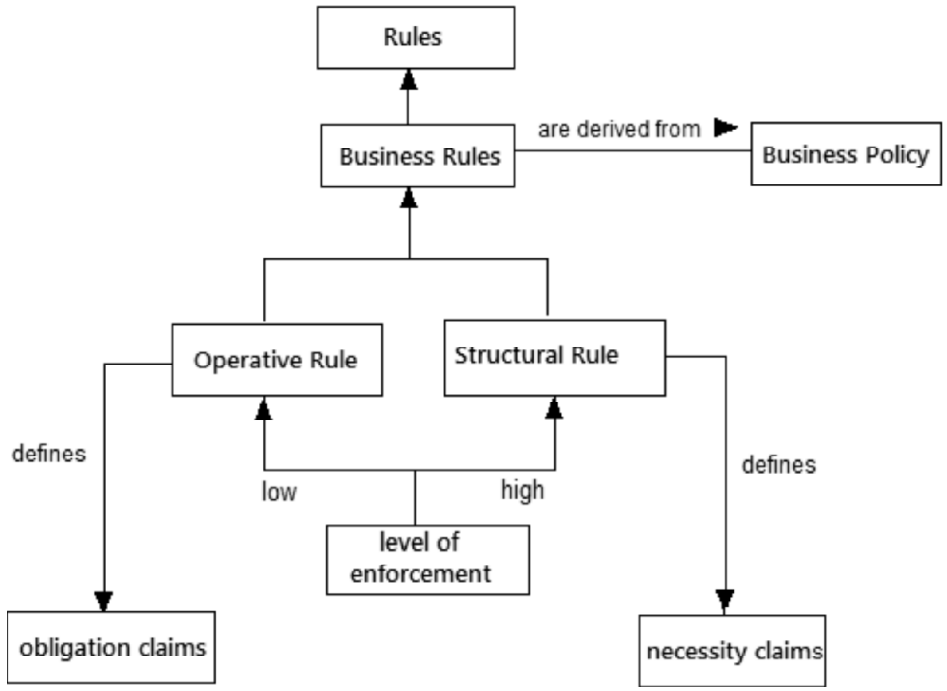

Figure 3: Types of Business Rule in SBVR

2.3. Semantic Formulation of SBVR

SBVR business rules are semantically formulated with logical formulations[12] like quantification and logical Operations which are based on first order logic and also beyond first order logic .Modal operations are also included which belongs to Intensional logic.

III. JESS

Jess is a *rule engine* for the Java platform in which we specify logic in the form of rules using the Jess rule language [6]. It is a platform specific language and lies in the third layer of MDA. Architecturally inspired by CLIPS, it has a LISP like syntax. JESS rule engine is composed of a working memory, rule base and an inference engine. Working memory and rule base acts as the database for storing the rules and the knowledge required to implement the rules. Detailed description is given below:

3.1 Working Memory

Every JESS Rule Engine holds a set of information or knowledge to process the rules and this collection of knowledge are known as *facts.* Rule engine only responds to the actions (addition, deletion, modification) made to this working memory, thereby constituting the important part of JESS. We can think of fact as a row in a database table that holds the instance of various properties. Facts in working memory can be divided in three categories namely *ordered facts, shadow facts and unordered facts.*

Unordered facts: Facts are specified in accordance to the template defined for a particular fact type, containing the named fields or slots to specify the input data.

E.g. *(deftemplate* student*(slot* name*)(slot* sex*)(slot* age*))*
Fact: *(*student *(*name *"John") (*sex *Male)(*age *18)*

Shadow facts: Shadow facts are similar to unordered facts but they allow to link with java class objects and loads them in JESS working memory

Ordered facts: Ordered facts are similar to the jess lists where the head of the list i.e. first element acts as a category for the fact and no named slots are there to define the fact

E.g. (student “John” Male 18)

Facts in accordance to the template shown above, can be added, modified and deleted from the working memory in the following ways.

- *Assert* – Add fact to working memory.

  E.g.(assert (borrower(id 1)(name “John”)(designation “faculty”)))

Here assert is the keyword used to add the fact followed by the head of the fact which is borrower in this case. Slots are present within the fact head which are associated with the particular values defined for that fact

- *Modify* – A fact asserted can be modified with the modify keyword followed by the fact id which is associated with every fact asserted in the working memory and can be obtained as: (bind ?factid (assert (borrower (name “john”)(designation “faculty”)))).

  E.g (modify ?factid (designation “student”))

- *Retract* – Remove existing facts.

  E.g. (retract ?factid))

3.2. Rule Memory

Rule Memory in JESS consists of the rules defined for execution. These are similar to if…then statements in procedural language but they are not used in procedural way. JESS rule has an LHS (if part) which contains the condition to be satisfied in order to execute the rule and the RHS (then part) for the course of actions to be undertaken i.e. function calls. These are defined using the defrule construct.

```
E.g.  (defrule admission
      (student {age > 3})
      =>
      (printout t “Admission granted in Nursery” crlf))
```

Here admission is the name of the rule, which constraints the condition that student age should be above 3 to perform an action i.e. grant admission in nursery

Action that are to be executed in the rule can be defined as function using deffunction construct.

```
E.g.  (deffunction minimum (?x ?y)
      (if (> ?x ?y) then (return ?y)
      else (return ?x)))
```

Here minimum is the name of the function with ?x and ?y as the variables passed as the arguments.

3.3. Execution Mechanism in JESS

Rules are executed whenever LHS or condition part gets satisfied with the facts in the working memory by pattern matching. This pattern matching in JESS is done with the specialized algorithm known as RETE Algorithm which is discussed in detail in next section.

3.3.1. RETE Algorithm

Rete match Algorithm is an efficient way of mapping large number of patterns to the large number of objects. It determines all the objects matched to a particular pattern [13].

- Rete algorithm is based on the fact that only few facts are added, changed or removed at every step in the process of inference.

- Instead of doing all these comparisons every time only new facts added can be taken into consideration which is the approach taken in Rete algorithm.
- Rete looks for changes to match in each cycle.

Rules are defined in *if then* form where if part of the rule is called its LHS (Left hand Side) and then part as the RHS (Right hand side). For the processing of rules the LHS part of the rule is compared with the facts which define the data on which the rules operate. The whole mechanism has been shown as a flowchart in Figure 4. and is explained below**:**

1. *Matching*: This phase includes the processing of rules by comparing the input rule with the facts in the working memory and finding out the rules which are satisfied. The matched rules are then added to the conflict set.
2. *Conflict Resolution*: There may be the case that multiple rules are satisfied in the matching process which are then placed in the conflict set. Conflict resolution is the process by which the rule with highest priority is selected for execution and in case no rule has satisfied the interpreter is halted. In the next iteration conflict set is referred by the facts
3. *Execution*: The rule which is selected from the conflict set is executed (fired) which may lead to generation of new facts to be added to the working memory and the process continues forward with the updated working memory.

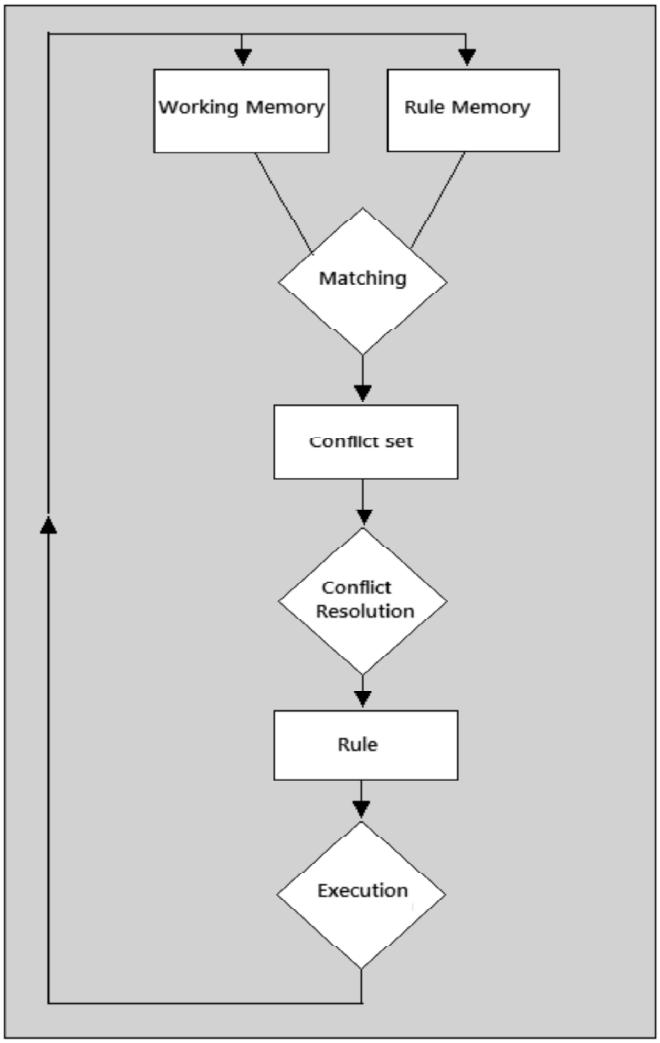

Figure 4: Flow chart Rete algorithm

3.4. Efficiency of JESS

JESS is an efficient rule engine to use if the system is bound to continuous updation with new facts. Its comparison with previous approaches can be analyzed below:

Earlier Approach: Mapping of rules to facts was done by matching all facts to each rule which takes high computational time in pattern matching

Time Complexity = O (n*m^p)
n: number of rule, p: premises in a rule, m: number of facts

Efficient Approach: The mapping process of a rule to the facts is not done in a regular procedural way as done in another procedural language rather JESS uses a special algorithm known as *Rete Algorithm* which speeds up the whole process with high order magnitude.

Time complexity = O (n)
n: number of rules

IV. PROBLEM STATEMENT

Taking into consideration the clause 9.2 of Business Rule Manifesto [14], *"Business People should have tools available to help them formulate, validate and manage rules"*, we have defined our problem as the transformation of business rules from CIM layer to the PSM layer as shown in Figure 5, taking source platform as the SBVR (overcoming the accuracy misleads of prior transformations that did not include SBVR as the source platform) and the final platform as JESS. The various advantages of taking Jess as the intended platform have been already explained above.

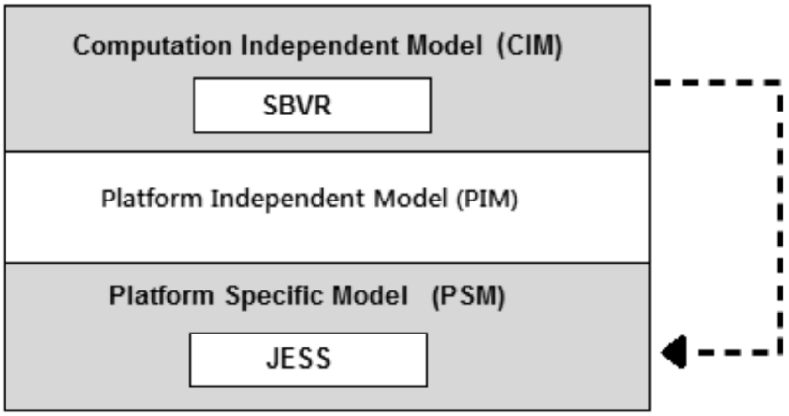

Figure 5: Transformation of SBVR to JESS

V. WHY SBVR TO JESS

In the field of Business Modeling, various transformations within the layers of MDA have been proposed. It includes conversion of Rules from CIM layer to PIM layer but there is no direct conversion from CIM layer to the PSM layer of MDA till now, which is the major motivation for the proposed research work. This transformation will not just give the direct mapping within the layers of MDA but also gives a streamlined representation of Business Rules which are efficiently processed due to the intending platforms chosen for source and destination.

VI. PROPOSED METHODOLOGY AND SOLUTION APPROACH

In our proposed approach, we will be designing a automated tool which will take Business Rule in the form of SBVR vocabulary as an input from user and gives a JESS representation as an output giving an automated structure to process the rule. We are assuming that the user has an basic knowledge of SBVR .

This approach works in phases given below which are explained further:

1. Identifying and selecting a subset of SBVR to be taken as input
2. Designing the input screen for the SBVR Rules
3. Defining a set of Rules for analyzing the input SBVR to be converted to JESS
4. Identifying the Output in JESS
5. Executing the output File to the results.

6.1. Identifying and selecting a subset of SBVR to be taken as input

Due to the wide spectrum of SBVR, it's not possible to process all SBVR constructs as an input within this limited time frame. So a subset of language is identified and chosen to be taken as an input for implementation of the rules to be converted to JESS. Subset is identified in accordance with the semantic formulations that has been defined for SBVR.

- Universal and existential quantifiers have been used in our approach as quantification formulation
- Logical operators of type P and Q , P or Q, Q if P, if P then Q have been used where P and Q are the expressions of the proposition of the rule.
- Modal operators like 'may' and 'must' have been included in our work which are identified as 'it is necessary' and' it is obligatory' respectively.

6.2. Designing Input Screen for SBVR Rules

On selecting the corresponding logical operator for the rule, vocabulary of the rule is given as input in the screen so as to process the rule. The input screen includes the various concepts explained before like noun concept, verb concept etc and need to be identified by user

6.3. Conversion Rules for mapping SBVR to JESS-

Once the rules are provided as a input in the input screen, a set of rules will be applied in order to map the input SBVR semantics to generate semantic model of output form i.e. JESS. These mapping rules forms the basis of conversion and are illustrated as follows:

- Entity (Noun concept) becomes the name of the fact template used to define facts as fact defines the data about the fact name or head which should be the noun concept of SBVR rule
- Every individual noun becomes the instance of facts which will be defined as slots in the fact template as individual noun is the instance of the noun.
- Attribute part of characteristic also become the instance of fact template as it describes the individual noun which is also the instance of fact.
- Verb concept of Q proposition for the logical formulation *if P then Q* and *Q if P* becomes a function as it defines the action to be taken on completion of P.
- The attribute component of characteristic of P in *if P then Q* and *Q if P* becomes the argument of the function as function requires the those instance on which the action is to be taken.
- Components of characteristic i.e. attribute, operator and value are used in the 'test' function of JESS to validate the condition
- Rules are mapped to the defined facts, and with every satisfied fact, rule is executed.

6.4. Identifying Output in JESS

- JESS Representation of rules works on creating rule definition, function definition and creation of fact template for facts in working Memory.
- Our tool will provide a auto generated view that will define the corresponding rule in JESS format and provides the skeleton of the rule which can be modified to run in JESS rule engine
- Function which will be called in action part are defined with the corresponding arguments retrieved from the SBVR vocabulary.
- Template for the facts will be defined so that the data for the Working memory can be provided by the user.
- Output will be shown in a text file which after modification can be embedded into JESS

6.5. Executing the output file in JESS-

The code generated in the text file can be executed in JESS after some rule specific requirements and the output will be shown corresponding to the rule.

VII. RESULTS AND SIMULATION

A rule with logical formulation 'and' is taken as input and the results has been shown below:

Rule : It *is obligatory that the visitor should have a ticket* **and** age *of visitor is above 10 years.*

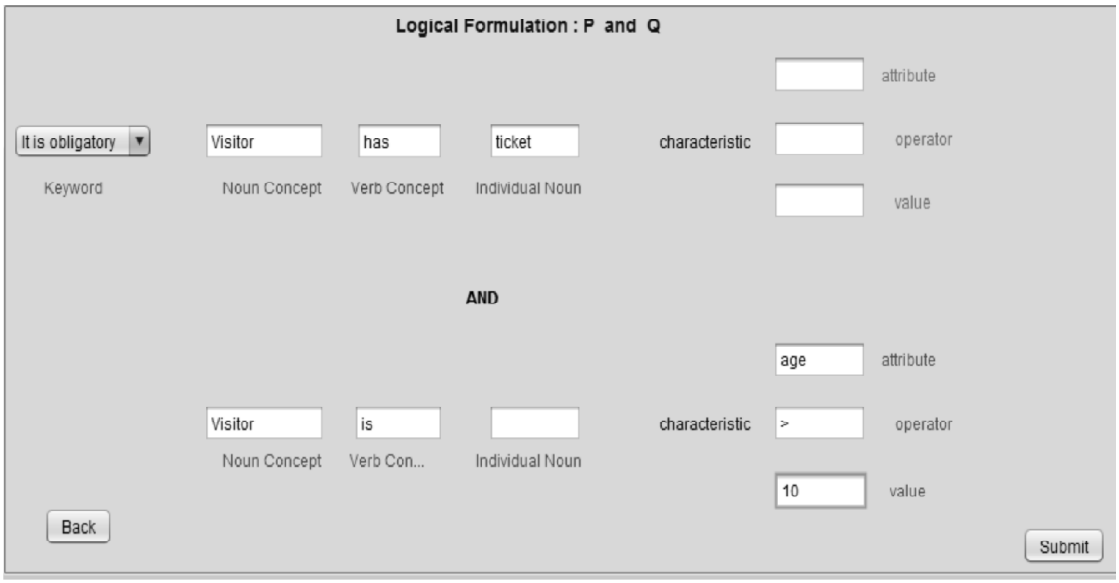

Figure 6: Input Screen for SBVR Rule

```
(deftemplate visitor(slot id)(slot ticket)(slot age))
(assert(visitor(id 1)(ticket 1)(age 12) ))
(assert(visitor(id 2)(ticket 0)(age 14) ))
(assert(visitor(id 3)(ticket 1)(age 65) ))
(assert(visitor(id 4)(ticket 1)(age 5) ))
(assert(visitor(id 5)(ticket 0)(age 9) ))
(assert(visitor(id 6)(ticket 1)(age 4) ))
```

Figure 7: Input Rule converted to JESS Rule in a text file

```
(defrule rule1

(declare (no-loop TRUE))

?var1 <- (visitor(id ?id1) (ticket ?ticket1) (age ?age1))


   (test (> ?age1 10))
   (test (eq ?ticket1 1))

=>
    (printout t "Rule 1 fired.......visitor with id "?id1 " allowed "  crlf)

)
```

Figure 8: JESS Rule compiled and Executed

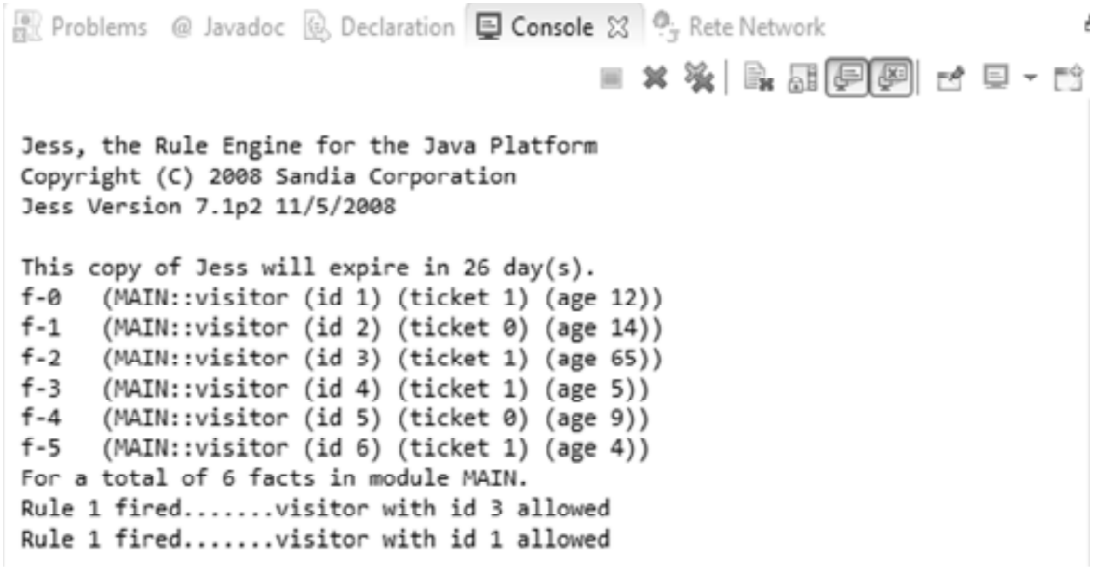

Figure 9: Output in JESS after Executing the Rule

VIII. CONSTRAINTS AND LIMITATIONS

- The implementation for the time being have been done for 2 propositions which can be easily scaled to multiple propositions in future.
- Negated Rules are not included in the implementation
- Brackets within the rules are not taken in account.

IX. CONCLUSION AND FUTURE WORK

A tool has been generated that convert Business Rules in SBVR to Business Rules in JESS and gives a text file as output which contains the implementation ready code. This code can be executed in JESS software with minimal changes. The tool can help in automated generation of JESS rules which are efficient to process in terms of time complexity and will help business people to formulate and manage their rules without intervention of IT people.

In Future, subset identified will be expanded to include all the SBVR constructs as the input and to give JESS executable file as the direct output of the tool.

REFERENCES

[1] David C. Hay and Keri Anderson Healy, GUIDE Business Rules Project, rev 1.2, October 1997.

[2] Tobias Kuhn, Sachin AceRules: Executing Rules in Controlled Natural Language.

[3] Production Rule Representation Request For Proposal, *http://www.omg.org/cgi-bin/doc?br/03-09-03.pdf*

[4] Imran Sarwar Bajwa, Mark G. Lee, Behzad Bordbar [2011] SBVR Business Rules Generation from Natural Language Specification. in proceedings of AAAI 2011 Spring Symposium-AI4BA, San Francisco, USA, Mar 2011, pp:2Rules.

[5] Benjamin N. Grosof, Yannis Labrou, Hoi Y. Chan[1999]A Declarative Approach to Business Rules in Contracts: Courteous Logic Programs in XML.

[6] Jess: The Rule Engine for the Java Platfor, Version 7.1p2,November 5, 2008.

[7] OMG, 2003, Model Driven Architecture, *available at, http://www.omg.org/mda/specs.htm*

[8] Mich, L. 1996. NL-OOPS: from natural language to object oriented requirements using the natural language processing system LOLITA. Natural Language Engineering. 2(2): 167-181.

[9] Harmain, H. M., Gaizauskas R. 2003. CM-Builder: A Natural Language-Based CASE Tool for Object-Oriented Analysis. Automated Software Engineering. 10(2): 157-181.

[10] Anandha G.S., Uma G.V. 2006. Automatic Construction of Object Oriented Design Models [UML Diagrams] from Natural Language Requirements Specification. PRICAI 2006: Trends in Artificial Intelligence, LNCS 4099/2006: 1155-1159.

[11] Bajwa I.S., Samad A., Mumtaz S. 2009. Object Oriented Software modeling Using NLP based Knowledge Extraction. European Journal of Scientific Research, 35(01): 22-33.

[12] OMG. 2008. Semantics of Business vocabulary and Rules. (SBVR) Standard v.1.0. Object Management Group, Available: *http://www.omg.org/spec/SBVR/1.0/*

[13] Charles L. Forgy, Rete: A Fast Algorithm for the Many Pattern/ Many Object PatternMatch Problem", Artificial Intelligence 19 (1982), 17-37)

[14] Ross R. G. (ed.) November 1, 2003, Business Rules Group, Business Rules Manifesto The Principles of Rule Independence, Version 2.0, *available at www. Business Rules Group org*