Two-level Architecture for Rule-based Business Process Management

Kanana Ezekiel
Vertiv Co
Accurist House, Baker Street
London, UK
Email: kanana.ezekiel@vertivco.com

Vassil Vassilev, Karim Ouazzane
School of Computing and Digital Media
London Metropolitan University
London, UK
Email: v.vassilev@londonmet.ac.uk,
k.ouazzane@londonmet.ac.uk

***Abstract*— One of the main challenges in Business Process Management (BPM) systems is the need to adapt business rules in real time. A serious obstacle is the lack of adaptable formal models for managing dynamic business rules. This is, due to the inadequacy of the models ability to describe the rule components, meta-rules, relationships and logical dependencies. To overcome this drawback, this paper presents a two-level rule-based approach to control BPM systems. The model accounts for logical representation of rules components and their relationships in Process-based Systems, as well as a method for incremental indexing of the business rules. The incremental indexing mechanism is described as an approach to control process execution and adaptation of business rules in real time based on rules propagation. Therefore this model provides a basis for an efficient and adaptable solution for managing business rules changes.**

Keywords-Business Process Management; Rule-based Systems; Meta-Rules; Rule Dependencies; Object-orientation.

I. INTRODUCTION

There are several definitions for business rules proposed in the literature. The most commonly used definitions of business rules are described [4]. In general, a business rule is defined as a rule that constrains, controls or structures some aspect of information, applications and processes in business. Business rules have been considered from many different perspectives. For example, business rules can be used by credit card companies to approve credit card applications. E-commerce businesses use business rules to understand customers shopping habits. Banks may use business rules to analyse data to establish suspicious or fraudulent online activities. Other applications that use business rules exist in areas such as insurance, airline, telecom, and manufacturing industries, etc.

In Business Process Management (BPM) systems, the behaviour of executing business process workflows is controlled by various business rules. Transforming and configuring dynamic and scattered business rules through process flow routines is very demanding. Typically, the organizations will have many business rules to enforce in their business processes. However, the business rules tend to change frequently. The most challenging task is to propagate these changes when there are multiple rule dependencies. In BPM systems, a change to business rules means reconfiguration of every process and other related rules. Inefficiency and inconsistency of the business rules are often unavoidable. The manageability and maintainability of the business rules is therefore becoming time consuming and a costly exercise. To address these problems, an adaptive Business Rules Framework for Workflow Management [1] has been developed. It is based on modelling of both business rule components and meta-rules, as well as business processes, flows and events in a unified manner, accounting for the structural patterns of description for various objects. This unified approach allows for the defining of the explicit and implicit relationships between business rules and indexing them incrementally, which eliminates the need for keeping a log of the changes.

This article has six sections to follow. Section II gives an overview of related work. Section III introduces the two-level approach for building the architecture of rule-based systems for BPM. Section IV describes the basic concepts used to construct the two-level architecture. Section V describes the current status of implementation of the whole framework. Section VI presents formal definitions and illustrates the use of dependency trees to define business rules relationships. Section VII concludes the article with a brief description of the next stage of implementation of the framework.

II. RELATED WORK

In recent years, substantial efforts have been made towards developing solutions to tackle the ever-growing problem of business rules adaptation. This section presents some methodologies and approaches adopted by existing rule-based systems. The existing commercial Business Rule Management Systems (BRMSs) integrate rule technology (rule engine) specifically for rule management. The IBM BRMS [2],[3] has the greatest business rules capabilities on the market. IBM BPM includes a customized version of IBM's Operational Decision Manager (ODM) tool for its business rules, which incorporates tools such as Eclipse to give inexperience programmers the ability to create and modify rules. The well-known IBM BRMS, WebSphere ILOG JRules [4], which provides a flexible tool for rule modelling, is now part of IBM ODM. While IBM BRMS provides integrated environment with rich and flexible tools for business rule modelling, there are some notable limitations in relation to the ability to manage changes to business rules. There is no straightforward way to change rules that affect more than one process. Multiple changes to business processes will need to be applied even for the simplest business rule changes. This seriously limits the business agility that business rules are designed to provide.

 ISBN: 978-1-61208-614-9

There is no separation of the various parts of the business rules components, i.e., Event, Condition and Action. This means a change made on the "condition" part of the rule will require invoking the whole rule. Separating rule components provides flexibility and increases performance, as only the part that needs changing is exposed on the business rule application. Henceforth, various parts of the rule need to be stored in appropriate structures to facilitate their management, similar to the existing structures for data in database systems. Rules are executed one by one in a procedural manner As a consequence, this creates additional work when the rules sequences change or when a separate rule in a particular sequence is changed. This complicates the ability to perform logical deduction hence its inability to manage changes to multiple business rule hierarchies [5].

C Language Integrated Production System (CLIPS) is specifically designed to facilitate the development of software to model human knowledge or expertise [6]. The CLIPS expert shell provides a platform where expert knowledge may be categorized as rules. To supplement its rules management capability, CLIPS works as an inference engine that enables it to perform the inference procedure whereby rules are interpreted to generate various actions as appropriate [7]. This mechanism employs the embedded pre-existing rules-based knowledge as "facts" to drive the firing mechanism of the inference engine and thereby produce a recommended conclusion to a problem. Even though CLIPS provide an interactive, text oriented environment for modelling rules, there is no dedicated knowledge base and, thus, facts are volatile and ~~are~~ purged from its memory as soon as its execution is terminated. To overcome this fundamental limitation, an external rule-base system must be added for a seamless integration with CLIPS. This adds to the complexity and cost for managing rules. The problem becomes worse when rules are scattered and changing.

Java Expert System Shell (JESS) [8][9] is another rule engine, originated from CLIPS and written entirely using Java. There is an extension called VISUAL JESS, which improves the comfort of using the tool. Pitfalls of JESS for dynamic systems are well documented [10].

Oracle BRMS is a leading Business Rules product, probably one of the finest products in the market. Oracle offers a Rule Author, a web-based graphical authoring environment that enables creation of business rules. In addition, Oracle provides an embedded business rules engine to its BPM system. The Oracle BPM application can add/remove and change the state of business objects in the working memory, and allows the rule engine to reason and update processes by triggering events or invoking specific processes based on the outcome of the rules. Like IBM BRMS, it faces similar limitations - it remains impossible to specify the dependencies between the rules based on the relationships between BPM objects. This causes multiple changes to be necessary to adjust already configured processes and update existing business rules even in the case of a simple rule change.

OpenRules [11], another powerful BRMS for rule-based application development, provides both complex Business Rule editor as well as a tool for building user interfaces. It allows the use of external tools such as MS Excel, Google Docs, and Eclipse IDE to create a complex, decision support system. OpenRules has similar limitations like the Oracle and IBM products. In this case it becomes even more complicated to deal with multiple changing rules as the rule management remains a tedious manual task.

JBoss Drools BRMS [12] is a sophisticated open source BRMS and has a lot of functionalities, which allow users to write and validate business rules that can then be added to Java Applications. While Drools distinguishes the structural elements of the rules syntactically it does not treat them in a special way semantically. At the same time, the users are free to define, classify, and modify the rules according to their specific requirements. Business rule components (i.e., Event, Condition, Action, etc.) are not defined as objects. This brings additional complexity in terms of change management. Furthermore, this work does not address the aspects of rules relationships and dependencies.

There are only a few proposals in the literature, which consider the business rules functionality and change [13]-[15]. Their focus is on rule execution and they do not provide support for modelling business rules. As a rule, they do not address how changes of business rules are managed. There is no clear, well-structured definition of the rule components and relationships; a common drawback of all industrial rule-based frameworks. We believe that a more flexible and efficient approach to manage business rule changes is required.

The next section outlines the two-level architecture of rule-based BPM systems, which addresses the above issues. Presenting a flexible approach for defining rules as objects, attributes/properties and relationships enabling logic and object programming power during rule implementation.

III. TWO-LEVEL ARCHITECTURE FOR BPM SYSTEMS

The formal model presented here is based on the understanding of the actual BPM system as an event-driven and constantly evolving process, with two functioning levels. The first level is the Process level, which governs the execution of business processes, while the second level, the Rule level, is a meta-level that controls the actual business rules. Features that are considered on the first level are: business processes, information and material flows, events, conditions and actions, which comprise the business domain. The users may intervene only via events that can trigger activities prescribed by the business rules. This way we can model manual, automated and fully-automatic processes as part of the business workflows. The second level considers the relationships between rules and dependencies between them, classifications of the rules, and meta-rules. The business rules are made up of events, conditions and actions, or the famous "When <event> If <condition> Then <action>" structure, whereas process execution level is made up of processes, steps, flows (material and information flows), roles, etc. For instance, if some events are observed during execution of a working process, then the corresponding business rules, which depend on these events, are triggered and lead to actions, which in turn perform the

 ISBN: 978-1-61208-614-9

transition to a new step, which may execute other processes or amend the parameters of the current process. The model uses business rules to glue together processes in a business process workflow. The rule control level provides a level of abstract "independence" between the two levels, suggesting that the rules can be changed without affecting the workflows that have been completed. The rule level controls the execution of business workflows adding the business logic to them. The business rules appear at all stages of the workflow from initiation, to execution, to termination. Based on the distinct roles they play in the workflow development, they can be organised in a taxonomic hierarchy: the Execution rules are divided into Flow and Process rules; the Flow rules are divided into Sequence, Fork and Join rules; and, Process rules are classified into Time-based and Non-Time-based rules.

IV. BASIC CONCEPTS OF THE TWO-LEVEL RULE-BASED ARCHITECTURE

This section presents basic concepts to support creation of objects, properties, and, relations for the model and meta-model. The concepts have been developed in a purely logical manner.

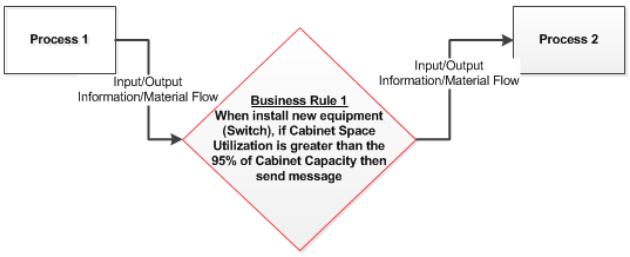

Figure 1. Example of a Business Rule.

Fig. 1 depicts a typical business rule. From such rule, the following concepts can be identified:

A. Business Objects

The business objects are the building blocks for implementing business rules and business processes. The following types of objects can be distinguished:

- **Processes:** Building blocks of the business workflows. *Examples:* Process1 (Manage Cabinet Space Availability) and Process2 (Order Cabinet).
- **Flows:** Capturing data/material and information in and out the processes. *Examples:* Cabinet Capacity, Cabinet Utilization, New Equipment, etc.
- **Events:** Asynchronously registered situations that trigger the rules. *Example:* Filling up the cabinet up to the max capacity
- **Conditions:** Synchronously occurring situations. *Example:* Sufficient space in the cabinet to mount a new server

B. Object Properties

Formally described, the business rules and workflows can be constructed in terms of object characteristics. The object properties provide information about the characteristics of the objects. For example, the object "Process" may have properties such as: process id, name, status, creation date, etc. From the viewpoint of the conceptualization of our ontology, object properties can be classified into one of the following types:

- **Identification properties** - examples are process id, name, type, etc.
- **Qualitative description properties** - these are categorical or nominal properties, which can be described qualitatively only - for example status, deviation, trend, etc.
- **Quantitative description properties** - these properties can be described using a fixed value that can be estimated quantitatively - for example, the number of closed processes, etc.

In [16], object properties are described as a common approach to specify characteristics or attributes of a real-world object instance, which in turn helps to understand how to interact with the object. By introducing property characterization for each object, our model can fulfil the requirements for flexibility and maintainability of the formulation of business rules to control processes.

C. Business Rules

The structure of business rules is based on the famous Event-Condition-Action paradigm [17]. Various business rule classifications exist in the literature [18]. In connection to BPM systems, the following rule classification were identified:

- **Initiation Rules**

  Initiation Rules depicts rules that specifically initiate a process. Depending on the conditions of the rule, a process can be launched and thus continue execution. Some Initiation rules are driven by events only, these are known as Start Event. The business workflows can be started only by Initiation Rules after a suitable triggering event. The triggering events can be manually or automatically invoked.

- **Event or Process Rules**

  Event or Process Rules group rules that are defined during the execution of a process. An example for such a rule is the filling up of a container which generates a warning about reaching the capacity limit.

- **Flow Rules**

  Flow Rule formally depicts rules that control the flow of processes. Intermediate processes depend on Flow Rules (if this is a specific name then capitalize) to progress from one process to another.

 ISBN: 978-1-61208-614-9

- **Termination Rules**

  The business process terminates based on a Termination Rule, which is triggered by suitable termination event AFTER the process is finished, or on process execution control rule DURING the process execution in the case of emergency. In Fig. 1, Execution Rule was used to check cabinet space availability. The decision to install new network switch onto a cabinet depends on such an execution rule. Some Termination Rules are driven by events only, hence known as End Event.

Objects are building blocks and they are described in this section. Object properties, apart from characterizing the objects quantitatively and qualitatively are also the main vehicle for analyzing the dependencies between the rules which apply to them. The more sophisticated the properties, the more elaborate the dependencies that can be formulated. To allow the mapping and displaying of identified concepts into required classes and properties, a concise and intuitive notation such as EBNF [19] can be used. Although other notations are possible, EBNF is sufficient for the purpose. The term "structured" means that all direct or indirect relations between objects and their properties can be represented into AND/OR trees. The following is EBNF notation for Condition concept, based on the use of objects and their properties.

```
<Condition Object>::=   <Object> <Condition Object> |
                        <Condition Object> AND <Condition Object> |
                        <Condition Object> OR <Condition Object> |
                        <Condition Object> XOR <Condition Object> |
                        NOT <Condition Object> |
                        <Condition Property>
<Condition Property>::= <Object> <Property> |
                        <Condition Property> AND <Condition Property> |
                        <Condition Property> OR <Condition Property> |
                        <Condition Property> XOR <Condition Property> |
                        NOT <Condition Property>
```

V. FORMAL DEFINITION OF BUSINESS RULES, RULE RELATIONSHIPS AND DEPENDENCY TREES

This section briefly presents the formal definition of relationships between rules. The section also exemplifies business rules dependency trees to map rule relationships.

A. Business Rules Formal Definitions

Consider a Business Rule set R containing a collection of rule samples controlling business processes. A Rule set R has one or more related rules that has been put together to guide the movement of processes. For instance, R may be made up of Initiation Rule, Flow Rule, Event or Process Rules and Termination Rule. Let every Rule in R be expressed in terms of $\{R_i,| i= 1,\ldots, n\}$. Each Rule definition R_i consists of a collection of Event (E), Condition (C) and Action (A). We refer to E, C and A to represent sets of Events, Conditions and Actions respectively, containing fragments of the Rule R. Now, let E be expressed in terms of $\{E_i,| i= 1,\ldots, n\}$. And C be expressed in terms of $\{C_i,| i= 1,\ldots, n\}$. Also A be expressed in terms of $\{A_i,| i= 1,\ldots, n\}$. In this research, we will use notation $E_{1i}(R_1)$, $C_{1i}(R_1)$ and $A_{1i}(R_1)$ where $E_{1i} \in E_1$, $C_{1i} \in C_1$ and $A_{1i} \in A_1$ to represent Business Rule basic definition. Note that for simplicity reasons, if a part of the Business Rule has no importance in a discussion then it will be omitted. For example, $C_{1i}(R_1)$ and $A_{1i}(R_1)$ will represent a Business Rule that contains Conditions and Actions only.

B. Relations Between Business Rules

The existence of a dependency between two rules expresses that communication occurs between components (Event, Condition, and Action) of the Business Rule. For example, one Business Rule action may trigger conditions of other Business Rules or condition of one Business rule may depend on an event of another Business Rule. Therefore, Business Rules relationships can be described by analyzing Business Rule components relationships. We consider the relationship between two rules to be represented by the symbol $\xrightarrow{Relate\ to}$. For example, $R_1 \xrightarrow{Relate\ to} R_2$ means Rule 1 relates to Rule 2. If one of R_1 action activates event for R_2, we declare as $A_{1i}(R_1) \xrightarrow{Relate\ to} E_{2j}(R_2)$. Business Rules relationships can be analysed and declared in one of the following possible six ways:

i. $E_{1i}(R_1) \xrightarrow{Relate\ to} E_{2j}(R_2)$
ii. $E_{1i}(R_1) \xrightarrow{Relate\ to} C_{2j}(R_2)$
iii. $E_{1i}(R_1) \xrightarrow{Relate\ to} A_{2j}(R_2)$
iv. $C_{1i}(R_1) \xrightarrow{Relate\ to} C_{2j}(R_2)$
v. $C_{1i}(R_1) \xrightarrow{Relate\ to} A_{2j}(R_2)$
vi. $A_{1i}(R_1) \xrightarrow{Relate\ to} A_{2j}(R_2)$

These relationships are defined based on Objects and Object properties involved in Condition, Event and Action components of the Rules. Moreover, relationship can be defined in terms of qualitative and quantitative characteristics of the object parameters. We examined six ways (i-vi) of representing rule relationships based on the partial order relationship. However, it is far simpler and natural, to apply the tree structure to the model and picture the relationships between rules. Therefore, tree structure and patterns to show relationship are introduced in the next section. Rule patterns are simple enough to represent number of rule relationships. However, in practice there can be hundreds, thousands or more rule relationships. In systems with substantial number of rule relationships, three or more rule dimensions are needed to clearly depict the relationship structure. This is one of the areas that need to be explored in future studies.

C. Business Rules Dependency Tree

In our approach the rule dependencies are defined after structuring them into dependency trees, which are in the form of AND-OR graphs corresponding to the mutual co-existence of the rules. As the name suggests, the relationships will be of two kinds: AND relationships, which

 ISBN: 978-1-61208-614-9

group several rules that can be fired simultaneously, and OR relationships, which group several rules that can be invoked alternatively. There are variants of AND/OR relationships: Direct AND Dependency, Direct OR Dependency, Indirect AND Dependency and Indirect OR Dependency.

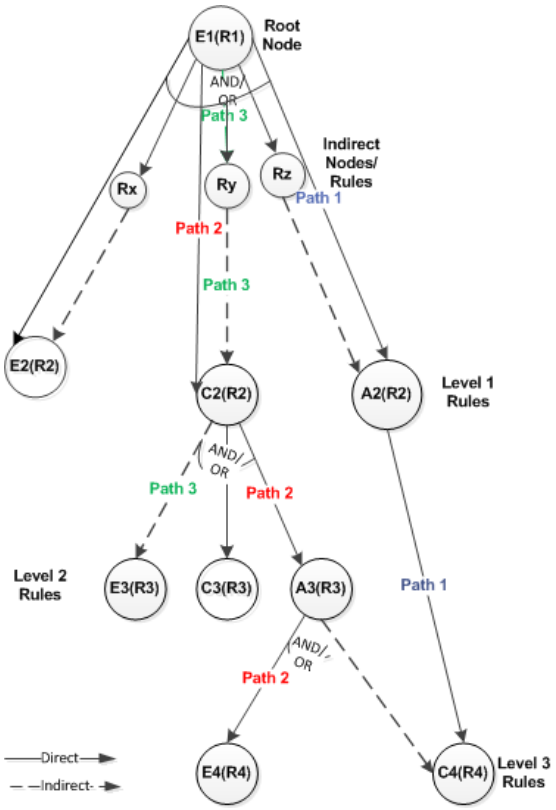

Figure 2. AND/OR Tree.

The AND/OR tree on Fig. 2 combines all relationship patterns:

- Precedence based dependencies
- Level based dependencies
- Path (Chain) based dependencies
- Node based dependencies
- Indirect node based dependencies

The dependency trees make it easier to understand the relationship between rules. The dependencies will be used in construction of the algorithm for real-time inference within BPM system. Structuring of the rules into dependency trees would also allow implementing of more efficient algorithms for searching the rules. Different patterns of inclusion of the rules in the trees will provide additional information to control the flow of execution as the business processes progress. In addition, we can use the trees to analyse the process behaviour in real time.

VI. PROTOTYPE IMPLEMENTATION

The implementation of the model presented here is currently underway using the open source Rule Management System DROOLS [20]. Since it is still work-in-progress, only preliminary developments are presented. DROOLS rule system comes from the area of knowledge representation. The knowledge representation arena is concerned with formally representation of knowledgebase and reasoning. In DROOLS, a rule has two-parts represented using first order logic. The structure of a rule is usually WHEN-THEN that is IF-THEN providing logic statements. This means we can infer conclusions from rule facts stored in the knowledgebase. DROOLS rule system is also perfect for rule adaptation and forward chaining. In DROOLS, the implementation of business rules is carried out using three main components: firstly, the *rule* class (drl) containing the actual rules, second, the *fact* class (pojo) containing the data affected by the rule, and third, the *tester* class (main), which calls both data and rules for execution.

To manage the rules and processes, our architecture implements Event, Condition, Action, Process and Flow (Information and Material) as separate fact classes. In addition, Initiation Rule, Event Rule, Flow Rule and Termination Rule are implemented as subclasses of the rule class which is instantiated in the main or tester class to allow runtime modification of the rules. This supports the reusability and allows adaptation of the rules and their components in the case of changes, as well as the definition of meta-rules using information associated with rule relationships. Fig. 3 illustrates the implementation of the `Condition` class using DROOLS.

```
//Condition class (Fact class)
package com.ABRIW.model;
public class Condition {

	private String Condition_Object;
	private String Condition_Property;
	private String AND;
	private String OR;

	public String getCondition_Object() {
		return ConditionObject;
	}
	public void setCondition_Object(String ConditionObject) {
		this.ConditionObject = ConditionObject;
	}
	public String getCondition_Property() {
		return ConditionProperty;
	}
	public void setCondition_Property(String ConditionProperty) {
		this.ConditionProperty = ConditionProperty;
	}
	public void setAndCondition(String ConditionObject, String AND, String ConditionObject) {
		this.ConditionObject = ConditionObject;
	}
	public void setORCondition(String ConditionObject, String OR, String ConditionObject) {
		this.ConditionObject = ConditionObject;
	}
	public void setAndCondition(String ConditionProperty, String AND, String ConditionProperty) {
		this.ConditionProperty = ConditionProperty;
	}
	public void setORCondition(String ConditionProperty, String OR, String ConditionProperty) {
		this.ConditionProperty = ConditionProperty;
	}

}
```

Figure 3. Condition class using our approach in DROOLS.

`Condition` class defines object related configurations such as object properties and methods. Its methods include `getConditionProperty, setConditionProperty` etc. as shown in Fig. 3. In a `Rule` class, we simply create an instance of a `Condition` class. This automatically inherits the default properties and methods of the `Condition` class. The other concepts of the ontology are implemented in a similar way.

VII. CONCLUSION AND FUTURE WORK

The paper presented a two-level architecture of BPM system, which supports efficient solution for adaptation of business rules, thanks to the incremental indexing of the rules and the formalisation of structural patterns of dependencies between them. This architecture supports BPM professionals and academics with adequate means for modelling of both business process workflows and business rules. In addition, it is the basis for a seamless integration of an efficient algorithm for adaptation of the business workflows to the changing conditions.

A prototype of the above model is being implemented in DROOLS using object-oriented (OO) technology. In this approach both the business workflow processes and business rule components are implemented as objects. Two of the fundamental features of OO technology, the encapsulation and the inheritance, are used conveniently for implementing the architecture following a bottom-up strategy. This approach allows the build up of the indexing mechanism in an incremental manner. The plan, on the next stage, is to complete the implementation of two separate inference engines on top of the model: a forward chaining inference algorithm, which account the logics of business process workflows and controls their execution, and a backward chaining inference engine, which propagates the changes and adapts the rules in real-time. Work has already begun on a series of algorithms, which account for the relationships and the dependencies between the rules. Our focus here will be in exploring the structural patterns of the rule relationships and the influence on the inference on Rule level.

The architecture presented here has wide potential for applying BPM systems in many areas, such as manufacturing, chemical process control, healthcare and anywhere, where the business processes can be described in terms of operational workflows. The big advantage of this architecture is the ability to modify the business rules logics without interrupting the business workflows. Moreover, by adding some meta-rules it could become possible to test the production rules and achieve consistency.

Other issues, which may be beneficial to explore further involve the relationships between different components of the model, i.e., relationships between rules and user roles, relationships between processes and business data, relationships between processes and workflows, etc.

ACKNOWLEDGMENT

The work reported here has been partially sponsored by Vertiv Co, formerly Emerson Network Power.

REFERENCES

[1] K. Ezekiel, V. Vassilev, and K. Ouazzane, “Adaptive Business Rules Framework for Workflow Management”, to appear.

[2] S. D. Hendrick, K. E. Hendrick, Business Value of Business Rules Management Systems, IDC #231195, 2012.

[3] A. Macdonald, *The value of IBM WebSphere ILOG BRMS*, 2010, IBM. [Online]. Available from: https://www01.ibm.com /software/integration/business-rule-management/jrules-family/ retrieved: 12.2017.

[4] J. Boyer, H. Mili, Agile Business Rule Development: Process, Architecture, and JRules Examples, Springer Science and Business Media 1, 2011, ISBN: 9783642190407.

[5] P. Haley, *Confessions of a production rule vendor*, 2013. [Online]. Available from: http://haleyai.com/wordpress/ 2013/ 06/22/confessions-of-a-production-rule-vendor-part-1/ retrieved: 12.2017.

[6] J. C. Giarratano, *CLIPS User’s Guide*, 2003. [Online]. Available from:http://www.ghg.net/clips/download/documen-tation /usrguide.pdf; retrieved: 12.2017.

[7] J. Giarratano, G. Riley, Expert Systems: Principles and Programming, Course Technology, 2004, ISBN: 0534384471.

[8] E. Friedman-Hill, Jess in Action: Java Rule-Based Systems, Manning Publications, 2003, ISBN: 1930110898.

[9] A. Grissa-Touzi, A, H. Ounally, and A. Boulila, “VISUAL JESS: An expandable visual generator of oriented object expert systems”, Engineering and Technology, pp. 108–111, 2005.

[10] R. Thirumalainambi, “Pitfalls of JESS for Dynamic Systems”, Art. Intelligence and Pattern Recognition, pp. 491-494, 2007.

[11] J. Feldman, *Creating, Testing, and Executing Decision Models with OpenRules*, 2011. [Online]. Available from: http://slideplayer.com; retrieved: 12.2017.

[12] M. Salatino, M. De Maio, and E. Aliverti, Mastering JBoss Drools, Packt Publishing, 2016, ISBN**:** 1783288620.

[13] Lijun, Introducing a rule-based architecture for workflow systems in retail supply chain management. MSc Thesis, University of Borås School of Business and IT, Sweden, 2012.

[14] F. Rosenberg, C. Nagl, and S. Dustdar, “Applying Distributed Business Rules – The VIDRE Approach”, IEEE Int. Conf. on Services in Computing, Chicago, IL, USA, pp. 471–478, 2006.

[15] M. Thirumaran et al., “Business rule management framework for enterprise web services”, Int.J. Web Service Computing, Vol.1, No.2, pp. 15-29, 2010.

[16] Y. Sun, B. Liefeng, and D. Fox, “Learning to Identify New Objects”, IEEE Int. Conf. on Robotics and Automation, Hong Kong, pp. 3165 – 3172, 2014.

[17] F. Bry, M. Eckert, P.-L. Patranjan, and I. Romanenko, “Realizing Business Processes with ECA Rules: Benefits, Challenges, Limits”, Int. Workshop on Principles and Practice of Semantic Web Reasoning, Springer, pp. 48-62, 2006.

[18] P. Jayaweera, M. Petit, “Classifying Business Rules to Guide the Systematic Alignment of a Business Value Model to Business Motivation”, Proc. Int. Workshop on Business/IT Alignment and Interoperability, Collection CEUR, Amsterdam, Vol, 456, 2009.

[19] International Organization for Standardization, Information technology. Syntactic metalanguage. Extended BNF, ISO/IEC 14977:1996, pp. 1-22, 1996.

 ISBN: 978-1-61208-614-9