Detecting System Use Cases and Validations from Documents

Smita Ghaisas, Manish Motwani, Preethu Rose Anish
Tata Research, Development and Design Center,
54 Hadapsar Industrial Estates, Pune, 411013,
India
{smita.ghaisas, manish.motwani, preethu.rose}@tcs.com

***Abstract*—Identifying system use cases and corresponding validations involves analyzing large requirement documents to understand the descriptions of business processes, rules and policies. This consumes a significant amount of effort and time. We discuss an approach to automate the detection of system use cases and corresponding validations from documents. We have devised a representation that allows for capturing the essence of rule statements as a composition of atomic 'Rule intents' and key phrases associated with the intents. Rule intents that co-occur frequently constitute 'Rule acts' analogous to the Speech acts in Linguistics. Our approach is based on NLP techniques designed around this Rule Model. We employ syntactic and semantic NL analyses around the model to identify and classify rules and annotate them with Rule acts. We map the Rule acts to business process steps and highlight the combinations as potential system use cases and validations for human supervision.**

Index Terms*—Requirement documents, System use cases, validations, NL analyses, Rules, Rule types, Rule acts, Rule intents.

I. Introduction

In typical distributed development scenarios, requirement documents are written in consultation with the clients and are handed over to offshore teams of analysts and developers. Oftentimes, outsourcing is a multi-vendor process in which Software Engineering activities are distributed; - requirements are prepared by one vendor and implemented by another. The documents need to be studied and analyzed before the implementation can begin. Large IT vendors generate an enormous amount of documented knowledge in a given business domain as a result of executing thousands of projects over the years. Project managers always hope to reuse this knowledge in the next new project in the same domain, but their teams are swamped by the task of having to read the available documents to identify useful pieces of knowledge. In large projects spanning over a few years, teams keep changing and new developers take over from their old counterparts. The task of having to read and infer from the project documents is a daunting one for the newcomers.

The scenarios highlight the need for automating the process of detecting and extracting knowledge elements from existing documents.

In this paper, we focus on two of the crucial knowledge elements – (1) System use cases based on business process step descriptions, and (2) Validations corresponding to business rules, and policies. System use cases take into account the design scope so that users can achieve their goals using the system [1]. A comprehensive set of system use cases therefore enhances the chances of meeting users' requirements. Identifying system use cases and corresponding validations involves analyzing large requirement documents to understand the descriptions of business processes, rules and policies of a client's organization. This activity consumes a significant amount of effort and time. We discuss an approach to automate the detection of system use cases and corresponding validations from documents.

For this purpose, we studied a sample from 500 requirement documents from 20 large projects aimed at developing Insurance systems. The documents contained descriptions of business processes, rules and policies of clients. We found that (1) the rule statements are oftentimes embedded in process descriptions, and (2) the rule statements that practitioners write are typically compound and complex sentences. Each statement contains several (intended) constraints which we term 'Rule intents' [2]. Based on our analysis of the documents, we arrived at a representation that allows for capturing the essence of rule statements in terms of the atomic Rule intents.

Each rule statement can be represented as a composition of Rule intents. We store the Rule intents and associated key phrases in a database. We then run an agglomerative clustering algorithm [3] over the rule statements to detect the Rule intents that co-occur most frequently. The groups of frequently co-occurring intents are manually inspected to name the 'Rule acts.' Analogous to the Speech acts [4, 5] in Linguistics that express apology, gratitude, and request, the Rule acts explicate system validations such as access control, data validation, and conditional execution. The Rule acts we identified from the document sample are sub-classes of the rule types that Ross defines [6, 12].

The requirement documents are written by practitioners and are intended for developers who implement the system. Therefore the rule statements in the documents are far more fine-grained than what can be described in terms of Ross's rule types. Hence the need to define Rule acts as sub-classes of the types defined in [6, 12]. Our approach is based on NLP techniques designed around this Rule Model. We provide a framework for detecting system use cases and corresponding validations as follows: We employ syntactic and semantic NL

analyses around the Rule Model to (1) identify and classify rules, (2) annotate them with Rule acts, and (3) recommend their possible mappings to business process steps. The process steps together with associated Rule acts make for a set of system use cases and validations to be implemented by developers.

The rest of the paper is organized as follows: In section II, we discuss the details of the Rule Model. Section III presents details of the method for an automated extraction and classification of rules. In Section IV, we discuss our early results on detecting system use cases and validations. Section V is a review of related work and Section VI presents conclusions and future work.

II. Rule Model

In this section, we define the key concepts and describe the method of training the dataset for detecting system use cases and validations.

We selected 20 documents from a population of 500 documents using the stratified sampling technique [7]. The documents are from 20 large projects in the Insurance domain. We performed a manual content analysis on the stratified sample. We had four coders who studied the requirement documents, each analyzing 5 documents. We use the following definitions in the description of our manual content analysis and in the discussions subsequently.

A. Definitions

Rule intent is defined as an atomic constraint embedded in a natural language rule statement which is usually a compound sentence. For example : A rule such as "*In Europe, you need to be above 18 to get a driving license*" has 3 Rule intents – (1) Geographical boundary – *In Europe*, (2) Threshold – *need to be above 18* and (3) Activity – *get a driving license*. We observe from our manual analysis of documents that any rule statement is composed of one or more Rule intents.

Rule intent patterns are syntactic representations of phrases and keywords associated with Rule intents in a rule statement. The Rule intent pattern is a combination of Parts of Speech (POS) tags [8], keywords and wild card characters ('*', '+', '.') where '*' represents a frequency of 0 or more occurrences of words, '+' represents a frequency of 1 or more occurrences of words and '.' represents exactly 1 occurrence of a word.

For example, consider the following sentence:

For the Family plan, the loss-claim should be submitted within 90 days from the date of the incident.

The sentence is decomposed into the following four parts based on the Rule intents it contains. Using the keywords and POS tags, Rule intent patterns are then created for each part. These patterns are stored in a database along with the corresponding Rule intents.

1. For/IN the/DT Family/NNP plan/NN, (intent: Information Type; pattern: + NNP * plan *)
2. ,/,the/DT loss-claim/NN should/MD be/VB submitted/VBN (intent: Activity; pattern: * claim + MD VB + VBN *)
3. within/IN 90/CD days (intent: Temporal Check; pattern: * within CD days *)
4. from/IN the/DT date/NN of/IN the/DT incident/NN (intent: Threshold; pattern: * from + date of +)

Rule acts are composed of frequently co-occurring Rule intents. As explained earlier, they explicate validations such as access control, data validation, and conditional execution which upon implementation, will conform to the corresponding relevant rules. The Rule acts are sub-classes of the rule types that Ross defines [12].

Fig.1 shows the Rule Model elements and their associations using the UML class diagram notation. We use this model to extract, classify and annotate rules with rule acts and map them with process step descriptions that contain the key phrases in the rule intent patterns. The process steps together with associated rule acts make for a set of system use cases and validations to be implemented. We describe the method of detection in Section III.

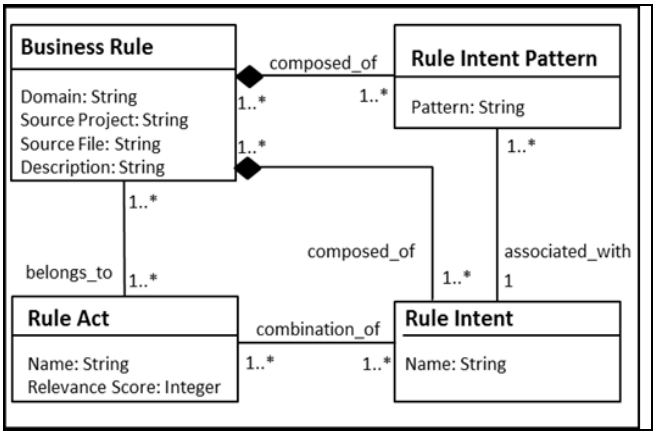

Fig.1 Rule elements and their associations

B. Manual Content Analysis

In order to facilitate reconciliation of analysis by 4 different coders, we prepared a coding guide based on our pilot iteration. The guide contains criteria for (1) identifying a rule statement using the basic definition [6, 12], and (2) identifying the Rule intents and patterns according to the definitions above. The sets are then exchanged by the coders and a second iteration is performed. They may also identify additional statements as business rules if they can rationalize their decision based on some criteria not yet present in the coding guide. At the end of the second iteration, we reconcile the results and refine the coding guide. This process is repeated two more times. Thus, every coder analyzes each document. At the end of the four iterations, we obtain a richer set of criteria to identify business rules and Rule intents. Towards the later iterations, the criteria that each coder identifies become repetitive and the process of identification 'saturates'. Table I illustrates examples of rule statements, Rule intents they contain, and corresponding Rule intent patterns. In all we discovered 29 intents and 517 Rule intent patterns. The details of the manual analysis will be published separately.

Unlike the observations in [23] and [24], the documents we studied did not have a uniform structure. They did not necessarily adhere to any specific template either. Viewing rule statements in terms of fine-grained Rule intents and associated patterns was motivated by this lack of structure.

TABLE I. RULE STATEMENTS, RULE INTENTS AND RULE INTENT PATTERNS

Rule Statement	Rule Intents	Rule Intent Patterns
During a call to the Service Router, the application in context will be locked to the active user.	**Chronology** – *During a call*	During *
	Event– *call to the Service Router*	* call TO + NN *
	System state – *application in context will be locked*	*application + MD VB VBN *
	System User – *to the active user*	+ TO * user *
If the customer type is policyholder, the following two additional fields are displayed in call outcome section: Interaction Attempted, Customer Interaction	**Information Type** –*If customer type is policyholder*	If + type * NNP *
	Cardinality – *two additional fields*	* CD * fields *
	Screen Element State – *fields are displayed in call outcome section*	* fields + displayed IN + NN *

III. AUTOMATED EXTRACTION AND CLASSIFICATION OF RULES

This section describes a method to automate the classification of business rules from project documents. The Rule acts used for classification are derived automatically from Rule intents. The clusters of frequently co-occurring Rule intents are termed Rule acts. We have named them as per the system validations they explicate.

A. Identifying Rule statements

During the rule identification phase, the *Rule Extractor* evaluates each sentence to determine if it is a rule by comparing the words in the training dataset with standard key phrases such as *should not, may not, until, will be, within, up to, and ensure* etc. It additionally verifies the sentence against Rule intent patterns from the database. If the statement contains at least one pattern, it is detected as a rule.

For example, the *Rule Extractor* will identify the sentence - *All confirmed acceptance must be done within 24 hours of receiving the claim notification* as a 'rule' because the given sentence matches against key phrases – *must* and *within*. Further, the sentence contains the Rule intent patterns - **** MD VB VBN **** and **** within CD hours **** that corresponds to Rule intents – ***activity*** and ***temporal check*** respectively.

B. Determining Rule acts

The *Rule Extractor* extracts rule statements from the training dataset and stores the output in a database. The *Rule Act Identifier* then runs an agglomerative clustering algorithm [3] over the rule statements to identify Rule intents that co-occur most frequently. These groups/clusters are manually analyzed to name the Rule acts. The mechanism to come up with Rule acts is unsupervised learning technique where we use Rule intents (labeled data) to form clusters and label these clusters as Rule acts (unlabeled data). The identified Rule acts are stored in a database. Table II shows some examples.

TABLE II. RULE ACTS OBTAINED FROM CLUSTERING RULE INTENTS

Rule Act : Definition	Rule Intents	Example Rule
Deadline: Rules that restrict time duration date.	• **temporal check** • **threshold** • **activity**	*For the Family plan, the loss-claim should be submitted within 90 days from the date of the incident.*
Conditional Execution: Rules describing checks to be performed by user or system while executing process steps	• **chronology** • **conditional flow** • **activity**	*Likewise, there is a suicide clause in the policy which is operative for one year. If reason for death is suicide and that occurred within one year after the date of proposal, then nothing is payable.*
Access Control: Rules that restrict stakeholder's access or constraints activities performed by them	• **stakeholder type** • **information type**	*The system only allows the Claim Officer to modify the Interim Claim payment amounts. The Claim Officer will not be allowed to alter anything else for the Claim in question.*
Policy Administration: Rules related to administration of insurance policy that involves activities such as policy issuance, renewal etc.	• **conditional flow** • **policy state** • **policy process** • **temporal check**	*It is helpful to confirm to the person that the policy proceeds do not form part of the estate for Inheritance Tax purposes. But there may still be some Inheritance Tax liability on the premiums paid if premiums have been paid within 7 years of death*
User Interface (UI): Rules related to user interface describing various screen elements, screen layout etc.	• **screen element state** • **screen representation** • **conditional flow**	*Failing the caller verification process, no more policy details can be retrieved from the system as the Policy details, Interaction history and associated plan buttons will remain disabled.*
Data Validation: Rules that validate data processed by the system or used by stakeholders to perform some task	• **Numerical value** • **information type** • **conditional flow**	*Regular withdrawals will cease when the amount of withdrawal exceeds half the plan value, eg: Plan value = Rs. 2000 Withdrawal = Rs. 1050 not allowed. Only a withdrawal of Rs. 1000 would be allowed in this case.*

The Rule acts differ in granularity from the rule types defined by Ross [12] as follows: The rule statement *If the claim payout is less than $ x, a person holding designation of y is authorized to waive requirement provided it is found in order,* would be

classified as **Derivation Rule (Inference Rule)** according to Ross's definition [12]. The rationale for this classification is that it infers (a derived term or fact) automatically based on the conditions specified explicitly. Using our approach, the sentence gets classified into two *Rule acts* - **Data Validation** (Intents: *Numerical Value, Conditional flow, Information Type*) and **Access Control** (Intents: *Stakeholder Type, Information Type*). The Rule acts thus explicate implementation specifics. They are therefore easier for the practitioners to relate to. A Rule act such as ***Deadline*** clearly indicates what the system should check and the corresponding Rule intents ***temporal check, activity*** and ***threshold*** indicate what parameters must be incorporated in the validations. When mapped to corresponding process step description statements, they together serve as a set of system use cases and system validations. We discuss the method of classification in Section IV.

IV. Identifying System Use cases and Validations from the Test Dataset

Our test dataset consists of 30 requirement documents from large projects in the Insurance domain. We use the *Rule Extractor* to extract the business rule statements. The rule statements are separately stored in a rule repository. The *Rule Classifier* takes these rule statements as an input and classifies them into Rule acts based on Rule intents they contain. The relevance score of a Rule act for a given rule is calculated comparing the number of Rule intents present in the rule statement with those that constitute the Rule act. This information is also stored in rule repository for all rule statements extracted from the test dataset. The rule repository contains information about source projects and source files as well.

Next, the *Rule Annotator* annotates the rule statements present in the documents. The annotator uses the information about the source project and the source file of a rule statement present in rule repository to locate it in the documents. After locating the rule, it highlights the rule statement and annotates it with its Rule acts, and Rule intents. It also highlights the parts of business process step descriptions that contain the key phrases in Rule intent patterns. A given rule statement may map to one or more process step descriptions and vice versa. The rule statements annotated with Rule acts and intents along with corresponding process step descriptions are then subject to human inspection for selecting system use cases and corresponding validations. The annotations make it easier for a reader to locate the important text without going through each sentence of the document. As discussed earlier, Rule intents that constitute a given Rule act clearly indicate what kinds of validations need to be implemented in the system. When mapped to corresponding process step descriptions, they serve as a set of system use cases and validations to be implemented. Table III shows an example.

From the test dataset we could extract a total of 881 rule statements. Of these, 738 were found to be correct. A manual analysis brought out that the number of rules extracted should have been 944. Thus, we have a recall of 78% and a precision of 84% for the extraction.

TABLE III. Rule acts mapped with process step descriptions

Process step: Receive completed switch form
Process step description: The insurance company at the time of issuing the policy along with sending the policy document, schedules sending of switch forms to the policy holder. There could be instances when the policy holder would have already exhausted their existing switch forms and hence may call up the insurance company and request for a switch form. The request can also come from the Fund advisors. On such requests, the insurance company sends the switch forms. The policy holder has to fill in all the particulars of the switch form and send it back to the insurance company. The policy holder may request for the entire spread to be switched to the targeted fund or may seek only a percentage of the fund to be switched.
Rule: Check if the correct policy information details, details of switch from one fund to another are available in the notification received and that the switch is properly authorized. It must be ensured that the date of receipt is marked on the switch request.
Rule intents: document validation, authorization check, information type, activity
Rule acts: Data validation, Conditional Execution

Table IV represents the distribution of business rules into Rule acts after classifying them using the method discussed earlier in this section.

TABLE IV. Rule classification

Rule acts	Distribution of Rules	Rules classified correctly	Precision (%)
Deadline	104	75	72.11
Conditional execution	192	117	60.94
Access Control	177	93	52.54
Policy Administration	88	72	81.82
User Interface (UI)	61	56	91.80
Data Validation	136	126	92.65

The average precision for classification is 71.10%. We find that the classification precision for Rule acts such as *User interface (UI)* and *Data validation* is quite high while that for *Access control* and *Conditional Execution* is low. We are working on refining the existing Rule intent patterns and identifying new ones using synonyms to improve the precision.

The Rule acts are then mapped to business process step descriptions. Table V shows the results of mappings between the Rule acts and process step descriptions. The average precision is 72.22% and average recall is 78.43%. As discussed earlier, these serve as recommendations to identify system use cases and validations.

Key to reading TABLE V:
A: Number of Business process step descriptions (Automated mapping)
B: Number of Business process step descriptions selected after human inspection of automated mapping
C: Number of Business process step descriptions that should have been mapped (Manual mapping)

TABLE V. BUSINESS PROCESS STEPS CORRESPONDING TO RULE ACTS

Rule acts (Number of rules classified correctly)	A	B	C	Precision (%)	Recall (%)
Deadline (75)	64	52	56	81.25	92.86
Conditional execution (117)	176	152	153	86.36	99.35
Access Control (93)	120	104	160	86.67	65.00
Policy Administration (72)	116	96	98	82.76	97.96
User Interface (UI) (56)	96	36	108	37.50	33.33
Data Validation (126)	148	80	88	54.05	90.91

The recall and precision values for the Rule act *User Interface (UI)* is unacceptably low. Our analysis shows that the process descriptions corresponding to this Rule act are written ambiguously. For example, a manual analysis of a statement such as '*System sends a communication to the FA with the details of number of units for each fund before the pre-defined cut off time'* suggests that this may require implementing a UI component through which the FA can see the details of number of units. However, this could also mean that an e-mail is sent to the FA and a UI may not be required. We are investigating such patterns to arrive at a set of key phrases that would help in a better detection. The reasons for low precision for the Rule act *Data Validation* and low recall for the Rule act *Access Control* are also being examined in the light of ambiguities in the documented process descriptions.

The threats to validity of our approach are as follows.

- Currently we have analyzed the documents from the Insurance domain alone. We expect to identify additional Rule intent patterns and Rule acts from other domains. The work reported here does not reflect these.
- The manual content analysis is done on stratified sample of repository (training set). It is possible that some important and interesting Rule intents or Rule intent patterns are not identified and hence left out.
- A rule may get classified into more than one Rule act. The final decision to classify a rule is to be made by human intervention.
- The results are being validated with developers for their usefulness in practice. Their reactions are positive. However, the empirical study is yet to be concluded.

Information represented in images and tables was not extracted; this is a technical limitation of the method.

V. Related Work

In this paper, we employ a method for an automated classification of business rules to detect system use cases and validations. Business rules encapsulate significant amount of domain, system and business environment related knowledge crucial for the system to be developed and used. They capture design rationale, explanations on why a particular algorithm, pattern, or data structure is used in a system [9, 10]. A large body of knowledge has emerged as a result of their significance to software engineering. Most of the works attempt to better understand the business rules by focusing on (1) definition, (2) classification, (3) extraction, and (4) management. Of these, the first three are relevant to be discussed in the context of the technique employed in our work.

Definition. Ellen Gottesdiener [11] defines business rules as declarative, atomic, expressed in natural language, distinct, business oriented and business owned. Ronald Ross [12] defines several dozens of atomic rule types and equates them to elements in the periodic table using which composite rules can be formed. Ceri and Fraternale [13] contend that business rules model the reaction to events, which occur in the real world. According to the Business Rules Group definition [6], a business rule either asserts business structure or controls or influences its behavior. It is also defined as a requirement on condition or manipulation of data [14] or a computational requirement that determines or affects how business is run [15].

Classification. The efforts pertaining to classification take into account various viewpoints. For example, Wieden et al [16] offer 15 different "semantically-oriented" rule types grouped into structural, behavioral and managerial categories, while Zoet M [17] et al propose a business rule categorization that is aligned to the business process management lifecycle. H. Herbst, [18] argue that the commonly used data oriented methods are insufficient and inconvenient for a complete modeling of business rules. To the best of our knowledge, there is no work reported on an automated classification of business rules from project documents. Cleland-Huang et.al. [19] describe a technique for an automated detection and classification of non-functional requirements. This work is closest in spirit to our work.

Extraction. Ali et al. [20] suggest an approach that takes rule repositories either in relational or text format as input and convert it into xml syntax by applying transformation method on SQL queries or a parsing and transformation method using xquery. Mahgoub et al. [21] integrate XML technology with Information Retrieval scheme (TF-IDF) for keyword/feature selection that automatically selects the most discriminative keywords used for association rules generation and use Data Mining technique for association rules discovery. Breaux and Antón [22] propose a method to mine rule semantics to understand legislative text. The method is applied on privacy regulations derived from the Health Insurance Portability and Accountability Act (HIPAA). The work reported in [23] focuses on semantic extraction of Access Control Policies.

To the best of our knowledge, this is first time an automated detection of system use cases and validations is attempted based on classification and annotation of rule statements in a document. The novelty of our approach lies in (1) providing the right granularity for representing rule statements in a Rule Model, and (2) using the Rule Model to detect important knowledge elements in software engineering; namely system use cases and validations. We focus on working with documents which still remains the preferred medium of work of practitioners who write requirements.

VI. Conclusion and Future Work

We discuss an approach to automate the detection of system use cases and corresponding validations from

documents. We employ syntactic and semantic NL analyses around a Rule Model to (1) identify, and classify rules, (2) annotate them with 'Rule acts', and (3) recommend their possible mappings to business process steps.

For the extraction, the precision is 84% while for the classification; ~ 72%. The Rule act to process step mappings recommendations show an average precision of ~ 72 % and a recall of ~ 78%. These are encouraging early results. We note that in the context of complying with rules, false negatives are riskier than false positives. We are exploring technical refinements to improve recall for some of the Rule acts. Nonetheless, the approach addresses an important practical problem by highlighting and annotating the possible system use cases and validations for human inspection and selection. To the best of our knowledge, this is the first ever attempt to automate the classification of rules in documents and use the classification for recommending possible system use cases and validations. Another interesting contribution of this work is an implementation-specific interpretation of Ross's rule types [12] in the form of Rule acts. As an immediate next step, we plan to investigate (1) logical grouping of the extracted rules from the angle of their complementarity, and (2) identifying completeness and ambiguity issue in the extracted rules.

We are currently working on an ontological representation for the Rule intents, Rule acts, and their associations. This is an extension of our previous work on domain knowledge repositories [25, 26]. With such a representation, business rules can be associated with additional domain knowledge elements such as business processes, data models, and system components. This would allow for an easy reference, traceability and reuse in development projects and in product configuration exercises that are highly rule-intensive.

Upon a more comprehensive analysis that covers additional business domains, we aim to create a richer corpus of Rule intents, associated patterns, and Rule acts. We plan to validate the approach in more diverse project situations.

ACKNOWLEDGMENT

We thank the BaNCS Insurance unit of Tata Consultancy Services for many useful discussions and practical insights into rule usage in large projects.

REFERENCES

[1] A. Cockburn, Writing Effective Use cases, *Reminder 13, p216,* Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA ©2000

[2] S. Ghaisas, M. Motwani, P. R. Anish, S. Sharma, Automated classification of Business rules from text, US Patent pending, Application No 13/778850

[3] P. Cimiano, A. Hotho, & S. Staab, "Comparing conceptual, divisive and agglomerative clustering for learning taxonomies from text." Proceedings of the European Conference on Artificial Intelligence (ECAI), pp. 435–439, 2004

[4] F. Ibekwe-SanJuan, S. Fernandez, E. SanJuan, and E. Charton, "Annotation of scientific summaries for information retrieval". Proceedings of the Workshop on Exploiting Semantic Annotations in Information Retrieval (ESAIR), Glasgow, Scotland, 2008

[5] S. Teufel and M. Moens. Sentence extraction and rhetorical classification for flexible abstracts. In D. Radev and E. Hovy, editors, Intelligent Text Summarization, The Association for the Advancement of Artificial Intelligence Spring Symposium, pages 16-25. AAAI Press, Menlo Park, CA, 1998

[6] Business Rules Group; Defining business rules, what are they really?, July 2000, http://www.businessrulesgroup.org/first_paper/BRG-whatisBR_3ed.pdf. Last accessed on August 2013

[7] J. J. Castillo, Stratified Sampling Method. Retrieved March 2012 from Experiment Resources,(2009), Webpage: http://www.experiment-resources.com/stratified-sampling.html

[8] OpenNLP Part-of-Speech (POS) Tags: Penn English Treebank, Webpage: http://blog.dpdearing.com/2011/12/opennlp-part-of-speech-pos-tags-penn-english-treebank/

[9] J. Boyer and H. Mili, Agile business rule development, Springer Verlag Berlin Heidelberg, ISBN: 978-3-642-19040-7, 2011

[10] G. R. Ronald, Business Rule Concepts: Getting to the Point of Knowledge (Third Edition), ISBN 0-941049-07-8, August 2009

[11] E. Gottesdiener, "Business RULES - Show Power, Promise". EBG Consulting, Inc. Application Development Trends, vol. 4 no.3.1997

[12] R. G. Ross, The Business Rule Book: Classifying, Defining and Modeling Rules, Business Rule Solutions Inc; 2nd edition, 1997

[13] S. Ceri, and P Fraternale, Designing Database Applications with Objects and Rules, The IDEA Mehtodology, Addison and Wesley, 1997

[14] P.G. Sellfridge, R.C Waters, E Chikofssky. "Challenges to the field of reverse engineering", Proceedings of Working Conference on Reverse Engineering (WCRE), Baltimore, Maryland, USA, pp. 144-150, 1993

[15] D. Rosca, S. Greenspan, M. Feblowitz, and C .Wild, "A decision making methodology in support of the business rules life cycle", Proceedings of IEEE International Requirements Engineering conference, pp.236-246, 1997

[16] M. Weiden, L. Hermans, G. Schreiber and S. V. D. Zee. "Classification and representation of business rules", Technical Report, University of Amsterdam, 2002

[17] M. Zoet, J. Versendaal, P. Ravesteyn, R. Welke.,"Alignment of business process management and business rules", Proceedings of the European Conference on Information Systems (ECIS), 2011

[18] H. Herbst, G. Knolmayer, T. Myrach and M. Schlesinger. "The specification of business rules: A comparison of selected methodologies", published in: A.A. Verijn-Stuart, T.-W. Olle (Eds.), Methods and Associated Tools for the Information System Life Cycle, Amsterdam et al.: Elsevier, pp. 29-46, 1994

[19] J. C.-Huang, R. Settimi, X. Zou, P.Solc, "Automated classification of non-functional requirements", Proceedings of IEEE International Requirements Engineering conference, vol.12, pp.103-120, May 2007

[20] S. Ali, B. Soh, and J. Lai, "Rule extraction methodology by using XML for business rules documentation," Proceedings of the 3rd International Conference on Industrial Informatics, pp. 357-361, 2005

[21] H. Mahgoub, D. Rosner, N. Ismail, F. Torkey, "A text mining technique using association rules extraction", International Journal of Information and Mathematical Sciences vol. 4, no.1, 2008

[22] T. Breaux, and A.I. Antón, "Mining rule semantics to understand legislative compliance", Proceedings of ACM workshop on Privacy in the electronic society, November 2005, Alexandria, VA, USA

[23] X. Xiao, A. Paradkar, S. Thummalapenta, and T. Xie. "Automated extraction of security policies from natural-language software documents". Proceedings of 20th International Symposium on the Foundations of Software Engineering (FSE), pp. 12:1–12:11, 2012

[24] R. Pandita, X. Xiao, H. Zhong, T. Xie, S. Oney, and A. Paradkar. "Inferring method specifications from natural language API descriptions", Proceedings of 34th International Conference on Software Engineering (ICSE), pp. 815–825, 2012

[25] S. Ghaisas and N. Ajmeri, Knowledge assisted requirements evolution, In Press, Managing requirements Knowledge, (MaRK) Ed., Springer http://www.springer.com/computer/swe/book/978-3-642-34418-3

[26] N. Ajmeri, R. Sejpal, S. Ghaisas,: "A semantic and collaborative platform for agile requirements evolution", Proceedings of 3rd International Workshop on Managing Requirements Knowledge, pp. 32--40. IEEE Press, 2010, Sydney.