2013 IEEE 37th Annual Computer Software and Applications Conference

Ontology-Based Classification of Non-Functional Requirements in Software Specifications: A new Corpus and SVM-Based Classifier

Abderahman Rashwan Olga Ormandjieva René Witte
Department of Computer Science and Software Engineering
Concordia University, Montréal, Canada

***Abstract*—A software requirements specification (SRS) contains all the requirements for a system-to-be. These are typically separated into functional requirements (FR), which describe the features of the system under development, and the non-functional requirements (NFR), which include quality attributes, design constraints, among others. It is well known that NFRs have a large impact on the overall cost and time of the system development process, as they frequently describe cross-cutting concerns. In order to improve software development support, an automated analysis of SRS documents for different NFR types is required. Our work contains two significant contributions towards this goal: (1) A new gold standard corpus containing annotations for different NFR types, based on a requirements ontology; and (2) a Support Vector Machine (SVM) classifier to automatically categorize requirements sentences into different ontology classes. Results obtained from two different SRS corpora demonstrate the effectiveness of our approach.**

***Keywords*-Software Requirements Engineering; Requirements Ontology; Requirements Corpus Development; SVM Classifier**

I. Introduction

Requirements Engineering (RE) is one of the most important phases of a software project: We know that the success or failure is highly dependent on successful requirements engineering, with industry statistics pointing to insufficient RE as the root cause in more than 50% of all unsuccessful software projects [1].

When an initial set of requirements has been elicited and evaluated, it can be captured in a requirements document. Natural language requirements specifications are the most commonly used form (as opposed to formal models, based on a logical framework), accounting for up to 90% of all specifications [2]. However, natural language requirements specifications are prone to a number of errors and flaws, in particular due to the ambiguity inherent in natural language. Moreover, there is a lack of available methods and tools to aid software engineers in managing requirements. As the requirements are written in informal natural language, they cannot be easily analyzed for defects, such as inconsistency or ambiguity. Our approach to overcome these challenges is based on natural language processing (NLP), machine learning techniques, and ontologies.

Requirements documents typically differentiate two types of requirements: functional requirements (FR), which define the features of the system-to-be, and non-functional requirements (NFR), which define the quality attributes of the system features (e.g., performance, reliability, usability) [1]. Recent studies showed that designers and developers often focus more on the behaviour of the system (FRs) and underestimate the development time and cost for the NFRs [3]. This typically leads to cost and time overruns, and can ultimately result in project failures. Hence, the detection and classification of NFRs has recently received more attention in RE, and is therefore the goal of our work described here.

Most of the terms and concepts in use for describing NFRs have been loosely defined, and often there is no commonly accepted term for a general concept [4]. In [5], the authors present a decomposition and operationalization of NFRs into types, managing them by refining and inter-relating NFRs, justifying decisions, and determining their impact, as elaborated in the NFR framework [5]. Decomposition refines NFRs into more detailed NFRs, while operationalization results in strategies for achieving the NFRs, such as *prototyping* for "usability" NFRs. An example of a hierarchical NFR decomposition of quality characteristics into sub-characteristics is the ISO/IEC 9126 international standard [6]. NFR refinement is often enhanced with domain-specific knowledge, as in [7], where the authors introduce knowledge and rules provided by a domain ontology to induce non-functional requirements in specified domains. Al Balushi and Dabhi [8] also use an ontology-based approach to requirements elicitation, aimed at empowering requirements analysts with a knowledge repository that helps in the process of capturing precise non-functional requirements during elicitation interviews. The approach is based on the application of functional and non-functional domain ontologies (quality ontologies) to underpin the elicitation activities. In contrast, our work aims at providing a more generic solution to model and classify several types of NFRs, independent from their context.

In this work, we present two important contributions: *(i)* a novel manually annotated gold standard corpus, containing documents belonging to three different requirement artifacts (four Software Requirements Specifications (SRS) documents, one supplementary specification document, and one use case document), with 3064 annotated sentences in total; and *(ii)* a novel classifier based on Support Vector Machines (SVMs) that outperforms existing approaches on the requirements classification task.

TABLE I
PROMISE corpus: NFR classes and number of sentences for each class

Class	A	LF	L	M	O	P	SC	SE	US	Total
# sent.	18	35	10	16	61	48	18	58	62	326

The remainder of this paper is structured as follows: Section II describes the background. In Section III, we detail our new, manually annotated NFR gold standard corpus. Section IV describes the design and implementation of our SVM-based machine learning NFR classifier. Section V presents the evaluation results and compares the results of our proposed classifier with similar work. Related work is described in Section VI. Finally, some concluding remarks are stated in Section VII.

II. Background

In this section, we briefly describe the background for our work.

A. Requirements Specifications and NFRs

Software Requirements Specification (SRS) documents are important artifacts in the software industry. A SRS contains all the requirements specifications for a software system, typically separated into functional requirements (FR) and non-functional requirements (NFR) [1]. NFRs usually impact the system as a whole and interact both with each other and with the functional requirements. SRS documents are typically written in informal natural language [2], which impedes their automated analysis. The goal of this work is to support software engineers with semantic analysis methods that can automatically extract and analyze requirements written in natural language texts, in order to *(i)* make SRS documents machine-processable by transforming them into an ontological representation; *(ii)* apply quality assurance (QA) methods on the extracted requirements, in order to detect defects, like ambiguities or omissions; and *(iii)* attempt to build traceability links between NFRs and the FRs impacted by them, in order to aid effort estimation models.

B. PROMISE Corpus

The PROMISE corpus [9] consists of 15 SRS documents, developed as term projects by MSc students at DePaul University. These specifications contain a total of 326 non-functional requirements and 358 functional requirements. The NFR types include availability (A), look-and-feel (LF), legal (L), maintainability (M), operational (O), performance (P), scalability (SC), security (SE), and usability (US). Table I presents the number of sentences for each NFR class.

III. NFR Corpus and Ontology Development

While evaluating the PROMISE corpus, we realized that *(i)* it does not cover all requirements artifact types, such as vision documents, use case descriptions, supplementary specifications, as well as information included in e-mails or minutes of meetings; *(ii)* Requirement sentences in these documents can not only be of a single requirements type, but also contain multiple classes, or no class at all; and *(iii)* SRS and other requirement documents are often written at very different levels of abstraction and with different formal writing styles, which impacts automatic requirements classification.

TABLE II
Documents Source

S	Doc	Type	Year	Source	# Sent.
1	Online shopping centre	SRS	2009	Indian Institute of Information Technology	107
2	Student Management System	SRS	2005	University of Portsmouth	174
3	Hospital Patient	SRS	2002	University of Calgary	259
4	Swedish Institute of Space Physics	SRS	2001	Swedish Institute of Space Physics	237
5	Supplementary Specs	Suppl. spec.	2009	Concordia University	255
6	Use case model SOEN 390	Use case	2007	Concordia University	2032
Total					3064

In order to be able to develop and evaluate automated requirements analysis tools, we needed a gold standard that provides fine-grained annotations of requirements, with an ontological classification of different NFR types. Since no such corpus existed, we developed a new corpus, based on a requirements ontology.

A. Requirements Ontology

In our approach, NFRs are classified based on a requirements ontology, modeled using the Web Ontology Language (OWL) [10]. The conceptualization is a simplified version of the one developed in [3], as shown in Fig. 1, by limiting it to the major classes and concepts that frequently appear in requirements documents. Most of the concept definitions are based on the ISO 9126 standard [6]; with some additional sources for further refinements, as indicated in Table III. This ontology contains several views: *(i)* The intra-model dependency view, which represents the associations with other functional requirements and resources in SRS; *(ii)* the inter-model dependency view, which is concerned with the different types of NFRs and divided into sub-categories; and *(iii)* the NFR measurement view, which represents a measurement model for NFR fit criteria. Table III show the class definitions, together with examples from our Concordia RE corpus, of the concepts selected for automatic ontology classification.

B. Manual Annotation

Our manual SRS corpus annotation process was implemented based on GATE Teamware,[1] a web-based platform for managing collaborative annotation. The annotation task was carried out by four annotators in order to guarantee the reliability of the corpus. The first step was to pre-process each document,

[1]GATE Teamware, http://gate.ac.uk/teamware/

TABLE III
NFR ONTOLOGY: MAJOR CLASS DEFINITIONS

Class	Definition	Example
Constraint	Constraints are defined in [11] as restrictions on the design of the system, or the process by which a system is developed, that do not affect the external behavior of the system but that must be fulfilled to meet technical, business, or contractual obligations.	"The system's design will rely heavily on existing patterns and models for the organization of system components."
Usability/ Utility	The ease with which a user can learn to operate, prepare inputs for, and interpret outputs of a system or component (External and internal quality [6], Utility [12]).	"The GUI under the rehearsal session should be designed to help students to prepare exams."
Security	A measure of the system's ability to resist unauthorized attempts at usage and denial of service while still providing its services to legitimate users (Functionality quality requirement [6]).	"The system shall allow system administrators to manage the users by creating, editing, or deleting users."
Efficiency	The amount of computing resources and code required by a program to perform its function [6].	"The system should be able handle the concurrent access of the maximum capacity of an exam room during an exam session."
Functionality	A set of attributes that bear on the existence of a set of functions and their specified properties (External and internal quality [6]).	"The system's intrinsic characteristic of being designed for change will allow it to easily integrate with any required third-party components."
Reliability	The ability of a system or component to perform its required functions under stated conditions for a specified period of time (External and internal quality [6]).	"The system shall provide the capability to back-up the Data."
Maintainability	The ability to change the system to deal with new technology or to fix defects (External and internal quality [6]).	"The system shall keep a log of all the errors."

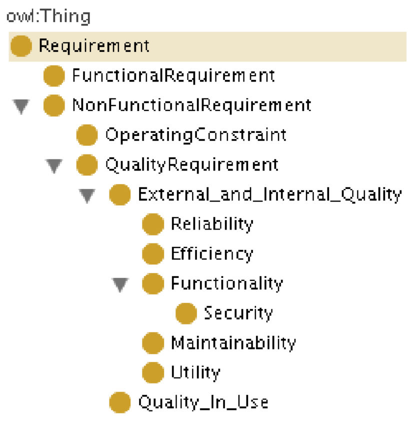


Fig. 1. Requirements ontology (excerpt)

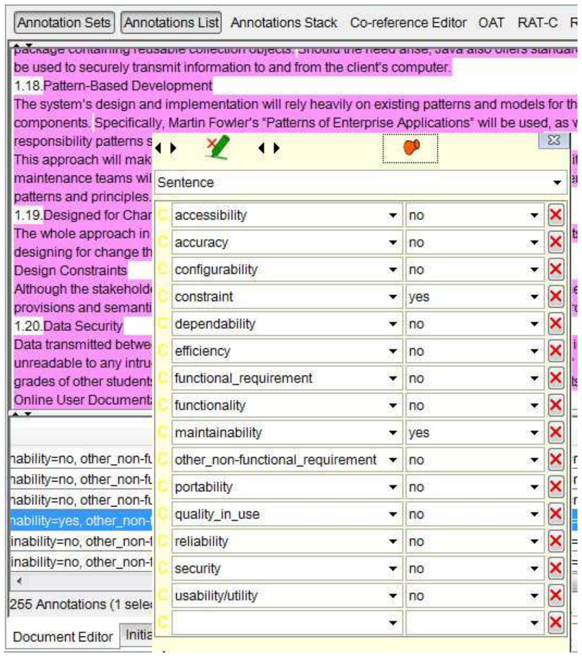
by automatically splitting it into sentences. Each document is assigned to several annotators, who then examine each sentence and select the corresponding type of software requirement. Annotators are free to choose any number of requirements for each sentence, including zero. Fig. 2 shows an example of this annotation process for a single sentence.

1) Corpus statistics: Our Concordia RE corpus categorizes annotation classes into four main categories: *(i)* Functional Requirements (FR); *(ii)* External and Internal Quality: (Accessibility, Accuracy, Configurability, Dependability, Efficiency, Functionality, Maintainability, Portability, Reliability, Security and Usability/Utility); *(iii)* Constraints; and *(iv)* other NFR.

We use standard definitions for each NFR class as outlined above, corresponding to the ISO 9126 standard [6]. Table IV shows the number of annotated sentences for each class.

2) Observed agreement: The agreement between the annotators in our corpus is measured for each pair as shown in Table V. The overall average for whole corpus is 78.36%, which indicates that the quality of data is high and not ambiguous between all annotators.

3) Gold standard: Once the annotators had completed their task, their results were used to create a gold standard for each document. The results of all the annotators for all



be used to securely transmit information to and from the client's computer.
1.18 Pattern-Based Development
The system's design and implementation will rely heavily on existing patterns and models for th
components. Specifically, Martin Fowler's "Patterns of Enterprise Applications" will be used, as v
responsibility patterns s
This approach will mak
maintenance teams wil
patterns and principles.
1.19 Designed for Char
The whole approach in
designing for change th
Design Constraints
Although the stakehold
provisions and semanti
1.20 Data Security
Data transmitted betwe
unreadable to any intru
grades of other student
Online User Document

accessibility	no
accuracy	no
configurability	no
constraint	yes
dependability	no
efficiency	no
functional_requirement	no
functionality	no
maintainability	yes
other_non-functional_requirement	no
portability	no
quality_in_use	no
reliability	no
security	no
usability/utility	no

hability=no, other_non-fu
hability=no, other_non-fu
hability=no, other_non-fu
hability=yes, other_non-
inability=no, other_non-f
inability=no, other_non-f
255 Annotations (1 sele

Fig. 2. Manual annotation process example

TABLE IV
NUMBERS OF ANNOTATION CLASSES SENTENCES PER EACH DOCUMENT. NR: NOT REQUIREMENT, FR: FUNCTIONAL REQUIREMENT, CO: CONSTRAINT, US: USABILITY/UTILITY, SE: SECURITY, EF: EFFICIENCY, FU: FUNCTIONALITY, RE: RELIABILITY.

Doc.	NR	FR	CO	US	SE	EF	FU	RE	Total
1	59	17	26	7	1	1	0	1	112
2	114	32	20	7	10	1	1	2	187
3	180	54	14	6	8	4	1	1	268
4	191	19	21	0	2	3	13	5	254
5	213	23	13	8	2	5	1	0	265
6	1365	642	16	0	34	0	0	0	2057
Total	2122	787	110	28	57	14	16	9	3140

sentences are compared whether they agree on the annotation; if yes, the annotations were retained for the gold standard. For all sentences where annotators disagreed, the gold standard annotation was obtained through a group discussion in regular meetings. Table II summarizes the results from the performed annotation work.

IV. DESIGN AND IMPLEMENTATION

Fig. 3 describes the overall system design, which contains two layers: *(i)* the NFR classifier, and *(ii)* an ontology application layer.

A. Automatic Classification of Requirements

We now describe our automatic sentence-based classifier for requirements documents. The core processing of requirements documents is implemented based on the *General Architecture for Text Engineering* (GATE) [13]. A custom text mining pipeline detects candidate sentences and classifies them using a machine learning algorithms. We trained our system both on the PROMISE and our Concordia RE corpus.

1) Preprocessing: Documents are pre-processed before classification, using standard tokenization, sentence splitting and stemming, as shown in Fig. 4.

2) Training: We built a machine learning-based FR/NFR classifier for SRS documents. The goal of this module is to classify input sentences into 3 major categories, with 7 classes in total: FR (Functional Requirements), Design Constraints, and several types of NFRs (security, efficiency, reliability, functionality, and usability/utility).

TABLE V
OBSERVED AGREEMENT BETWEEN EACH PAIR OF ANNOTATORS

Doc	1,2	1,3	1,4	2,3	2,4	3,4	Avg.
1	0.72	0.57	0.71	0.66	0.74	0.57	0.66
2	0.81	0.78	0.84	0.77	0.83	0.84	0.81
3	0.86	0.85	0.92	0.82	0.841	0.85	0.86
4	0.65	0.68	N/A	0.75	N/A	N/A	0.69
5	0.89	0.85	N/A	0.850	N/A	N/A	0.86
6	0.86	0.79	N/A	0.71	N/A	N/A	0.79
Avg.	0.80	0.75	0.82	0.76	0.8	0.76	0.78

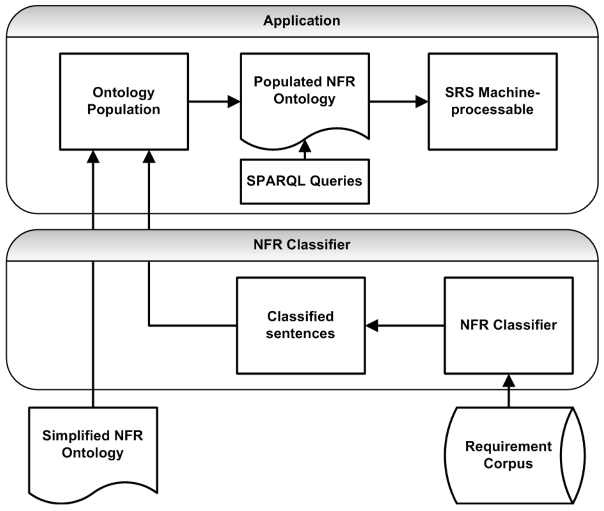


Fig. 3. High-level system design

Example input sentences for each category are: *"The ASPERA-3 data set shall be stored on a local SwRI archive"* (FR), *"These data products will be put into a form known as the Instrument Data File Set (IDFS)"* (CO) and *"The APAF ground data system shall have built-in error handling"* (RE).

In our experiments, Support Vector Machines (SVM) with a third order polynomial kernel provided the best performance. The features used for training are the unigram of the sentences' tokens, using its stem. Instead of a multi-classification, we perform a binary classification for each type of FR/NFR, as some sentences contain two or more types of requirements; e.g., *"Web-based displays of the most current ASPERA-3 data shall be provided for public view"* is annotated as both a functional requirement (FR) and design constraint.

3) Implementation: The above described steps are performed by a pipeline implemented using GATE [13] that first extracts features from the documents, which are then fed into a machine learning component. This pipeline contains Processing Resources (PRs), in particular the ANNIE components [13] and a machine learning PR. To obtain the features for machine learning, documents are pre-processed by using the ANNIE English Tokenizer PR, the ANNIE Sentence Splitter, and the Snowball stemmer.

B. Ontology population

After all sentences in a document have been classified, we populate our NFR ontology with OWL individuals, linking the sentences in the source documents with the corresponding classes in the ontology. This provides for rich queries and

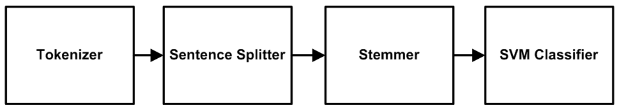
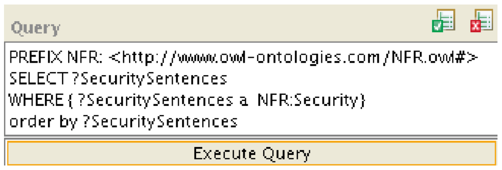


Fig. 4. Classifier Design



```
PREFIX NFR: <http://www.owl-ontologies.com/NFR.owl#>
SELECT ?SecuritySentences
WHERE { ?SecuritySentences a NFR:Security}
order by ?SecuritySentences
```

Fig. 5. SPARQL query for all security NFR sentences in the ontology using Protégé

reasoning on the resulting knowledge base, using SPARQL and an OWL-DL reasoner.

1) Implementation: We integrated the *OwlExporter* [14] component to populate the extracted functional and non-functional requirements into our requirements ontology. The OwlExporter is a GATE plug-in that can export the annotations of a document to individuals in a Web Ontology Language (OWL) model.

2) Application: SPARQL [15] (SPARQL Protocol and RDF Query Language) is a powerful standard query language for ontologies and RDF databases. An example of a simple retrieval of all security sentences from the ontology is shown in Fig. 5.

V. Evaluation

The ML classifier is evaluated using 6-fold cross validation with the metrics *Precision*, *Recall* and *F-Measure* [13], defined as follows:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}, \text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}},$$

$$\text{F-measure} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}},$$

where TP (true positive) is the number of correctly classified requirements, FP (false positive) the number of requirements incorrectly classified, and FN (false negative) the number of requirements incorrectly not classified.

a) Concordia RE Corpus: The results obtained on the Concordia RE corpus are summarized in Table VI. The overall weighted average F1 measure is 84.96%.

TABLE VI
Results for the SVM classifier on the Concordia RE corpus

Class	#	Precision	Recall	F-Measure
FR	787	0.82	0.82	0.82
Constraint	110	0.91	0.91	0.91
Security	57	0.97	0.97	0.97
Usability/Utility	28	0.97	0.97	0.97
Efficiency	14	0.98	0.98	0.98
Functionality	16	0.98	0.98	0.98
Reliability	9	0.99	0.99	0.99
Weighted Average	1021	0.84	0.84	0.84

TABLE VII
Comparison between SVM and Indicator classifiers on PROMISE corpus

Class	SVM			Weighted Indicator [17]		
	Prec.	Recall	F-Meas.	Prec.	Recall	F-Meas.
AV	0.93	0.66	0.77	0.88	0.11	0.19
LE	0.8	0.61	0.69	0.7	0.16	0.26
LF	0.64	0.63	0.64	0.51	0.11	0.19
MA	0.77	0.41	0.53	0.88	0.1	0.19
OP	0.64	0.66	0.65	0.72	0.11	0.19
PE	0.84	0.70	0.76	0.62	0.27	0.37
SC	0.66	0.38	0.48	0.72	0.11	0.19
SE	0.83	0.7	0.8	0.8	0.18	0.29
US	0.79	0.62	0.7	0.98	0.14	0.25
Avg.	0.77	0.6	0.67	0.76	0.14	0.23

b) PROMISE Corpus: We also applied our algorithm on the PROMISE corpus, using Weka [16], in order to evaluate its improvement over previously published results. Table VII compares the performance of our SVM classifier (column SVM) with the approach described by Cleland-Huang et al. [17] (column 'Weighted Indicator'). As can be seen from the table, the precision is roughly comparable, but our approach has significantly higher recall.

A. Analysis

To analyze the classification results, we compute the confusion matrix [18], which presents the false positives and false negatives of the classifier outcomes. Table VIII illustrates the confusion matrices for the seven classifiers. For example, the confusion matrix of the Functional Requirement (FR) Classifier shows that 76 sentences are classified as false positive, and 4 sentences as false negative. The true positive and true negative results are located in the diagonal.

Analyzing the confusion matrices and the evaluation results, we can see that there is still room for improvement for the FR classifier (false positives), as well as the Constraint classifier (false negatives). For improving some of the categories, it will be necessary to increase the amount of training data, especially in the Reliability, Efficiency and Functionality classes, in order to improve the results for these classifiers.

VI. Related Work

Several attempts have been made to develop automated tools for detecting and classifying requirements from SRS using the PROMISE corpus. Cleland-Huang et al. [19] used a stemmer to stem the words of the documents and then selected keywords based on their high probability of occurring in NFR statements. Their system classifies a statement as NFR if the density of selected keywords in that statement exceeds a particular threshold. The LASR system [20] uses linguistic features to train a supervised learning decision trees classifier to classify a requirement sentences into two classes, FR and NFR. Casamayor used a semi supervised learning with TF-IDF technique [21].

TABLE VIII
CLASSIFIER CONFUSION MATRICES

	Yes	No
Yes	774	4
No	76	2210
FR		

	Yes	No
Yes	97	13
No	1	2954
Constraint		

	Yes	No
Yes	56	2
No	9	2998
Security		

	Yes	No
Yes	21	7
No	0	3037
Usability		

	Yes	No
Yes	9	5
No	0	3051
Efficiency		

	Yes	No
Yes	5	11
No	0	3049
Functionality		

	Yes	No
Yes	9	0
No	0	3056
Reliability		

In contrast to these works, we developed a novel NFR corpus that contains richer annotations based on a formal ontology. Our new SVM-based classifier also significantly improves on previously described work.

VII. CONCLUSIONS AND FUTURE WORK

We developed a novel, manually annotated (gold standard) corpus for sentence-based classification of requirements, with a particular focus on non-functional requirements, which have so far not been modeled in detail. Our corpus can be used for many automated analysis tasks and will be made available under an open data license at http://www.semanticsoftware.info/re-corpus.

To improve semantic tool support for the domain of requirements engineering, we developed a new classification algorithm for the automatic categorization of requirements in software specifications, with a particular focus on non-functional requirements (NFRs). The results of this work will be of interest to researchers, as well as practitioners from industry, who are interested in estimating the effort for building requirements in general and improving software quality in particular, and use measurement data in requirements engineering.

The ontological foundation of our work allows to automatically transform software requirements documents into a semantic representation, which can then be further processed in order to *(i)* estimate the cost of the software system and *(ii)* measure the quality of the written requirements. In future work, our goal is to address existing mis-classifications by developing additional syntactic and semantic features for the classifiers. Additionally, we aim to apply SRS quality assurance methods through reasoning on the populated ontology.

The long-term vision of this work is to detect quality problems in different requirement artifacts, in order to decrease the probability of software project failures.

ACKNOWLEDGMENTS

Rolan Abdukalykov, Olga Ormandjieva, Ishrar Hussain, Mohamad Kassab, and Zakaria Siddiqui are acknowledged for annotating the corpus. We also thank Matthew Smith for managing the manual annotation process.

REFERENCES

[1] A. V. Lamsweerde, *Requirements Engineering: From System Goals to UML Models to Software Specifications*, 1st ed. Wiley, 2009.

[2] M. Luisa, F. Mariangela, and I. Pierluigi, "Market research for requirements analysis using linguistic tools," *Requirements Engineering*, vol. 9, no. 1, pp. 40–56, Feb. 2004.

[3] M. Kassab, *Non-Functional Requirements: Modeling and Assessment*. VDM Verlag, 2009.

[4] M. Glinz, "On Non-Functional Requirements," *Requirements Engineering, IEEE International Conference on*, pp. 21–26, Oct. 2007.

[5] L. Chung, B. A. Nixon, E. Yu, and J. Mylopoulos, *Non-Functional Requirements in Software Engineering*. Boston/Dordrecht/London: Kluwer Academic Publishers, 2000.

[6] "ISO/IEC 9126-1:2001 software engineering – product quality – part 1: Quality model."

[7] T. Jingbai, H. Keqing, W. Chong, and L. Wei, "A context awareness non-functional requirements metamodel based on domain ontology," in *Proceedings of the IEEE International Workshop on Semantic Computing and Systems*, ser. WSCS '08. Washington, DC, USA: IEEE Computer Society, 2008, pp. 1–7.

[8] T. H. Al Balushi, P. R. F. Sampaio, D. Dabhi, and P. Loucopoulos, "ElicitO: a quality ontology-guided NFR elicitation tool," in *Proceedings of the 13th international working conference on Requirements engineering: foundation for software quality*, ser. REFSQ'07. Berlin, Heidelberg: Springer-Verlag, 2007, pp. 306–319.

[9] "Promise software engineering repository." [Online]. Available: https://code.google.com/p/promisedata/wiki/nfr

[10] D. L. Mcguinness and F. van Harmelen, "OWL web ontology language overview," W3C, W3C Recommendation, Feb. 2004.

[11] D. Leffingwell and D. Widrig, *Managing software requirements: a unified approach*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2003.

[12] D. Firesmith, *Common Concepts Underlying Safety, Security, and Survivability Engineering*, ser. Technical note. Carnegie Mellon University, Software Engineering Institute, 2003.

[13] Hamish Cunningham et al., *Text Processing with GATE (Version 6)*. University of Sheffield, Department of Computer Science, 2011.

[14] R. Witte, N. Khamis, and J. Rilling, "Flexible Ontology Population from Text: The OwlExporter," in *The Seventh International Conference on Language Resources and Evaluation (LREC 2010)*. Valletta, Malta: ELRA, May 19–21 2010, pp. 3845–3850.

[15] "SPARQL query language for RDF," W3C Recommendation, World Wide Web Consortium, Tech. Rep., Jan. 2008.

[16] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The WEKA data mining software: an update," vol. 11, no. 1. New York, NY, USA: ACM, Nov. 2009, pp. 10–18.

[17] J. Cleland-Huang, R. Settimi, X. Zou, and P. Solc, "Automated classification of non-functional requirements," *Requirements Engineering*, vol. 12, pp. 103–120, 2007.

[18] T. Fawcett, "Roc graphs: Notes and practical considerations for researchers," HP Laboratories, Tech. Rep., 2004.

[19] J. Cleland-Huang, R. Settimi, X. Zou, and P. Solc, "The detection and classification of non-functional requirements with application to early aspects," in *Proceedings of Requirements Engineering, 14th IEEE International Conference*, Minneapolis/St. Paul, MN, 2006, pp. 39–48.

[20] I. Hussain, L. Kosseim, and O. Ormandjieva, "Using linguistic knowledge to classify non-functional requirements in SRS documents," in *Natural Language and Information Systems*, ser. Lecture Notes in Computer Science, E. Kapetanios, V. Sugumaran, and M. Spiliopoulou, Eds., vol. 5039. Springer Berlin / Heidelberg, 2008, pp. 287–298.

[21] A. Casamayor, D. Godoy, and M. Campo, "Semi-supervised classification of non-functional requirements:an empirical analysis," vol. 13, no. 44, 2009, pp. 35–45.