Towards a Software Requirements Change Classification using Support Vector Machine

Amani Khelifa[1], Mariem Haoues[2], and Asma Sellami[1]

[1] Mir@cl Laboratory, University of Sfax, ISIMS, BP 242. 3021. Sfax-Tunisia.
khelifaamani2015@gmail.com, asma.sellami@isims.usf.tn,
[2] Mir@cl Laboratory, University of Sfax, FSEGS, BP 1088. 3018. Sfax-Tunisia
mariem.haoues@isims.usf.tn

Abstract. Requirements are the basis for all software projects. Thus, the requirements phase needs much more attention in order to specify the problems that the software system is intended to solve. However, identifying correctly and completely the software requirements encompasses many issues due mainly to their inconsistencies, ambiguities, incompleteness, and instability. In addition, requirements change requests are inevitable during the software life-cycle (SLC). Change request expressed in natural language format are hard to analyze since they may affect different types of software requirements. To provide an appropriate response to a change request, this paper aims to: (i) investigate how well machine learning techniques are used in the classification of software requirements as well as requirements change requests, and (ii) give an overview of our research that proposes to use the natural language processing and Support Vector Machine (SVM) classifier to automatically classify the requirements change requests into mainly two categories (functional change and technical change).

Keywords: machine learning · software requirements · requirements change requests · natural language processing · classification · functional change · technical change.

1 Introduction

In today competitive world, software organizations have placed a premium on customer satisfaction. Customers want customized products responding to their specific needs. This mandate requires a much closer relationships between developers and customers.

The success or the failure of a software project is highly dependent on the software requirements identification. In fact, software projects are critically vulnerable when the requirements-related activities are poorly performed [12]. In reality, requirements elicitation, analysis and management are very common problems for all the development methodologies. In fact, software requirements are usually expressed in natural language. However, due to the ambiguity inherent in natural language, the requirements specification is prone to a number of errors and flaws [10].

In addition to a customized product, customers are also looking for a rapid delivery. Thus, developers need to be more selective in the use of software attributes. Ultimately, the classification of software requirements is required to allow not only customers and developers to be selective in using software attributes but also project managers to make the right decision. On the other hand, software projects typically involve various customers with different requirements. For that reason, the automated classification of software requirements into functional requirements and non-functional requirements is gaining more attention. However, it is still a challenge due to the variability of natural language and the absence of a controlled vocabulary [1].

Software requirements are subject to change and difficult to articulate during the SLC. A single change request may affect different types of software requirements at the same time. This made their evaluation a hard task. For an accurate change analysis, our research aims to use the natural language processing and SVM to classify the requirements change requests into (i) functional change and (ii) technical change.

The remainder of this paper is organized as follows: Section 2 presents an overview of the requirements engineering and surveys some works studying the software requirements classification. Section 3 gives a summary of our work. Finally, section 4 concludes the presented work.

2 Background and Related Work

This section describes the background for our work and surveys some works.

2.1 Requirements Engineering

Regarding the requirements definition, it is important to distinguish between the user requirements and the system requirements. User requirements are written for the customer mostly in natural language and do not contain any technical details. Throughout the SLC, user requirements are represented using different formats. Whereas, system requirements are written for developers and contain a detailed description of the user requirements including functional and non-functional requirements as well as the technical constraints.

Identifying user requirements is a critical task. In fact, the more user requirements are clear, precise and well-defined, the better software designers will understand the functionality to be developed. In turn, the software testers will be able to understand exactly what the software must do when they verify the developed functionality. Oppositely, unclear, imprecise and inaccurate requirements force the designers and testers to ask for more clarifications.

As shown in Fig. 1, the main activity in the requirements engineering are: Requirement elicitation, Requirement analysis, Requirement acceptance and requirement management. The requirement elicitation determines, explains, and reports stakeholders needs. During the requirements analysis phase, more details

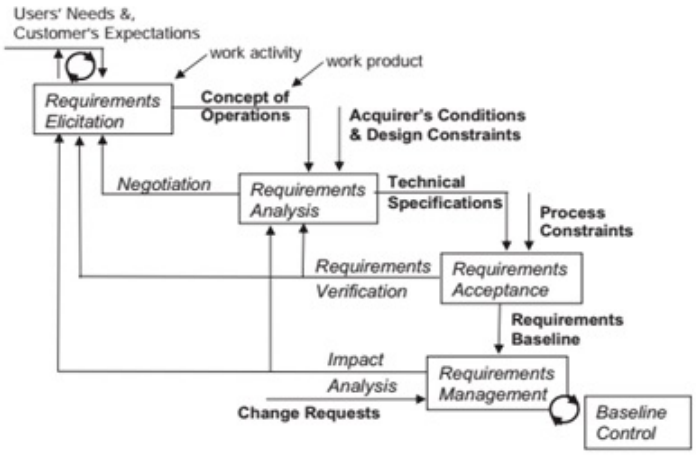

Fig. 1. Process flow of requirements engineering [5]

about the users' needs and the customer's expectations are provided. Both activities require the negotiation with the customer and users. The requirements acceptance concerns the requirements verification. During this activity, requirements defects can be detected. By the end of this phase, the requirements baseline is provided. Thereafter, when a change request is proposed, the requirement management is required in order to analyze the change impact on the requirements baseline. Hence, some modifications can be then made.

2.2 Software Requirements Classification

This section involves how well machine learning methods have been used in the software requirements and change requests classification. For instance,

- Rashwan *et al.,* [10] proposed a method that automatically captures and analyzes requirements written in natural language. Requirements are classified into functional requirements and nonfunctional requirements. This study used a new manually annotated standard corpus based on Promise corpus [11] and a new classifier based on SVM. The execution of their classifier with their own corpus gives an accuracy equal to 84%.
- Kurtanović & Maalej [8] proposed to classify automatically software requirements into functional requirements (FR) and NFR using SVM classifier. They used a sample of Amazon software reviews and evaluated their classifier using different metrics in a series of experiments. They obtained a precision and recall up to ∼92% in distinguishing between FR and NFR. For the identification of specific NFR, they achieved the highest precision and recall for security and performance with ∼92% precision and ∼90% recall.
- Tamai & Anzai [13] proposed to classify software requirements written in natural language into (i) quality requirements (QR) (ii) functional requirement

and (iii) non-Requirement (non-R) using the convolutional neural network. They used thirteen documents available on the web from NIRS [3], IPA[4], and JUAS[5]. The evaluation shows that the precision of classifying FR, QR, and non-R, are respectively equal to 89% , 70%, and 86%.
- Yang & Liang [15] proposed an approach to automatically identify and classify software requirements into FR and NFR using a combination of information retrieval technique (TF-IDF) and Natural Language Processing technique (regular expression). They used the user reviews collected from a popular APP iBooks in English App Store. The results show that the NFR classification has a precision equals to 75% and that of FR is equals to 35%.
- Winkler & Vogelsang [14] mentioned that it is necessary to distinguish between relevant requirements and information in SRS documents. They used the conventional neural networks with a set of 10,000 elements extracted from 89 real-world automotive SRS documents. The proposed approach in this paper was able to extract requirements with a precision equals to 73%.
- Maalej *et al.,* [9] applied several probabilistic techniques to classify app reviews into four types: bug reports, feature requests, user experiences, and text ratings. This study used different classifiers (*e.g.,* Naive Bayes, Decision Tree and MaxEntropie). They showed that app reviews can be classified with an accuracy between 85% and 92%.

Table 2 lists the different approaches proposed in the literature focused on the classification of software requirements and requirements changes. As it can be observed in this table, different classifications for software requirements have been proposed. For instance, researchers distinguished between FR and NFR (*cf.,* [8], [15], etc.). Other studies, such as Rashwan *et al.,* [10], give more refined classifications. Regarding the requirements changes classification, Maalej *et al.,* [9] proposed to classify the user reviews into four categories (Bug reports, Feature requests, User experiences, and Ratings). This classification does not consider the one proposed by the ISO community [2]. In addition, researchers mix between non functional requirements and project constraints as well as the quality characteristics levels.

In our work, we adapt the classification proposed by [2]. Thus, we propose to classify requirements change requests into two major categories, with 10 classes in total: functional change and technical change (functional suitability, performance efficiency, compatibility, usability, reliability, security, maintainability, portability, and Constraint). Where, a functional change request affects the FR and a technical change request affects the NFR or project constraints [7].

3 Our Proposition

This section introduces the research questions, describes the dataset and gives an overview of the research method that will be used in our work.

[3] NIRS: National Institute of Radiological Sciences
[4] IPA: Information Technology Promotion Agency, Japan
[5] JUAS: Japan Users Association of Information Systems

3.1 Research Questions

In our study, we address the following research questions:

1. **RQ1** How well can we automatically classify the requirements change requests as Functional Change (FC) and Technical Change (TC)?
2. **RQ2** How well can we automatically classify the eight NFR classes based on ISO/IEC 25010 [6]?

3.2 Data Set for Requirements Change Requests

Table 1 gives more details about the documents used to collect the requirements change requests in our study. TC and FC are mainly collected from the Promise corpus [11] and the UCI machine learning repository [4]. The Promise corpus includes 15 SRS documents that contain a total of 326 NFR and 358 FR. The NFR types include different quality characteristics (*e.g.,* maintainability, security, performance efficiency, etc.) and quality sub-characteristics (*e.g.,* availability, time-behavior, etc.). On the other hand, the UCI Machine Learning Repository is a collection of databases that can be used by researchers. It has been successfully used within the context of existing software projects. From this database, we collected 303 requirements that includes TC and FC. The total number of requirements change requests collected form these two databases is equal to 1000.

Table 1. Documents Source

Document	**Type**	**Year**	**Categories**	**Number of Requirements**
PROMISE Repository: NFR Data Set	SRS	March 17, 2007	TC	625
PROMISE Repository: WASP Data Set	use case	December 14, 2015	FC	72
UCI Machine learning repository: Re2015 Training set	SRS	March 16, 2015	FC / TC	303
				Total = 1000

Table 2. Summary of the related work focused on the software requirements classification

Study	**Method**	**data set**	**Feature**	**Class**	**Precision**
Rashwan *et al.,* [10]	Super Vector Machine	Promise corpus Concordia corpus	4 SRS, 1 Supplementary specification and 1 Use case **Total** 3064	(i) Functional; (ii) External and Internal Quality; (iii) Constraints; and (iv) other NFR	SVM + promise 77% SVM + Concordia 85%
Kurtanović & Maalej [8]	Super Vector Machine	Amazon software reviews	255 FUR and 370 NFR, **Total** 625	(i) Functional; and (ii) nonfunctional	92% 92%
Tamai & Anzai [13]	Artificial neural network	SRS (Moriyama City, NIRS, JUAS, etc.)	9518 FUR and 1584 QR **Total** 11 102	(i) Functional; (ii) nonfunctional; and (iii) non-requirements	89% 70% 86%
Yang & Liang [15]	NLP Technique & TF-IDF	App ibooks	217 user review FUR 622 user review NFR **Total 839**	(i) functional; and (ii) nonfunctional	37% 75%
Winkler & Vogelsang [14]	Convolutional neural network	Doors database related to an industrial partner	89 SRS **Total 10,000 content elements**	(i) Requirements; and (ii) Information	73% 90%
Maalej *et al.,* [9]	Naive Bayes; Decision Tree; and MaxEnt	Apple store data Google store data	1000 (Apple store) 1000 (Google store) 1200 additional comments iOS 1200 comments for Android application **Total 4400**	(i) Bug reports (ii) Feature requests (iii) User experiences (iv) Ratings	94% 96% 92% 91%

3.3 Overview of the Research Methodology

The main steps of our approach are illustrated in Fig 2.

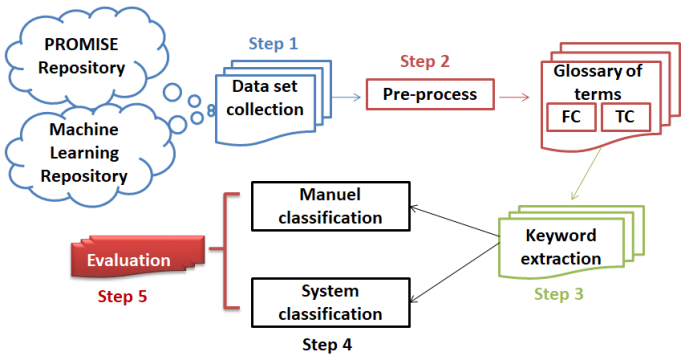

Fig. 2. Phases for the proposed approach

Step 1: data set collection

As mentioned in Table 1, in this study data have been collected from Promise corpus [11] and the UCI machine learning repository [4]. Indeed, the data set to be used includes two SRS documents and one USE CASE document. These documents contain the functional requirements with a total number of 518, Non Functional Requirements with a total number of 469, and 13 constraints. The NFR types include Functional suitability, Performance efficiency, Compatibility, Usability, Reliability, Security, Maintainability, and Portability. Table 3 presents an overview of the data sets, including the software attributes, their corresponding definitions, examples of sentences per class, and their total numbers.

Step 2: Pre-process

The pre-process step is required because of the variability of natural language and the absence of a controlled vocabulary. For instance, we suggest to:

- Tokenize sentences: to classify the content of user reviews in better granularity, we split it into sentences as the unit of classification.
- Eliminate the punctuation marks (*e.g.,* ",", ".", etc.).
- Eliminate stop words (*e.g.,* "is", "the", "on", etc.) in general we have to remove the words with a length of less than three letters.
- Eliminate user reviews that only give the users opinion on a software application (*e.g.,* "Not gooood at all", etc.).
- Eliminate the emotions since they only gave users opinion (*e.g.,* ":(", etc.).
- Transform the slang words or abbreviations into their basic forms (*e.g.,* "idk" into "I don't know", etc.)
- Filtering out spam reviews (*e.g.,* so slow)

Step 3: Extract Keywords

In this step, we first manually identify user reviews greater than ∼70% and classify these reviews into NFRs and FRs or constraints. Based on these classifications we extract keywords to be used for automated identification and classification. The selected concepts (key words) are frequently listed in the requirements documents. The key concepts for NFR definition are derived from the quality characteristics definition as described in the ISO 25010 software quality model [6]. This model includes eight quality characteristics: functional suitability, performance efficiency, compatibility, usability, reliability, security, maintainability, and portability. For instance, "slow", "seconds", "time", "duration" are examples of keywords for Performance efficiency.

Step 4: Apply SVM

This step focuses on how to classify user reviews written in natural language using SVM. For that purpose, collected data falls into training data (700 requirements) and testing data (300 requirements). Thereafter, we use the SVM classifier to automatically classify data into the two categories (FC and TC).

Step 5: Evaluation

In this step, we will use the most known metrics for machine learning evaluation: Precision, Recall and F-measure. These metrics are defined as follows:

$$Precision = \frac{TP}{(TP + FP)} \tag{1}$$

$$Recall = \frac{TP}{TP + FN} \tag{2}$$

$$F - measure = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{3}$$

Where:

- TP: True positives are the number of correctly classified change requests.
- FP: False positives are the number of change requests incorrectly classified.
- FN: False negatives are the number of change requests incorrectly not classified.

4 Conclusion

The main purpose of the herein presented work is to investigate the applicability of machine learning techniques in the classification of software requirements and requirements change requests. Our literature review proved that different techniques have been used to achieve this purpose. However, the proposed classifications do not respect that one given by the ISO community [2]. In addition, researchers confound between concepts (for example NFR characteristic and project constraints).

This paper also gave an overview of our approach that proposes to use SVM for requirements change requests classification into functional and technical change. Our work will be useful for researchers as well as industrial, who are interested in requirements engineering and software estimation.

References

1. Abad, Z., Karras, O., Ghazi, P., Glinz, M., Ruhe, G., Schneider, K.: What works better? a study of classifying requirements. In: the 25th International Requirements Engineering Conference. pp. 496–501 (2017)
2. Abran, A., Castelo, D., Mitwasi, G.M., Vogelezang, F., Aguiar, M., Fagg, P., Soneira, P., Woodward, C., Ben-Cnaan, P., Lesterhuis, A., Symons, C.: Glossary of terms for Non-Functional Requirements and Project Requirements used in software project performance measurement, benchmarking and estimating. (2015)
3. Abran, A., Desharnais, J., Kitchenham, B., Ren, D., Symons, C., Woodward, S., Baklizky, D., Fagg, P., Lesterhuis, A., Santillo, L., Vogelezang, F., Buttle, C., Gencel, c., Meli, R., Soubra, H., Woodward, C.: Guideline on Non-Functional & Project Requirements: How to Consider non-functional and Project Requirements in Software Project Performance Measurement, Benchmarking and Estimating (2015)
4. Asuncion, A., Newman, D.: The UCI Machine Learning Repository of Software Engineering Databases. (2015), https://archive.ics.uci.edu/ml/index.php
5. Fairley, R.: Managing and Leading Software Projects. Wiley-IEEE Computer Society Pr (2009)
6. ISO/IEC: 25010 system and software quality models. Tech. rep. (2010)
7. ISO/IEC14143-1: Information Technology - Software Measurement - Functional Size Measurement. Part 1: Definition of Concepts (2007)
8. Kurtanovi, Z., Maalej, W.: Automatically classifying functional and non-functional requirements using supervised machine learning. In: the 25th International Requirements Engineering Conference. pp. 490–495. (2017)
9. Maalej, W., Kurtanović, Z., Nabil, H., Stanik, C.: On the automatic classification of app reviews. Requir. Eng. **21**(3), 311–331 (Sep 2016)
10. Rashwan, A., Ormandjieva, O., Witte, R.: Ontology-based classification of non-functional requirements in software specifications: A new corpus and svm-based classifier. In: the 37th Annual Computer Software and Applications Conference. pp. 381–386. (2013)
11. Sayysad Shirabad, J., Menzies, T.: The PROMISE Repository of Software Engineering Databases (2005), http://promise.site.uottawa.ca/SERepository
12. Society, I.C., Bourque, P., Fairley, R.: Guide to the Software Engineering Body of Knowledge (SWEBOK(R)). IEEE Computer Society Press, Los Alamitos, CA, USA, 3rd edn. (2014)
13. Tamai, T., Anzai, T.: Quality requirements analysis with machine learning. In: the 13th International Conference on Evaluation of Novel Approaches to Software Engineering, Funchal, Madeira, Portugal. (2018)
14. Winkler, J., Vogelsang, A.: Automatic classification of requirements based on convolutional neural networks. In: the 24th International Requirements Engineering Conference Workshops. pp. 39–45. (2016)
15. Yang, H., Liang, P.: Identification and classification of requirements from app user reviews. In: the 27th International Conference on Software Engineering and Knowledge Engineering, Pittsburgh, PA, USA, July 6-8. pp. 7–12. (2015)

Table 3. Classes' definitions and dataset overview

Class/ Software Attributes	**Definition**	**Example**	**Requirements**
Functional Requirements	functional user requirements that define what the software must do ? [3]	Allows the mobile users to propose and schedule meetings	518
Functional suitability	degree to which a product or system provides functions that meet stated and implied needs when used under specified conditions [6]	-	-
Performance efficiency	Performance relative to the amount of resources used under stated conditions [6]	The system shall refresh the display every 60 seconds	115
Compatibility	degree to which a product, system or component can exchange information with other products, systems or components, and/or perform its required functions [6]	Doesn't work on iPad When I open it crashes my iPad	25
Usability	degree to which a product or system can be used by specified users to achieve specified goals [6]	The product shall have a consistent color scheme and fonts	114
Reliability	degree to which a system, product or component performs specified functions under specified conditions and period of time [6]	The system shall achieve 95 up time	37
Security	degree to which a product or system protects information and data [6]	The system shall prevent attacks including denial of service	72
Maintainability	degree of effectiveness and efficiency with which a product or system can be modified [6]	The product shall be expected to operate for at least 5 years	39
Portability	degree of effectiveness and efficiency with which a system, product or component can be transferred from one hardware, software or other operational environment [6]	The system shall interface with the faculty central server	67
Constraint	project requirements and constraints that express the technical requirements [3]	The website will comply with W3C standards	13
			Total = 1000