Challenges of Non-functional Requirements Extraction in Agile Software Development using Machine Learning

Hoda Mohamed Abd El Sameaa
Department of Information Systems and Technology
Faculty of Graduate Studies for Statistical Research, Cairo University, Egypt

Nesrine Ali abd el Azim
Assistant Professor
Department of Information Systems and Technology
Faculty of Graduate Studies for Statistical Research
Cairo University, Egypt

Nagy Ramadan
Associate Professor
Department of Information Systems and Technology
Faculty of Graduate Studies for Statistical Research
Cairo University, Egypt

ABSTRACT
Current developments in Requirement Engineering methods have seen mutation resulting from the use of machine learning algorithms to resolve several complex Requirements Engineering problems. One of these problems is the identification and classification of non-functional requirements in the requirements documents. Machine based-learning techniques for this challenge have been shown hopeful outcomes than traditional natural language processing approaches. However, there is still lacking of a systematic understanding these machine learning approaches. Despite the fact that non-functional requirements are critical to a software project's success, there is still no accords about what they are and how we will elicit, document, and validate them. Thus, the important task of Requirements Engineering is to properly extract non-functional requirements records from requirement files and arrange them into categories. However, this task is waste of time and prone to errors. This paper presents non-functional requirements importance, relates them to the process of software development and identifies its challenges and current area of research.

General Terms
Requirement specifications, machine learning

Keywords
Requirement Engineering, Non-Functional Requirements, Machine learning

1. INTRODUCTION
The software development methods are System Development Life Cycle (SDLC), Agile, and Object Oriented Analysis and Design (OOAD) methods [1]. Due to the high rate of development of software using agile methods relative to other methods, it becomes one of the most adopted technologies in software projects. Agile method has great value in changing requirements, which increases the productivity and quality of the final product.

Requirement engineering is an essential function of the software development process. Requirements engineering process can be broken down into various stages, such as requirements development and analysis, system modeling, requirements specifications and validation of requirements [2]. Customized software must meet all customer requirements in order to be accepted by the customer or user. Requirements are categorized into two types: Functional Requirements (FRs) and Non-Functional Requirements (NFRs).

Functional requirements (FRs) clarify the function (or service statement) that the system must provide. It is a general assertion of what the framework should do. Non-Functional Requirements (NFRs) describe system properties and constraints that must be met by the system in order to be accepted by customer [3].

Often NFRs are identified and specified in the development process relatively late and are barely explicitly managed. This implies that developers may fail to appreciate the importance of assessing NFRs earlier. The automatic extraction and classification of requirements can help developers to identify and manage NFRs from text documents earlier. The automatic extraction and classification of requirements are the focus of several requirements engineering researchers [1], [4].

2. BACKGROUND
2.1 Non Functional Requirements
Although NFRs have been around since the early days of software engineering, there is no consensus on a name or definition of a NFR. It's been described as a software requirement that doesn't describe what the software will really perform, but rather how it will do it [5]. They are also known as quality characteristics, technical requirements, or quality of provision requirements. NFRs express the software requirements and their effects inside the system such as:

- Performance requirement is concerned with resource required, response time, throughput or other thing else related to performance.
- Security is concerned with system or program protection, like authentication and authorization.
- Reliability is regarded to the capacity of the system to carry out its appropriate tasks and operations in its specific environment without failure or system downtime.
- Usability refers to the ease with which the user can learn, operate, prepare inputs and explain outputs by interacting with the system.
- Scalability requirement is the capability of software to change with the business environment.
- Accessibility requirements are related to user concerns about how easily the system can be used by people with different abilities to fulfill a specific goal in a specific use context.

- Availability is related to determine how long the system runs, how long it takes to fix a fault, and how long it takes between lapses.
- Data integrity: The term "data integrity" refers to the process of maintaining, ensuring data accuracy and consistency throughout its lifecycle. In case of this factor is missing, data is lost because a database is fault.
- Maintainability refers to how easily a software system's faults may be identified and fixed.
- Verifiability refers to the number of tests, analyses, and demonstrations required to evidence that the system will work as purposed.
- Interoperability is referred to the ability of a software system to join or simplify the interface with other systems.
- Portability is related to the variety of hardware and software platforms on which an information system can run, as well as the ease with which the system can be moved from one environment to another.
- Reusability is the degree to which components of an information system, or its design, can be reused in the development of other applications.
- Flexibility refers to the ability of a user to make changes to an information system without having to change the software itself in order to adapt to diverse environments, configurations, and user expectations.

2.2 Machine learning

Machine learning [6] is a kind of Artificial Intelligence that enables software to act intelligently. Machine learning's main goal is to create algorithms that can take in inputs and utilize statistical analysis to expect outputs within a reasonable range.

There are two types of machine learning algorithms: supervised and unsupervised learning. In supervised learning, a person supplies the necessary income and output, as well as the accuracy of forecasts during the algorithm's training. The algorithm applies what it has learned to new data after it has finished learning. But unsupervised learning does not require the algorithm to be trained with the required outputs, it relying on an iterative process known as deep learning to analyze data.

Fraud detection, spam filtering, network security threat identification, predictive maintenance, and creating News feed are all examples of machine learning applications[6].

2.3 Agile requirements engineering

Agile requirements engineering [7] differs from conventional requirements engineering in that it takes an iterative and incremental discovery approach. Unlike traditional requirement engineering, agile does not encourage upfront detailed planning for the entire project, promotes both quick clean delivery and customers' involvement in the entire process of software development. Although agile engineering requirements improve the understanding of customer needs and the ability to adapt to the evolving needs of the current dynamic environment, they present clear challenges concerning NFRs.

3. RELATED WORK

Kurtanovic et al [8] classified requirements into Functional Requirements (FR), Non-Functional Requirements (NFR), and subcategories of non-functional requirements using the Support Vector Machines (SVM) algorithm. The researchers used the PROMISE repository, which is the most commonly used requirements database and it is known to present an unbalanced set of functional and non-functional requirements. However, there are only 625 categorized (labeled) specifications written in natural language in this repository, which is unbalanced.

Maalej and Nabil [9] proposed a feature of app store analytics that automatically classify user feedback into bug reports, feature requests, user impressions, and ratings (i.e. simple praise or dispraise repeating the star rating). the authors presented a number of results that can be used to help designers build review analytic tools.

The results of this research don't apply to particular stores (such as Amazon), other languages with another orientations and inferences than English, or other types of reviews, like hotel or movie reviews.

The authors use supervised learning techniques to implement automatic classification. the authors did this by manually labeling 4,400 reviews using content analysis in order to construct a truth set. This is clearly a high overhead, would be making scale difficult, using semi-supervised learning can avoid this problem.

Deocadez et al [10] applied SSL techniques in a dataset of App Store reviews to demonstrate their feasibility and capabilities for both transductive (predicting current instance labels during training) and inductive (predicting labels on unseen future data) results.

The results show that only a small amount of data is required to achieve comparable results to classical supervised techniques, and the trained models can correctly assign labels to the collected data as well as identify unseen future reviews. But the results may not apply to other app stores, especially those with varying quality assurance standards. Furthermore, they only considered reviews written in English, so the results cannot be applied to reviews written in other languages [10].

On the basis of lingual relations, in [11] the authors suggested a rule-based technique for classifying non-functional requirements from the PROMISE corpus. To automatically extract thematic roles from the SRS documents, the researchers used text mining preprocessing methods like ANNIE, MuNPEx, JAPE, Snowball, and ANNIE POS Tagger. This proposal aims to reduce the pressure and effort required by designers and analysts in identify NFRs from very vast requirement documents, as well as to enhance the quality of these documents. The precision and recall of the PROMISE corpus classification findings were 97 percent and 96 percent, respectively. The researchers, on the other hand, do not go into detail on the machine learning techniques employed in the classification.

In [12] the research aims to help analysts extract relevant non-functional specifications from accessible unconstrained natural language documents more efficiently using automated natural language processing. The authors developed NFR Locator, a tool-based method for classifying and extracting sentences from existing natural language texts into their relevant NFR categories, to achieve this aim. Despite NFR Locator was dealing with textual sentences, it can't extract data from images or tables. Also the process and tool cannot be applied to other systems or domain.

Chuanyi Li et al [13] suggest a classification approach that uses combined of project-specific and non-project-specific keywords, as well as machine learning algorithms. The

suggested technique succeeds in attaining high classification accuracy by employing keywords as features, decreased significant human labor in developing machine learning-based classifiers, and obtained steady performance in detecting minority groups, regardless of the number of few cases they have. The authors found that using the suggested features, SVM performed the best, also this method worked effectively when working with unbalanced request classes. However, User requisitions types that were used as categorization goals throughout this paper were manually identified by aggregating current requirements categories. Different people might have dissimilar thoughts about how to categorize and name these groups. Furthermore, Not-specific to the project keywords for every kind of client request are manually extracted from requirements papers, which submissive to researcher prejudice.

Deocadez et al [14] suggest using semi-supervised learning to automatically classify functional and non-functional requirements in mobile app reviews published to app stores. The authors assessed the performance of the three semi-supervised learning algorithms that's chose using the standard accuracy metric. For classifying the apps reviews, the authors used the FURPS (Functionality, Usability, Reliability, Performance, and Supportability) as a basis model. But due to differences between app stores in quality assurance standards its make it hard to popularize to other app stores. The authors only analyzed reviews written in English; therefore their outcome cannot be applied to reviews written in other languages. The authors used the performance measurements for this study F-Score = 2 (P R)/ (P + R) [9], the authors assert that this measure is extensively utilized for information retrieval tasks, but they don't give the reasons for using it.

In [15], used a combination of four classification techniques BoW, TF-IDF, CHI2, and AUR-BoW to automatically classify user reviews into four categories of NFRs (reliability, usability, portability, and performance), FRs, and Others with three machine learning algorithms Naive Bayes, J48, and Bagging[15]. The authors used textual semantics to enrich user reviews by using word2vec to classify user reviews automatically. The authors compared user reviews from two popular Apps: iBooks and WhatsApp, which belong to various categories (domains) from different App stores, to the combinations of categorization approaches and machine learning algorithms (platforms). The authors performed tests to compare the F-measure of classification results across all combinations and discovered that combining AUR-BoW with Bagging results in the highest F-measure. However, the machine learning algorithms applied in this study were Naive Bayes, J48, and Bagging without providing the reasons for applying these algorithms. They picked these algorithms for user review categorization since they had previously been successfully used for object classification in a number of previous studies.

In [16], the authors used unsupervised machine learning algorithms to classify NFRs. The Latent Dirichlet Allocation (LDA) algorithm classifies documents dependent on the frequency of word co-occurrences[12].The Biterm Topic Model (BTM) technique, differently from the LDA approach, is dependent on word co-occurrence manner and learns topics by looking for word-word (such as biterm) manner. The Naive Bayes Classification method is a supervised learning method that uses preprocessing to decrease requirement specification asymmetry through taking benefit of rich sentence characteristics and latent co-occurrence relations. However, clustering algorithms used for classifying NFRs had a poor performance. This could mean that the dataset under study is completely disorganized and subcategories of NFRs aren't separated fully. So, an unsupervised algorithm (like Hierarchical or K-Means) will fail to fulfill exactly segmentation.

The aim of [17] is to assist application developers to identify and study NFRs in application development by automatically classifying user reviews. In order to reduce human efforts and to obtain valuable information from user reviews. In this study the authors analyze the security requirements descriptions existent in the Software Requirements Specification (SRS) document and then develop the classification models. By Using the J48 decision tree method, the authors analyze the descriptions of the security requirements using text mining tools, and after that categorized them to four types of security requirements: authentication, authorization, access control, cryptography-encryption, and data integrity. the authors constructed the prediction model in the same way they did for any other type of security requirement. However, this study adapted ROC value to compute the performance of their classifiers. But ROC suffers from less bias compared to other measures, such as the F-Score, which skews toward the positive class, especially in case of an imbalanced class. the authors doing experiments by using Weka3 with different machine learning algorithms: Naive Bayes, J48, and Bagging. the authors did not explain why this algorithm was used to classify user reviews. It is claimed that these techniques were chosen because they had previously been successfully used for object classification in several previous research [17].

4. CHALLENGES OF NFR EXTRACTION

Despite the many advantages offered by considering NFR in software developments, however, there are still some associated challenges.

- In traditional software development technology as well as in agile software development technology, the users and developers spent most of their efforts on modeling functional requirements, while non-functional requirements are often neglected or retrofitting late in software development can lead to lower quality and errors in later stages of the development process[18]which often result in projects failing.
- Despite the fact that non-functional requirements are becoming more widely regarded as a vital success metric for projects, it receive less attention in software development industrial practices than functional requirements. There are numerous guidelines for sketching and modeling functional requirements. Focusing on NFR and specifying NFR in conjunction with FR is still an open research study, also integrating NFR into the different phases of software development is still difficult and very much challenging.
- Typically, People are capable of figuring out what they want from the system, but they often don't care or realize how to get it. To develop a high-quality software product, NFR needs to be extracted from the requirements documents to be implemented.
- Can't deal with every NFR in the same way, for example, usability and security requirements must be handled in a different way. As a result, defining one solution to deal with all NFRs is problematic.

- Different stakeholders have different perspectives and preferences when it comes to eliciting functional and non-functional requirements. The NFR gathered from various stakeholders may be conflict with one another. Rarely can NFR be said to be fulfilled. Conflict resolution decisions must be based on priority.
- Ambiguous specification of the system's features is usually the most important concerns, and the insufficiency in addition to lack of consistency of these features usually results in a system where customer gratification remains a question mark.

5. CONCLUSION& FUTURE WORK

Non Functional requirements (NFRs) play a major function in the success of software systems and it is as important as Functional requirements. It is necessary to deal with NFRs at early stages of agile software developments because its affect the database decision, programming language, or operating framework. As the absence of NFRs requires a high cost to solve the problem. Its offer high quality product that is accurate, consistent and reliable. This paper suggests developing an efficient approach for automatic extraction of NFRs in agile methods in order to prevent the problems that are found previously, and try to improve the performance through customer satisfaction. The goal is to be able to extract NFRs that are embedded in FRs and classify NFRs in available documents through automated natural language processing and machine learning techniques.

6. REFERENCES

[1] A. M. Davis. Software requirements: objects, functions, and states. Prentice-Hall, Inc., 1993.

[2] Handa, N., Sharma, D. A., & Gupta, D. A. (2019). Non Functional Requirements Analysis using Data Analytics. International Journal of Advanced Science and Technology, 27, 383 - 393.

[3] Binkhonain, Manal, and Liping Zhao."A review of machine learning algorithms for identification and classification of non-functional requirements." Expert Systems with Applications: X 1 (2019): 100001.

[4] Kurtanovi ´c, Z., &Maalej, W. (2017, September).Automatically Classifying Functional and Non-functional Requirements Using Supervised Machine Learning. Paper presented at IEEE 25th International Requirements Engineering Conference Workshops, Lisbon, Portugal.

[5] NupurChugh and AdityaDev Mishra, "Assimilation of Four Layered Approach to NFR in Agile Requirement Engineering", International Journal of Computer Applications (0975 – 8887) Volume 78 – No.5, September 2013.

[6] https://searchenterpriseai.techtarget.com/definition/machine-learning-ML.

[7] Sillitti, Alberto, and Giancarlo Succi."Requirements engineering for agile methods."Engineering and Managing Software Requirements. Springer, Berlin, Heidelberg, 2005.309-326.

[8] Kurtanovi´c, Z.; Maalej, W. Automatically classifying functional and non-functional requirements using supervised machine learning. In Proceedings of the 2017 IEEE 25th International Requirements Engineering Conference (RE), Lisbon, Portugal, 4–8 September 2017; pp. 490–495.

[9] W. Maalej and H. Nabil. 2015. Bug report, feature request, orsimply praise? On automatically classifying app reviews. In IEEE23rd International Requirements Engineering Conference (RE).

[10] Deocadez, R., Harrison, R., & Rodriguez, D. (2017, June). Preliminary Study on Applying Semi-Supervised Learning to App Store Analysis.In Proceedings of the 21st International Conference on Evaluation and Assessment in Software Engineering (pp. 320–323).

[11] P. Singh, D. Singh and A. Sharma, "Classification of Non-functional Requirements from SRS Documents Using Thematic Roles," 2016 IEEE International Symposium on Nan electronic and Information Systems (iNIS), Gwalior, 2016, pp. 206-207.

[12] J. Slankas and L. Williams, "Automated extraction of non-functional requirements in available documentation," 2013 1st International Workshop on Natural Language Analysis in Software Engineering(NaturaLiSE), 2013, pp. 9-16, doi: 10.1109/NAturaLiSE.2013.6611715.

[13] Chuanyi Li , Liguo Huang , JidongGe , Bin Luo , Vincent Ng , Automatically Classifying User Requests in Crowdsourcing Requirements Engineering, The Journal of Systems & Software (2017), doi: 10.1016/j.jss.2017.12.028.

[14] Deocadez, R. , Harrison, R. , & Rodriguez, D. (2017). Automatically classifying requirements from app stores: a preliminary study. In Proceedings of the IEEE twenty–fifth international requirements engineering conference workshops September.

[15] Lu, M., & Liang, P. (2017, June).Automatic Classification of Non-Functional Requirements from Augmented App User Reviews.Proc.21st International Conference on Evaluation and Assessment in Software Engineering, Karlskrona, Sweden.

[16] Abad, Z. S. H. ,Karras, O. , Ghazi, P. , Glinz, M. , Ruhe, G. , & Schneider, K. (2017). What works better? A study of classifying requirements. In Proceedings of the IEEE twenty-fifth international requirements engineering conference (RE) September.

[17] Jindal, R. ,Malhotra, R. , & Jain, A. (2016). Automated classification of security requirements. In Proceedings of the 2016 international conference on advances in computing, communications and informatics September.

[18] Tamai, T., &Anzai, T. (2018). Quality Requirements Analysis with Machine Learning. In ENASE (pp. 241-248).