NFR Modeling from Explicit Natural Language Requirements Specification

1M. Durgadevi and 2S. Abirami

1Department of Information Science and Technology, Anna University, Chennai, India – 600 025.
2Department of Information Science and Technology, Anna University, Chennai, India – 600 025.

ABSTRACT

Non functional requirements are the constraints that must be adhered during system development. Non functional requirement (NFR) may be more critical than functional requirement, if they are not met, then the system becomes useless and eventually lead to failure of the system. This paper proposes novel techniques to aid analysts in extracting NFRs more effectively in available unconstrained natural language requirement specification through automated natural language processing and also automate the NFR model generation for uniform comprehension. The proposed work is mainly focused on identifying and extracting NFRs by using a robust set of rules on the requirements document. From the evaluation, it is clear that the overall performance of the system is consistent and precision and recall rate of system performance are also high.

INTRODUCTION

Requirements analysis is an essential phase in system analysis and encompasses tasks that go into determining the needs or conditions to meet for a new or altered product, taking account of the possibly conflicting requirements of the various stakeholders, such as beneficiaries or users. It is the initial stage of the systems analysis process, and perhaps the most important. If the client’s requirements are not gathered and defined accurately, the rest of the project becomes meaningless, since it does not reflect what the client actually wants. Mostly researches in the literature contribute towards the extraction of function requirements from natural language text.

In addition to Functional Requirements (FR), Non-Functional Requirements (NFR) should also be considered while creating a conceptual model. Just as with functional requirements, a system’s success depends greatly upon adherence to non-functional requirements Hoskinson(2011). Non-functional requirements define the overall qualities or attributes of resulting system. NFRs place restrictions on the product being developed, the development process and specify external constraints that the product must meet. When NFRs are missed or ignored, significant, costly issues can arise Bertman(2010). It is given the need to analyze and implement NFRs from a wide variety of available sources, and system analysts need to quickly identify and categorize NFRs. Therefore, this project work proposed a system, named as NFR extractor which extracts Non-Functional Requirements automatically from available natural language requirements document and use them to create a NFR model.

Forth coming sections were described the above in detail. Section 1 presents related work, and a summary of results or progress in the field of automation of requirements analysis. A brief survey of the supplementary topics, i.e. visualization of conceptual models is also explored. In section 2, the architecture of the system is presented in the form of a visual layout of the functional blocks. The algorithms and functionality of the each of the functional blocks and modules within is explained. This chapter concludes with a summary of the functionality. Section 3 deals with the performance evaluation for the proposed work. Section 4 concludes this research work, with the achievements of the current research, shortcomings of the same, and offers suggestions on further improvements.

Corresponding Author: M. Durgadevi, Department of Information Science and Technology, Anna University, Chennai, India – 600 025, E-mail: m.durgadevit@gmail.com

© 2015 AENSI Publisher All rights reserved.
1. Related Work:

While NFRs have existed since the early days of Software Engineering, consensus does not exist for the name or the definition of a NFR Glinz (2007). Many simply refer to them as the “ilities,” the quality aspects of a system. Others have taken to labeling NFRs as systemic requirements by Browne (2008). The IEEE Recommended Practice for Software Requirements terms NFRs as constraints (IEEE) (1998). Our concern resides with how NFRs place different constraints on systems, how to quickly identify such constraints, and then to extract relevant information for the NFR. While the number of such constraint categories is rather large Chung et al. (2000) identified 156 NFR categories, the proposed work choose to concentrate on 6 categories frequently appearing in literature and practical use. Non Functional Requirements specify how a system should be rather than what a system should do. The latter is specified by the functional requirements of the system. As already mentioned, NFR should be identified and represented in conceptual model so as to get the stake holder’s acceptance on the complete working of the project. Failure to do the above task has lead to the failure of majority of the software projects.

There are two ways of representing NFRs namely User Requirements Notation (URN) and Goal oriented Requirements Language (GRL). This was discussed by Saleh et al. (2004). GRL specifies three categories of constructs that are used to project the NFRs namely intentional elements, actors and links or relationships. An intentional element can be a goal, task, soft goal, belief or a resource. Saleh added that NFRs are difficult to elicit by non-expert users and hence analysts are needed to extract correct NFRs.

Imran Sarwar Bajwaet al. (2012) discussed a novel approach to automatically transform Natural Language (NL) specification of software constraints to Object Constraint Language (OCL) constraints. NL specifications were subjected to shallow semantic and deep semantic parsing to extract the concepts and constraints and were finally represented in OCL according to the chosen template as UML models are incomplete without textual constraints.

According to UML, a note is used to represent a NFR. But this is not sufficient. Hence a class diagram can be used to model NFRs according to Saleh. Non-Functional attributes were represented in the second compartment of the class. The name of the class corresponded to the NFR being modeled and the entire requirement was specified in a note using OCL.

2. NFR Modeling:

The functional blocks involved in the system that performs automatic generation of the NFR model from natural language text is shown in the Figure 1. The boxes with thick outlines indicate the major modules and the ones with thin outlines indicate input or output of these major modules. The dashed circle represents the pre-defined rules that are applied on the input to get the output of the module.

![Fig. 1: Architecture of NFR Extractor](image-url)
The system architecture of NFR extractor has been divided into six major modules namely,

- Sentence tokenization
- Classification of FR and NFR
- Grouping similar NFR sentences
- Deep parsing
- Extraction of design elements of NFR
- Visualization

The flow of the system is as follows: First, the requirements document is subject to sentence tokenization. Then each tokenized sentence is classified into functional and Non Functional Requirements (NFRs) by using Naïve Bayes Classifier. Grouping of similar NFR sentences into any one of these NFR categories such as Performance, Security, Look and Feel, Availability, Scalability and Maintainability. Then each sentence is subject to deep parsing. Using parsed sentence’s typed dependencies, explicit design elements of specific NFRs are extracted using a set of predefined rules. Importance Score of each NFR category would be calculated based on the frequency of NFR and its relative importance to that particular domain. Finally, visualization of the NFR model is represented by all the extracted design elements of NFRs in such a way that it is easy to comprehend the problem.

**Grouping Similar NFR sentences:**

Sentences which are classified as NFR would be input of this module. Grouping of similar NFR sentences into any one of the NFR categories such as Performance, Security, Look and Feel, Availability, Scalability and Maintainability was done by string matching algorithm using keywords as tabulated in Table 1. The set of keywords for each NFR category were identified as illustrated in Slankas et al. (2013). In Table 1 (*) indicates keywords that were newly identified in this proposed work.

**Table 1:** Keywords for each NFR category

<table>
<thead>
<tr>
<th>NFR Category</th>
<th>Keywords</th>
</tr>
</thead>
<tbody>
<tr>
<td>Performance</td>
<td>fast, every, seconds, up to date*, capable, average, in acceptable time*, in timely manner*, response time*, capacity, returned no later*, no longer than*, in under*, within seconds, must be completed*, maximum processing time*, minimum processing time*, degradation of performance*, in acceptable time*, able to complete*</td>
</tr>
<tr>
<td>Look</td>
<td>appearance, match the color*, user interface*, interface*, look and feel*, appealing*, color, look, navigation, feel, attractive*, animation*, design, menu, simulate the look*, GUI*, display, appear, similar format*</td>
</tr>
<tr>
<td>Availability</td>
<td>achieve, 24 hours per day*, availability, available, uptime*, downtime*, between hours*, everyday*, demand at any time*, 365 days per year*, all days of year*</td>
</tr>
<tr>
<td>Security</td>
<td>cookie, encrypted, only registered*, ensure*, accessed by only authorized*, authorized, vulnerability*, username, inactivity, authenticate*, certificate, session, password, computer viruses*, integrity checking*, security, permission*, privilege*, protect*, private*</td>
</tr>
<tr>
<td>Scalability</td>
<td>Support*, multiple remote users*, multiple users*, simultaneous, handle, concurrent* per hour*, per day*, per second*, per minute*, at same time*, at a time*, handle up to*, grow*, manage*</td>
</tr>
</tbody>
</table>

Whenever a sentence contains any one of the keyword or combination of keywords which are specific to each NFR category as mentioned in above Table 1 , then that sentence grouped in that particular NFR category.

**Example 1: Performance:**

*Whenever a sentence contains ‘at least within’, ‘every hour’, ‘in acceptable time’ and etc. (clauses as tabulated in Table 1), then it is grouped into the performance NFR category.*

The system should respond to user within 60 seconds.

The system should process the result in acceptable time.

Likewise, all other NFR sentences are also grouped to any one or more than NFR category by using the method as described above.

**Extracting Design Elements from each NFR Sentence:**

The extraction of design elements that are used to form the NFR model is the core of the functional module. Here, we create the NFR model from the building blocks of subject, operation and value. The syntactic information is used to determine the valid constituents of the NFR model. At this stage, typed dependencies of each sentence are used. We examine the typed dependencies for each sentence against a set of rules. In these rules, we explore the grammatical relationships among the constituents of the sentence, and decide whether to create subject, operation, value.

While some rules are from established linguistic patterns from reference books, others are created afresh to take advantage of the deep parsing method that is used to extract grammatical structures. The rules are grouped...
into design rules. The identification of subject, operation and value is as a result of these rules. Rules used for extracting subject and operation are common to all NFR categories while rules applied for extracting NFR value are distinct to each NFR category.

**Rule for extracting subject:** Any noun that appears as a subject is identified as subject of NFR model:

For example, the sentence, ‘The system shall be able to handle 1000 simultaneous users.’ results in the dependencies det(product-2, The-1), nsubj(able-5, system-2), aux(able-5, shall-3), cop(able-5, be-4), root(ROOT-0, able-5), ccomp(able-5, support-6), num(users-9, 1000-7), amod(users-9, simultaneous-8), dobj(support-6, users-9). Here, ‘system’ is identified as it is the ‘subject’.

**Rules for extracting operation:** Any verb that appears as root or clausal complement is identified as operation of NFR model which is done by that particular subject:

For example, the sentence, ‘The system shall be able to handle 1000 simultaneous users.’ results in the dependencies det(product-2, The-1), nsubj(able-5, system-2), aux(able-5, shall-3), cop(able-5, be-4), root(ROOT-0, able-5), ccomp(able-5, support-6), num(users-9, 1000-7), amod(users-9, simultaneous-8), dobj(support-6, users-9). Here, ‘support’ is identified as it is the ‘operation’.

**Rules for extracting values:** Rules which are used for extracting values are varied for each NFR category.

### Performance:

The performance NFR value is extracted by using grammatical relationships, such as admod (adjective modifier) and num (numeric modifier) as illustrated in Table 2 below.

<table>
<thead>
<tr>
<th>Sentence</th>
<th>Result of parsing</th>
<th>Extracted value</th>
</tr>
</thead>
<tbody>
<tr>
<td>The product shall produce search results in an acceptable time.</td>
<td>det(product-2, The-1), nsubj(produce-4, he-1), nsubj(produce-4, product-2), aux(produce-4, shall-3), root(ROOT-0, produce-4), nmod(results-6, search-5), dobj(produce-4, results-6), det(time-10, an-8), amod(time-10, acceptable-9), prep_in(produce-4, time-10)</td>
<td>Acceptable time</td>
</tr>
<tr>
<td>The system shall refresh the display every 60 seconds.</td>
<td>det(system-2, The-1), nsubj(refresh-4, he-1), nsubj(refresh-4, system-2), aux(refresh-4, shall-3), root(ROOT-0, refresh-4), det(display-6, the-5), nmod(seconds-9, display-6), det(seconds-9, every-7), num(seconds-9, 60-8), xcomp(refresh-4, seconds-9)</td>
<td>60 seconds</td>
</tr>
</tbody>
</table>

As illustrated in Table 2, the values in typed dependency amod(time-10, acceptable-9) and typed dependency num(seconds-9, 60-8) are extracted using regular expression.

### Security:

The security NFR value is extracted by using grammatical relationships, such as admod (adjective modifier) and nn (noun compound modifier)

### Availability:

The availability NFR value is extracted by using grammatical relationships, such as admod (adjective modifier) and conj (conjunct)

### Look and Feel:

The look and feel NFR value is extracted by using grammatical relationships, such as admod (adjective modifier), dobj (direct object), cop (copula) and prep (preposition)

### Scalability:

The scalability NFR value is extracted by using grammatical relationships, such as admod (adjective modifier) and num (numeric modifier)
Maintainability:

The maintainability NFR value is extracted by using grammatical relationships, such as admod (adjective modifier) and advmod (adverbial modifier).

Extraction of subject, operation and value for sample input requirements specification sentences by using a set of predefined rules is illustrated in Table 3.

Table 3: Extracted design elements for sample input NFR sentences

<table>
<thead>
<tr>
<th>Sentence</th>
<th>Subject</th>
<th>Operation</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>The system should respond to users within 30 seconds.</td>
<td>System</td>
<td>Respond</td>
<td>30 seconds</td>
</tr>
<tr>
<td>The interface should have simple look.</td>
<td>Interface</td>
<td>Have</td>
<td>Simple look</td>
</tr>
<tr>
<td>The system will be available to technical team on 24×7 basis.</td>
<td>System</td>
<td>Available</td>
<td>24×7 basis</td>
</tr>
<tr>
<td>The records should accessed only by authorized users.</td>
<td>Records</td>
<td>accessed</td>
<td>Authorized users</td>
</tr>
<tr>
<td>The product will be updated on regular basis.</td>
<td>Product</td>
<td>Update</td>
<td>Regular basis</td>
</tr>
<tr>
<td>The system should handle 1000 simultaneous users.</td>
<td>System</td>
<td>Handle</td>
<td>1000 simultaneous users</td>
</tr>
</tbody>
</table>

NFR Model generation:

After all design components for each NFR category are extracted the NFR model is created by programmatically creating code in the DOT programming language, which is part of Graphviz, graph visualization software. The dot layout offers a hierarchical layout, with the edges and nodes laid out in a top to bottom, left right manner. The generated NFR model for sample input NFR sentence is shown in Figure 2.

Fig. 2: NFR Model

3. Performance Evaluation:

Evaluation of Specific NFR Extraction:

The performance evaluation of the process of extracting specific Non Functional Requirements has been done by calculating precision, recall and F-score based on True Positive (TP), False Positive (FP), False Negative (FN) values. TP indicates correct classification. FP indicates an NFR belonging to a different category being classified under the category being tested. FN indicates an NFR belonging to the current category being classified under a different category. F-score gives a weighted average of the precision and recall values.

The precision, recall and F-score are calculated using the following formulas given in 1, 2 and 3:

\[
\text{Precision} = \frac{TP}{TP+FP} \tag{1}
\]

\[
\text{Recall} = \frac{TP}{TP+FN} \tag{2}
\]

\[
F - \text{score} = 2 \times \frac{P \times R}{P + R} \tag{3}
\]

The performance was evaluated by running the process for the data extracted from Promise Dataset and the results are summarized in the Table 4 and in figure 3.
From the evaluation results, it is clear that performance is improved, specifically precision and recall of the system is relatively high compared to other approaches. Though overall performance of the system is consistent, yet extraction of availability NFR had less precision and recall, means it had high false positive and false negative rate. So Rules that are used for extracting availability NFR has to be improved.

![Fig. 3: Performance Evaluation of Performance Extraction](image)

### REFERENCES


https://github.com/RealsearchGroup/NFRLocator, the website for Promise dataset.