The Detection and Classification of Non-Functional Requirements with Application to Early Aspects

Jane Cleland-Huang, Raffaella Settimi, Xuchang Zou, Peter Solc
Center for Requirements Engineering
School of Computer Science, Telecommunications, and Information Systems
DePaul University
{jhuang,rsettimi,xzou}@cs.depaul.edu

Abstract

This paper introduces an information retrieval based approach for automating the detection and classification of non-functional requirements (NFRs). Early detection of NFRs is useful because it enables system level constraints to be considered and incorporated into early architectural designs as opposed to being refactored in at a later time. Candidate NFRs can be detected in both structured and unstructured documents, including requirements specifications that contain scattered and non-categorized NFRs, and freeform documents such as meeting minutes, interview notes, and memos containing stakeholder comments documenting their NFR related needs. This paper describes the classification algorithm and then evaluates its effectiveness in an experiment based on fifteen requirements specifications developed as term projects by MS students at DePaul University. An additional case study is also described in which the approach is used to classifying NFRs from a large free form requirements document obtained from Siemens Logistics and Automotive Organization.

1. Introduction

Non-functional requirements (NFRs) describe important constraints upon the development and behavior of a software system. They specify a broad range of qualities such as security, performance, availability, extensibility, and portability. These qualities play a critical role in the architectural design [15] and should therefore be considered and specified as early as possible during system analysis.

Unfortunately NFRs are often discovered in an ad-hoc fashion relatively late in the development process. Quality constraints of stakeholders, elicited during the requirements gathering process, get documented across a range of artifacts including memos, interview notes, and meeting minutes, and analysts frequently fail to obtain a clear perspective on the system-wide non-functional requirements. Resulting requirements specifications are often organized by functionality with non-functional requirements scattered throughout the specification. This can lead to important conflicts going undetected and architectural solutions that fail to meet the stakeholders' real needs. This paper proposes a solution to this problem by introducing a method, known as the NFR-Classifier, for retrieving and classifying NFRs scattered across both structured and unstructured documents.

The NFR-Classifier can also be used to detect and classify early aspects. An early aspect is a concern that cross-cuts the dominant decomposition of a system [4] and is found in the requirements specification or other early design documents. Many "early aspects" are identical to high-level NFRs such as security, performance, portability, and usability. Intermediate level aspects include concerns such as logging and authentication, while lower-level concerns focus on programmatic concepts such as buffering and caching. The classification method described in this paper is applicable to the high and intermediate level concerns that are specified in the requirements specification or other early documents. Lower level concerns that belong to the solution domain at the programmatic or design level are not discussed further. Early discovery of aspects is significant not only for purposes of architectural design, but also so that candidate aspects can be evaluated and modeled in the design and code of the system, thereby minimizing the need to 'mine' and refactor aspects from the code at a later date.

The NFR-Classifier uses information retrieval methods to find and identify NFRs. The method assumes that different types of NFR are characterized by the use of relatively distinct keywords that we call 'indicator terms'. When those indicator terms are learned for a specific NFR type, they can be used to detect requirements, sentences, or phrases, related to that type. The process, which is depicted in Figure 1, includes the three primary phases of *mining, classification,* and *application*. During the

14th IEEE International Requirements Engineering Conference (RE'06)
0-7695-2555-5/06 $20.00 © 2006 IEEE

training phase indicator terms are mined from existing requirements specifications in which NFRs have been manually categorized by type. These terms are then used during the retrieval phase to detect and classify other NFRs. Finally during the application phase, the classified requirements are used to support more advanced software engineering activities such as requirements negotiation or architectural design.

The remainder of this paper is structured as follows. Section 2 surveys existing methods for eliciting and discovering non-functional requirements and early aspects. This section also describes a preliminary experiment we conducted using a fixed key-term approach. Section 3 introduces the new NFR-Classifier method, and describes the process for mining terms and using them to classify NFRs. Section 4 reports on an experiment we conducted to evaluate the effectiveness of the approach using fifteen different requirements specifications constructed as term projects by MS students at DePaul University. Section 5 describes an industrial case study based on a large user requirement document developed for a project at Siemens Logistics and Automation plant. In this study, the classification results obtained from reusing previously mined indicator terms are compared to results obtained from retraining the tool. The observed positive association between the performance of our classifier and the training set size is discussed in Section 6, and Section 7 concludes with a discussion of the application of this approach to the requirements analysis process, and suggestions for future work.

2. Existing NFR classification methods

There are two primary approaches to NFR classification. These include elicitation methods that support stakeholders as they reason about, identify, negotiate, and model NFRs; and also detection methods for semi-automated or manual extraction of NFRs from a variety of existing documents.

2.1 Elicitation techniques

Elicitation methods frequently rely upon creative brainstorming or the use of checklists and NFR templates to trigger stakeholders' input. For example Win-Win methods [12] provide generic checklists and require stakeholders to contribute, prioritize, and negotiate NFRs perceived to be important to the success of the system. The Architectural Assessment Method (ATAM) models NFRs using utility trees in which stakeholders describe quality requirements within a hierarchical abstraction of high-level goals [13]. The NFR framework provides catalogs to help analysts define NFR quality goals, potential implementation solutions, and to identify conflicts [6]. All of these techniques provide stakeholders with a structured approach for brainstorming and documenting their NFR needs and also produce a set of categorized NFRs. In these cases, the NFR-Classifier can provide input into the process by retrieving and displaying NFR related comments made by stakeholders and recorded in documents produced during earlier elicitation activities.

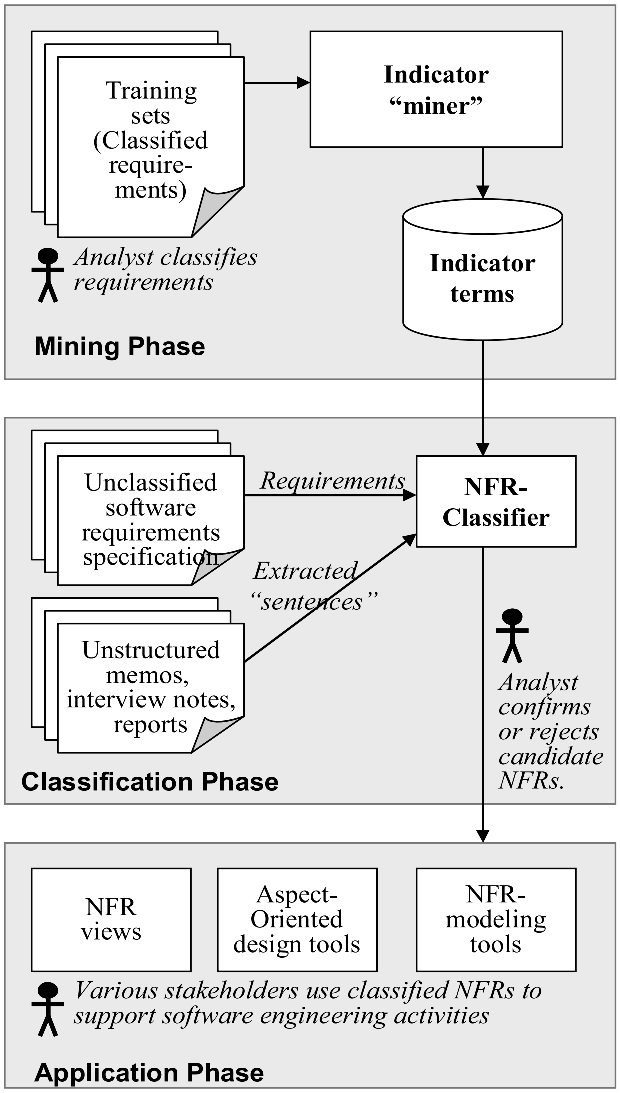
Training sets (Classified require-ments)
Indicator "miner"
Indicator terms
Analyst classifies requirements
Mining Phase
Unclassified software requirements specification
Requirements
NFR-Classifier
Extracted "sentences"
Unstructured memos, interview notes, reports
Analyst confirms or rejects candidate NFRs.
Classification Phase
NFR views
Aspect-Oriented design tools
NFR-modeling tools
Various stakeholders use classified NFRs to support software engineering activities
Application Phase

Figure 1. The NFR classification process

2.2 Detection techniques

With the increasing popularity of Aspect-Oriented Programming (AOP), several researchers have developed techniques for detecting low-level aspects in the design and code and 'early-aspects' from requirements

Table 1. Results from fixed keyword retrieval and classification

NFR Type	Keywords extracted from a SIG Catalog [6]	Recall	Precision	Specificity
Security	Confidentiality, integrity, completeness, accuracy, perturbation, virus, access, authorization, rule, validation, audit, biometrics, card, key, password, alarm, encryption, noise.	0.798	0.567	0.871
Performance	space, time, memory, storage, response, throughput, peak, mean, index, compress, uncompress, runtime, perform, execute, dynamic, offset, reduce, fixing, early, late	0.609	0.396	0.843

specifications [14]. At the program level, aspects can be 'mined' using clone detection techniques such as pattern matching against the abstract syntax tree, or through analyzing the system's meta-model [5]. Runtime methods such as the analysis of execution traces [19] can also be used at the code level. None of these approaches are applicable to the detection of early aspects, including NFRs, which are less formally expressed and exist prior to the system becoming executable.

However, several semi-automated techniques have been proposed for mining early aspects. Rosenhainer proposed a basic information retrieval (IR) method that required an analyst to manually search through the requirements looking for candidate aspects [16]. If a candidate aspect were found, then it was used as the basis of an IR style query to find related requirements. Several researchers have described effective methods for implementing such artifact based searches [2,7,8,11,18]. However Rosenhainer's approach is labor intensive as it requires manual inspection to discover a 'starting point' for the analysis, and it also depends on finding a good starting requirement that contains similar terms to other aspect-related requirements.

The Theme/Doc method [3] also provides semi-automated support for early aspect mining. An analyst parses the requirements specification to identify keywords, which are then used by the tool to generate a visual representation of the relationships between behaviors. This view is used by the analyst to identify candidate aspects. Again, the approach is rather labor intensive as the analyst needs to perform a preliminary manual search for keywords in addition to a later analysis of the candidate concerns. It also presupposes that the requirements specification is grammatically structured in a certain way. However Theme/Doc provides the opportunity to discover aspect types that are unique to a specific project in addition to commonly occurring types.

Sampaio et al proposed a method that uses natural language processing to first identify viewpoints based on nouns occurring in the specification and then to find actions (ie verbs) that occur across multiple viewpoints. This approach has potential for identifying unique aspect types, but it requires significant user feedback to evaluate viewpoints and assess the feasibility of the candidate aspects. It may also miss aspects if different action verbs are used to represent the same concern across different viewpoints [17].

Although these techniques for early aspect identification are applicable to the problem of NFR classification, they all require significant user involvement. This suggests the need to find a more automated approach for identifying NFRs.

2.3 Keyword classification method

As a precursor to our work on the NFR-Classifier we investigated whether a pre-defined fixed set of keywords could be used to classify each type of NFR. This simpler approach would avoid the need to develop and use a training set. A small experiment was conducted in which a set of keywords, listed in Table 1, were extracted from catalogs of operationalization methods for security and performance softgoal interdependency graphs (SIGs) [6]. These catalogs represent extensive bodies of knowledge related to goals and potential solutions for each of these NFRs and so provided a standardized set of keywords. The keywords were used to retrieve NFRs from a set of fifteen requirements specifications developed by DePaul MS students as term projects for a course in Requirements Engineering[1]. Any requirement containing one or more of these keywords was classified as a candidate security or

[1] Approximately 80% of the students in this course work in the software industry as professionals. Out of the 45 students in the class the top fifteen projects were selected for this experiment, based primarily upon grades assigned by the course instructor. The projects represented a broad range of topics. The high ratio of NFRs to functional requirements reflects the time limitations that inhibited the writing of a more complete set of requirements. Requirements were specified using the Volere template http://www.systemsguild.com/

14th IEEE International Requirements Engineering Conference (RE'06)
0-7695-2555-5/06 $20.00 © 2006 IEEE

performance requirement respectively. A multiple categorization approach was taken in which a requirement containing both a security keyword and a performance keyword was classified into both categories.

Results were evaluated for each NFR type using the standard metrics of recall, precision, and specificity where recall measures the number of NFRs that were correctly retrieved and categorized; precision measures the total number of correctly retrieved NFRs in respect to the total number of retrieved NFRs [10], and specificity measures the ability of the classifier to correctly reject requirements that are not of the right NFR type [1]. The results of this experiment are shown in Table 1 and indicate that the security keywords returned a recall of 80% and precision of 57%, while the performance keywords returned a recall of 61% and 40% precision. Observation showed that many of the security keywords were in fact shared by other types of NFRs, and that several of the target NFRs did not contain any of the keywords and so were not retrieved. One of the primary stumbling blocks of this approach (and the reason that we did not evaluate more NFRs using this method) was the difficulty of finding accepted and standardized catalogs for the other types of NFRs specified in our data sets.

3. The NFR-Classifier

The NFR-Classifier addresses this problem by using a training set to discover a set of weighted indicator terms for each NFR type. This approach means that the NFR-Classifier is limited to recognizing and retrieving NFR types for which it has been trained. However this is not overly limiting. Although more than one-hundred and fifty NFR types and numerous lower level aspects have been documented [6], in practice a much smaller subset of common ones are generally of interest during the system design process. The approach also has several benefits over the standard keyword method. Indicator terms can be automatically mined from existing pre-categorized requirement specifications, and therefore customized for an organization in accordance with their own standard terminologies and policies.

The NFR-classifier method consists of two stages. During the first stage, a set of indicator terms is identified for each NFR category. This step assumes the existence of a set of correctly pre-classified requirements that can be used for training. The requirements in the training set are used to compute a probabilistic weight for each potential indicator term in respect to each NFR type. The weight measures how strongly an indicator term represents a requirement type. For example, terms such as “authenticate” and “access” that occur frequently in security requirements and infrequently in other types of requirements, represent strong indicator terms for security NFRs, while other terms such as ‘ensure’ that occur less frequently in security requirements or are found in several different requirement types, represent much weaker indicators.

Once indicator terms are mined and weighted, they can be used in a second step to classify additional requirements. A probability value that represents the likelihood that the new requirement belongs to a certain NFR type is computed as a function of the occurrence of indicator terms of that type in the requirement. A requirement is then classified according to a certain NFR type if it contains several indicator terms representative of that type. Requirements receiving classification scores above a certain threshold for a given NFR type will be classified into that type, and all unclassified requirements will be assumed to be functional requirements. Because classification results can only be considered successful if a high percentage of the target NFRs are detected for a specific type, in all of the experiments described in this paper the threshold was established with the objective of achieving high recall results.

Prior to classification, the requirements must be preprocessed and reduced to a set of keywords [10]. The preprocessing step first eliminates all common “stop” words that do not provide any relevant information on the document lexical content (for example conjunctions and prepositions). The remaining words are then reduced to their stemmed form, by eliminating plurals, past tenses, and other suffixes. The following sections more formally describe the two steps of mining and classification.

3.1 Indicator terms mining

Let Q be a given requirement type. Indicator terms of quality type Q are found by considering the set S_Q of all type Q NFRs in the training set. The cardinality of S_Q is defined as N_Q. The frequency $freq(d_{Q,i},t)$ of occurrence of term t in document $d_{Q,i}$ is computed for each document in S_Q. Each term is assigned a weight score that measures how well the term helps identify a requirement of quality type Q.

So for each type Q and keyword t, the weight score $Pr_Q(t)$ is defined as a probability value computed as follows:

$$\Pr_Q(t) = \frac{1}{N_Q}\sum_{i=1}^{N_Q}\frac{freq(d_{Q,i},t)}{|d_{Q,i}|}\cdot\frac{N_Q(t)}{N(t)}\cdot\frac{NP_Q(t)}{NP} \quad (1)$$

The first factor $\frac{1}{N_Q}\sum_{i=1}^{N_Q}\frac{freq(d_{Q,i},t)}{|d_{Q,i}|}$ in the expression above computes the average frequency of occurrences of term t in type Q NFRs rescaled by the documents size $|d_{Q,i}|$. The factor increases if a term appears more frequently in type Q documents, and therefore it can be

Table 2: Counts of requirement specifications by project and quality type

	Project Number															
Quality type	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**TOTAL**
Availability	1	1	1	0	2	1	0	5	1	1	1	1	1	1	1	**18**
LookAndFeel	1	2	0	1	3	2	0	6	0	7	2	2	4	3	2	**35**
Legal	0	0	0	3	3	0	1	3	0	0	0	0	0	0	0	**10**
Maintainability	0	0	0	0	0	3	0	2	1	0	1	3	2	2	2	**16**
Operational	0	0	6	6	10	15	3	9	2	0	0	2	2	3	3	**61**
Performance	2	3	1	2	4	1	2	17	4	4	1	5	0	1	1	**48**
Scalability	0	1	3	0	3	4	0	4	0	0	0	1	2	0	0	**18**
Security	1	3	6	6	7	5	2	15	0	1	3	3	2	2	2	**58**
Usability	3	5	4	4	5	13	0	10	0	2	2	3	6	4	1	**62**
TOTAL NFRs	8	15	21	21	37	44	8	71	8	15	10	20	19	16	12	326
Functional	*20*	*11*	*47*	*25*	*36*	*26*	*15*	*20*	*16*	*38*	*22*	*13*	*3*	*51*	*15*	*358*
TOTAL	28	26	68	47	73	70	23	91	24	53	32	33	22	67	127	684

considered as a potential indicator term for type *Q* NFRs. The second factor $\frac{N_Q(t)}{N(t)}$ is the percentage of *Q* type documents in S_Q containing *t* with respect to all requirements in the training set containing *t*, whose number is denoted by *N(t)*. This factor decreases if the indicator term *t* is used broadly throughout the requirements specification. If the term is only used in *Q* type requirements, it will evaluate to 1 for that type. The third factor $\frac{NP_Q(t)}{NP_Q}$ is the ratio between the number $NP_Q(t)$ of system projects containing type Q documents with term *t* and the number NP_Q of all projects in the training set with type Q NFRs. The purpose of this rescaling factor is to decrease the weight $Pr_Q(t)$ for terms that are project specific, and to increase the weight for terms that appear in several projects containing type *Q* documents.

For each term *t*, a probability score $Pr_Q(t)$ is computed. Terms are ranked by decreasing order according to $Pr_Q(t)$. The top K terms are identified as indicator terms for the type *Q*. Experiments to evaluate and select the best value for K are presented in Section 4.

3.2 NFR classification

A classification algorithm for NFRs is defined by computing a probability score $Pr_Q(R)$ that evaluates the probability that a certain NFR *R* is classified as *Q* type. This probability score depends on the lexical content of requirement *R*. We assume that type Q NFRs are more likely to contain indicator terms for that type.

Let I_Q be the set of indicator terms for a quality type Q. We assume that the weighted indicator terms in I_Q are identified and their weights computed from a training set that contains correctly pre-categorized NFRs. The indicator terms are mined using the expression in (1).

The classification score that an unclassified requirement R belongs to a type Q is defined as follows:

$$\Pr_Q(R) = \frac{\sum_{t \in R \cap I_Q} \Pr_Q(t)}{\sum_{t \in I_Q} \Pr_Q(t)} \tag{2}$$

The numerator is computed as the sum of the term weights of all type Q indicator terms that are contained in R, and the denominator is the sum of the term weights for all type Q indicator terms. The probabilistic classifier for a given type Q will assign higher score $\Pr_Q(R)$ to an NFR R that contains several strong indicator terms for Q. Results on the application of our classifier are reported below.

4. Evaluating the classifier model

The training set used in the experiments again consisted of the 15 requirements specifications developed as term projects by MS students at DePaul University. These specifications contained a total of 326 NFRs and 358 functional requirements. NFR types included availability, look-and-feel, legal, maintainability, operational, performance, scalability, security, and usability. As there were insufficient portability and process requirements, these NFR types were not included in the study. Counts for each requirement type are displayed in Table 2, while Table 3 depicts the top 15 indicator terms that were mined from each NFR type.

Table 3. Top 15 indicator terms 'mined' from the training set

Rank	Availability	Legal	Look & Feel	Maintain-ability	Opera-tional	Perform-ance	Scalability	Security	Usability
1	avail	compli	appear	updat	interfac	second	simultan	onli	us
2	achiev	regul	interfac	mainten	environ	respons	handl	access	easi
3	dai	standard	profession	releas	server	time	year	author	user
4	time	sarban	appeal	new	oper	longer	capabl	user	train
5	hour	oxlei	color	chang	product	fast	support	inform	product
6	pm	php	look	dure	system	minut	expect	ensur	abl
7	year	pear	simul	promot	databas	take	concurr	data	understand
8	technic	legal	product	product	browser	process	abl	authent	successfulli
9	downtim	law	compli	addit	window	user	number	secur	intuit
10	long	estimat	scheme	everi	web	system	user	system	learn
11	system	regard	logo	budget	comput	let	launch	malici	system
12	product	complianc	sound	develop	applic	maximum	process	prevent	click
13	seven	rule	brand	season	us	complet	next	incorrect	minut
14	defect	requir	feel	integr	internet	flow	product	product	self
15	asid	izognmovi	sea	oper	abl	everi	connect	ar	explanatori

4.1 Classifying the NFRs

To evaluate the effectiveness of the NFR-Classifier, a leave-one-out cross validation technique was applied against the fifteen software requirements specifications (SRS). Fifteen iterations were conducted. During each iteration indicator terms were extracted from fourteen SRS's that constituted the training set, and term weights for each NFR type were calculated based on the function in (1). Two alternate methods were evaluated for selecting indicator terms from the training set:

1) **Top K terms**, where K is a positive number. For each NFR type, the K terms with the highest weights were selected as the indicator terms.
2) **All terms**, where every term with a non-zero weight with regard to a specific NFR type was selected as an indicator term for that type.

The extracted indicator terms were then used to classify requirements in the remaining SRS using the function in (2). A multiple classification scheme was used so that for any given NFR type, all requirements that scored higher than a certain threshold value were classified as that NFR type. This meant that a single requirement could be classified into more than one NFR type, although implicitly if it represented an atomic requirement, it could only logically belong to one type. As a side note, an additional experiment was conducted to compare the efficacy of multiple classification versus a "pick-top" method in which each requirement was assigned to only the NFR type for which it received the highest classification value. When the pick-top method was used, recall was problematic for almost all categories. For example, recall dropped to 33% for legal, 9% for look-and-feel, and 41% for maintainability. The average recall obtained using this method was only 52% as opposed to 76% using the multi-category method. Based on these results, the multiple-classification approach was adopted.

The goodness of the classifiers was evaluated by the three metrics of recall, precision and specificity for each NFR type in each single iteration of the experiment. Overall results were calculated by combining the results of the fifteen iterations. As each SRS was subjected to classification in only one experiment, the combined results therefore represented the classification of each individual requirement only one time.

4.2 Selection of indicator terms

Three different values for K were used in our experiments: K=5, K=10 and K=15. Table 4 shows the overall recall and precision of the classification using

Table 4. Comparison of top-K versus "All" terms extraction methods.

Indicator Selection Method	*Recall*	*Precision*
Top-5	0.6564	0.3242
Top-10	0.7423	0.2400
Top-15	0.7669	0.2480
ALL	0.7392	0.2720

Table 5. Confusion matrix of the classification

Actual	Total#	Classified as								
		A	L	LF	MN	O	PE	SC	SE	US
Availability	18	**16**	0	3	6	10	10	6	4	10
Legal	10	0	**7**	2	0	3	1	0	1	5
Look-and-feel	35	0	7	**18**	9	20	0	1	9	22
Maintainability	17	8	0	5	**15**	11	4	5	3	10
Operational	61	10	0	32	10	**44**	3	11	17	42
Performance	48	20	1	7	7	27	**30**	12	20	35
Scalability	18	11	0	5	4	12	3	**13**	6	11
Security	57	6	3	10	12	38	5	8	**46**	38
Usability	62	13	4	26	7	34	14	24	35	**61**

Top-5, Top-10, Top-15, and 'all' indicator terms respectively. A classification threshold of 0.04 was maintained for all four of the experiments, meaning that only requirements that were given scores greater than 0.04 for a particular NFR type were classified.

The results indicated that among the Top-K methods, Top-5 returned the worst recall, about 10% lower than the other two methods. The classification accuracy showed no significant difference between the Top-10 and Top-15 methods. Precision remained at about 24% while recall improved slightly from 74.23% to 76.69% when changing the selection method from using Top-10 to Top-15 terms. The results also indicated that if all terms were retained as indicators then the recall was reduced by 2.7%, while precision was increased by 2.4% compared to the Top-15 method. As recall and precision tend to trade-off against each other, this difference was again considered relatively insignificant. If the threshold had been slightly raised to remove some of the extra 'background noise' caused by the additional less significant terms, then recall would have likely decreased and precision increased to levels similar to Top-15.

Surprisingly, there was therefore no significant difference between the use of Top-10, Top-15, and "all" terms approach, however it is possible that the smaller set of terms may be over-fit to the training set and may be less applicable when used to classify requirements from different sources.

4.3 Classification results

A confusion matrix is a useful instrument for analyzing classification results [9], as it has the ability to depict true and false positives as well as true and false negatives. Table 5 illustrates the confusion matrix for the classification results for Top-15.

The correct classifications (true positives) are depicted on the diagonal, and have been highlighted in the diagram. For example, the matrix shows that 16 availability NFRs were correctly categorized. By looking in the column labeled "A" for availability we also see that 8 maintainability requirements were incorrectly classified as availability. Additionally operational, performance, scalability, performance, and usability requirements were also similarly incorrectly classified. Looking across the row labeled "Availability" we also see that availability requirements were incorrectly classified under several NFR types including three as 'look-and-feel', and six as 'maintainability'. Although there are only 18 actual availability requirements, the multiple classification approach means that many of them will be classified more than once.

Table 6 reports the three metrics of recall, precision, and specificity that were derived from the confusion matrix for each NFR type. Recall is computed as $recall = \frac{TruePositives}{TruePositives+FalseNegatives}$, while precision is defined as $precision = \frac{TruePositives}{TruePositives+FalsePositives}$. Specificity in binary classification is the proportion of true negatives to all negatives [1] and is computed as

Table 6. Results using top-15 terms at classification threshold value of 0.04

NFR Type	Recall	Precision	Specificity
Availability	0.8889	0.2462	0.7792
Legal	0.7000	0.3684	0.9525
Look-and-feel	0.5143	0.2093	0.6907
Maintainability	0.8824	0.2459	0.8220
Operational	0.7213	0.2604	0.4151
Performance	0.6250	0.1887	0.8561
Scalability	0.7222	0.2000	0.7825
Security	0.8070	0.2771	0.6468
Usability	0.9839	0.2798	0.3447
Overall	0.7669	0.2480	

$specificity = \frac{TrueNegatives}{TrueNegatives + FalsePositives}$. Here the specificity for a specific NFR type, say Availability, is computed as the proportion of all non- Availability NFR types that are not misclassified as Availability to the total number of non-Availability NFRs. A high specificity of a NFR type indicates the ability of the classifier to correctly differentiate this NFR from others.

As depicted in Table 6, different NFR types responded differently to the classification method. Usability had the highest recall of 98.39%, and the confusion matrix shows that only 1 out of the 62 usability NFRs were misclassified or unclassified. However, usability NFR also had the lowest specificity of 34.47%, meaning it was quite difficult to differentiate non-usability NFRs from usability ones. As shown in the last column of the confusion matrix, there are a total of 173 non-usability NFRs that have been misclassified as usability. In contrast look-and-feel achieves the lowest recall of 51.43% and a specificity of 69.07%.

4.4 Analysis of results

In general these results suggest that the NFR-Classifier can effectively detect several different types of NFRs, but that additional work is needed to improve results for certain NFR types such as 'look-and-feel.' It was observed for this NFR type, that categories of words such as colors tended to occur across multiple requirements, and future work will therefore investigate the possibility of using categories of indicator terms or extended training to improve these retrieval results.

5. Industrial case study

As an initial proof-of-concept the indicator terms mined from the fifteen projects were used to detect and classify candidate NFRs from a word document describing the customer requirements for an integrated engineering toolset (IET) under development at Siemens Logistics and Automation plant. The IET document contained 137 pages, 2,250 paragraphs, and 30,374 words. To classify the NFRs in the document, it was first saved as a text file, parsed to remove unwanted characters, and then deconstructed into 2,064 "sentences." These sentences were not necessarily grammatically complete, as they included bullet points, and text extracted from tables etc. Some sentences corresponded to actual requirements in the text and others to less structured narrative. The data was treated to remove stop words and stem terms to their roots and was then parsed by the NFR classifier. In addition to automated classification, all of the sentences were manually classified into NFR types in order to create an 'answer' set against which classification results could be compared. The counts for each NFR type are depicted in Table 7.

5.1 Fixed key words

In the first experiment, the fixed keywords shown in Table 1 were used to classify security and performance NFRs. Security NFRs were retrieved with recall of 58%, precision of 30%, and specificity of 89%, while performance NFRs were retrieved with recall of 35%, precision of 22%, and specificity of 92%. These results strengthened our earlier conjecture that the use of this fixed set of keywords did not consistently produce good classification results.

5.2 Using prior indicator terms

In the second experiment, the terms extracted from the 15 original SRS's were used. Both "Top-15" and "all" indicator term approaches were evaluated, however no significant differences were observed in recall and precision metrics. Results from the "all" indicator approach which showed slightly higher recall, are shown in Table 7. Results using these previously mined indicator terms were relatively good for availability, security, and usability, which all had recall values around 80%, but were disappointing for several of the other NFR types. For example look-and-feel and performance NFRs were retrieved only at recall levels of approximately 33%, while legal, operational, and scalability requirements also returned relatively low recall values.

An analysis of the targeted NFRs in the user requirements document revealed a mismatch of terms between the MS Projects and the IET data. For example, performance NFRs in the high-level IET document tended to use more general terms such as "fast" and "quickly", compared to the much more precise use of terms such as "per second" in the MS project training sets. This mismatch explained why indicator terms mined from the original training sets had not been effective in classifying certain types of NFR in the new data set.

5.3 Retraining the classifier

Because of the poor classification results from the first two experiments we decided to conduct a third study in which the NFR-Classifier was retrained using a training set composed of one third of the sentences in the IET document. The training set was used to mine new indicator terms which were then used to classify NFRs in the remaining two thirds of the document. The fraction of "one third" was selected to provide sufficient NFRs in each type for training purposes. One third of the requirements from each requirement type, including functional requirements were randomly selected for the training set. Results obtained using the new indicator terms were generally much improved. All of the availability requirements were recalled; operational, security, and usability NFRs were recalled at relatively high values ranging from 73 - 87%; and only the NFR type of look-and-feel performed badly with a recall of 40%. In fact even this was a significant improvement from the previous recall value of 13%. There were insufficient scalability requirements in the IET document to train the tool to recognize this type of NFR. Overall, recall rose from 62.9% to 79.9% and precision also increased from 35.2% to 42.5%.

The results from this initial study suggested that retraining the NFR-classifier using a training set that is 'closer' to the data being classified significantly improved the results. Additional research is needed to better understand whether a set of indicator terms built from a much larger training set would be applicable across a broader range of projects, or if a hybrid approach could be used which takes as a starting point a set of standard indicator terms and then enhances and refines the terms during a training session. In this case a hybrid approach would have averted the problem of having insufficient scalability requirements in the new training set.

6. Training set size

In our experiments, we noticed that recall values tended to be higher for NFR types that were better represented in the training set. For NFR types whose relative size in the training set was equal to 7% or higher, the recall values increased with the number of pre-classified NFRs of those types used to train the classifier. When we considered only those NFR types with a higher number of requirements in the training set, the correlation value between recall and NFR relative size was equal to 0.723 for the MS project data and was equal to 0.833 for the IET data, showing a strong positive association.

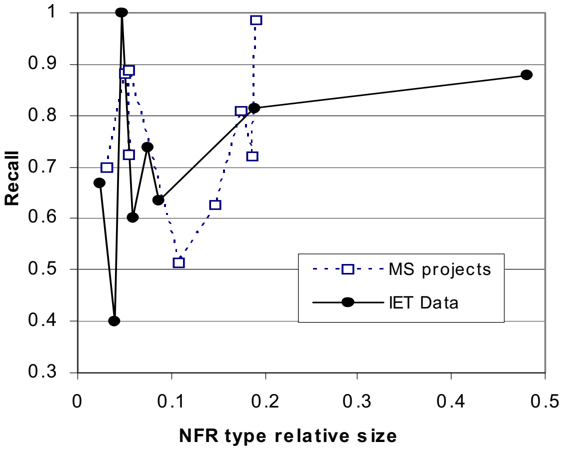

Figure 2. Possible correlation between training set size and recall

Although these results do not prove the positive association between recall and training set size to be true in general, it seems intuitively correct that the NFR classifier is expected to perform better for NFR types that have a higher number of requirements in the training set.

7. Conclusions

This paper has introduced a new approach based on information retrieval methods for detecting and classifying non functional requirements from both structured requirements specifications as well as free-form text. Although it still requires an analyst to evaluate the correctness of candidate NFRs, it requires less effort than previous semi-automated classification methods such as the Theme/Doc method [3]. Having the ability to classify NFRs in this way is useful for software engineers as they design, construct, and analyze a software system because it can help them to understand stakeholders' needs and to

Table 7. Results from retrieving and classifying NFRs from IET requirements document

QualityType	Using "all" indicator terms mined from 15 MS projects				Using indicator terms mined from 30% of IET Data			
	NFR count by Type	Recall	Precision	Specificity	NFR count by Type*	Recall	Precision	Specificity
Availability	18	0.917	0.550	0.963	12	1.000	0.261	0.860
Legal	9	0.333	0.182	0.964	6	0.667	0.235	0.948
Look & Feel	15	0.300	0.059	0.804	10	0.400	0.182	0.926
Maintainability	33	0.591	0.394	0.914	22	0.636	0.203	0.763
Operability	73	0.333	0.235	0.749	48	0.813	0.488	0.801
Performance	23	0.200	0.375	0.979	15	0.600	0.474	0.958
Scalability	2	1.000	0.030	0.874	2	Insufficient data to mine terms		
Security	29	0.895	0.134	0.534	19	0.737	0.341	0.885
Usability	183	0.820	0.730	0.722	122	0.877	0.618	0.500
All NFRs*		**0.599**	**0.299**			**0.799**	**0.435**	

** Reflects the number of NFRs remaining after the training set was extracted.*

see cohesive views of various system constraints. Candidate NFRs retrieved from stakeholders' interviews or meeting minutes, can be detected and classified by the tool and then used as input into architectural design meetings or retrieved by a security engineer as he or she analyzes specific security concerns. Candidate aspects can be evaluated early in the design process, saving costs that would otherwise have been incurred in later refactoring efforts. For example, a tool currently under development at DePaul University incorporates NFR detection into a collaborative tool for modeling system-wide non-functional goals. The viewpoints provided by the NFR tool provide useful inputs into the modeling process.

This paper has provided an initial validation of the approach but additional work needs to be conducted to answer critical questions such as whether a larger training set might improve consistency of classification results across NFR types and across different projects and organizations. It will also be interesting to investigate under what conditions retraining is necessary and whether the use of a hybrid approach incorporating standard keywords, previously mined terms, and retraining of the tool within the current context could improve results. The examples in this paper focused on classifying NFRs and high-level aspects, but future work is needed to evaluate the technique against intermediate level aspects such as logging and authentication. These open issues will be examined in our ongoing work.

Acknowledgments

The work described in this paper was partially funded by NSF grants CCR- 0306303 and CCR-0447594. Additional funding and access to project artifacts was provided by Siemens Corporate Research and Siemens Logistics and Automation plant in Grand Rapids, MI.

References

[1] D.G. Altman and J.M. Bland,, "Statistics notes: diagnostic tests 1: sensitivity and specificity", *British Medical Journal*, 308, 1552. 1994.

[2] G. Antoniol, G. Canfora, G. Casazza, A. De Lucia and E. Merlo. "Recovering Traceability Links between Code and Documentation", *IEEE Transactions on Software Engineering*. Vol. 28, No. 10, 2002, pp. 970-983.

[3] E. Baniassad and S. Clarke. "Finding Aspects in Requirements with Theme/Doc", In Proceedings of Early Aspects 2004: Aspect-Oriented Requirements Engineering and Architecture Design, Lancaster, UK, 22 Mar. 2004.

[4] E. Baniassan, P. Clements, J. Araujo, A. Moreira, A. Rashid, and B. Tekinerdogan, "Discovering Early Aspects", *IEEE Software,* Vol 23, No 1, Jan/Feb 2006.

[5] M. Bruntink, A. van Deursen, T. Tourwe, R. van Engelen, "An Evaluation of Clone Detection Techniques for Identifying Crosscutting Concerns" , *20th Intn'l Conference on Software Maintenance (ICSM'04),* Chicago, Sept. 2004, pp. 200-209.

[6] L. Chung, B. Nixon, E. Yu, and J. Mylopoulos, *Non-Functional Requirements in Software Engineering*, Kluwer Academic, 2000.

[7] J. Cleland-Huang, R. Settimi, C. Duan, X. Zou, "Utilizing Supporting Evidence to Improve Dynamic Requirements Traceability", *International Requirements Engineering Conference,* Paris, France, Aug/Sept, 2005.

[8] J. Cleland-Huang, R. Settimi, O. BenKhadra, E. Berezhanskaya, S. Christina, "Goal-Centric Traceability for Managing Non-Functional Requirements", International Conference on Software Engineering, St. Louis, USA, May 2005. pp. 362-371.

[9] T. Fawcett. "ROC Graphs: Notes and Practical Considerations for Researchers". HP Labs Tech Report HPL-2003-4.

[10] W.B. Frakes and R. Baeza-Yates, *Information retrieval: Data structures and Algorithms*, Prentice-Hall, Englewood Cliffs, NJ, 1992

[11] J. Huffman Hayes, A. Dekhtyar, S. K. Sundaram, "Advancing Candidate Link Generation for Requirements Tracing: The Study of Methods", *IEEE Transactions on Software Engineering,* Vol. 32, No. 1, January 2006. pp. 4-19.

[12] H. In and B. W. Boehm, "Using WinWin Quality Requirements Management Tools: A Case Study". Annals Software Eng. 11(1): 141-174 (2001)

[13] R. Kazman, M. Klein, and P. Clements, "ATAM: Method for Architecture Evaluation", *CMU/SEI Technical Report, CMU/SEI-2000-TR-004, ADA382629*, Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2000. Available online at:http://www.sei.cmu.edu/publications/ documents/00.reports /00tr004.html.

[14] A. Kellens and K. Mens, "A Survey of Aspect Mining Tools and Techniques", INGI Technical Report 2005-08, UCL, Belgium, Deliverable 6.2a for the workpackage 6 of the IWT project 040116 "AspectLab". June 2005.

[15]B. Nuseibeh, "Weaving Together Requirements and Architecture", *IEEE Computer*, Vol. 34, No. 3, March 2001, pp. 115-117.

[16] L. Rosenhainer, "Identifying Crosscutting Concerns in Requirements Specifications", *Workshop on Early Aspects: Aspect-Oriented Requirements Engineering and Architecture Design*, 2004, Vancouver, Canada, Oct. 2004. http://trese.cs.utwente.nl/Docs/workshops/oopsla-early-aspects-2004/http://trese.cs. utwente.nl/Docs/workshops/oopsla-early-aspects-2004/

[17] A. Sampaio, N.Loughran, A. Rashid, P. Rayson (2005) *Mining Aspects in Requirements*. Workshop on Early Aspects (held with AOSD 2005).

[18] R. Settimi, J. Cleland-Huang, O. BenKhadra, J. Mody, W. Lukasik, and C. DePalma, C., "Supporting Change in Evolving Software Systems through Dynamic Traces to UML", *IEEE International Workshop on Principles of Software Evolution*, Kyoto, Japan, (Sept. 2004), 49-54.

[19] P. Tonella and M. Ceccato, "Aspect Mining through the Formal Concept Analysis of Execution Traces", *11th Working Conference on Reverse Engineering (WCRE'04),* Netherlands, Nov. 2004, pp. 112-121.