PREREQIR: Recovering Pre-Requirements via Cluster Analysis

Jane Huffman Hayes
Dept. of Computer Science
University of Kentucky
hayes@cs.uky.edu

Giuliano Antoniol
Dépt. de Génie Informatique
École Polytechnique de Montréal
antoniol@ieee.org

Yann-Gaël Guéhéneuc
Dept. of Computer Science
Université de Montréal
guehene@iro.umontreal.ca

Abstract

High-level software artifacts, such as requirements, domain-specific requirements, and so on, are an important source of information that is often neglected during the reverse- and re-engineering processes. We posit that domain specific pre-requirements information (PRI) can be obtained by eliciting the stakeholders' understanding of generic systems or domains. We discuss the semi-automatic recovery of domain-specific PRI that can then be used during reverse- and re-engineering, for example, to recover traceability links or to assess the degree of obsolescence of a system with respect to competing systems and the clients' expectations. We present a method using partition around medoids and agglomerative clustering for obtaining, structuring, analyzing, and labeling textual PRI from a group of diverse stakeholders. We validate our method using PRI for the development of a generic Web browser provided by 22 different stakeholders. We show that, for a similarity threshold of about 0.36, about 55% of the PRI were common to two or more stakeholders and 42% were outliers. We automatically label the common and outlier PRI (82% correctly labeled), and obtain 74% accuracy for the similarity threshold of 0.36 (78% for a threshold of 0.5). We assess the recall and precision of the method, and compare the labeled PRI to a generic Web browser requirements specification.

1. Introduction

Software systems change. Their evolution must be carefully managed and must be intimately related to high-level software artifacts such as requirements, analysis models, and design models. Yet, as systems evolve, the high-level artifacts are not updated and the source code often becomes the sole reliable source of system information.

High-level software artifacts, such as requirements and information available prior to requirement specification, are vital not only to successful development [5] but also to successful maintenance and evolution activities. In particular, we are interested in *pre-requirements information* (PRI) that include system concepts, user expectations, the environment of the system, etc. Examples of PRI for a generic word processor might include "be able to specify the language for spell checking," "run under Linux," and "not require conversion of existing files."

Consequently, in this paper, we address the problem of recovering PRI in a semi-automated and cost-effective way. Recovering PRI is difficult because it requires interactions with stakeholders and because information gathering is mostly performed via interviews that are time consuming and expensive. PRI structuring is labor-intensive and is traditionally performed manually.

Few research works address the problem of recovering, validating, or evolving PRI [31], despite the relevance of PRI and requirement documents during the reverse- and re-engineering processes. Indeed, PRI could help in: ensuring a common terminology among stakeholders; identifying reuse opportunities; assessing the degree of obsolescence of a system with respect to competing systems and the clients' expectations; recovering traceability links; adding new features; developing test cases; improving existing functionalities; or porting a system to a new paradigm or environment. This is true because all these activities require high-level documentation detailing implemented functionalities, domain concepts, explicit or implicit dependencies, and so on.

Therefore, we posit that much of the PRI for a particular domain or generic system exists in the mental models of the diverse stakeholders of the system. Mental models capture the stakeholders' understanding of the domain and the system to-be-built or evolved, including how it should work, stakeholder needs and expectations of the system functionality, usefulness, etc. The present work focuses on textual PRI.

Applying information retrieval (IR) techniques and data clustering, we propose *PREREQIR*, a method to obtain, structure, analyze, and label projections of stakeholders' domain-specific mental models in the form of PRI, interchangeably called *user needs*. We apply PREREQIR to the PRI obtained from 22 volunteer participants for a generic Web browser, a common software system used in different ways by many stakeholders. We show that the PREREQIR method helps us automatically obtain two sets of PRI for Web browsers: a set of common user needs; and a set of uncommon, or outlier, user needs. We compare the two sets of domain-specific user needs to a generic Web browser requirement specification. We conclude that the method allows us to recover, structure, analyze, and label the PRI document with good accuracy: recall and precision both in the 70% range when cluster similarity is 0.36 or higher.

The main contributions of this paper are as follows:

- Examines the problem of obtaining, semi-automatically structuring, analyzing, and labeling PRI using IR techniques and clustering,
- Reports on the PRI of 22 participants that is obtained, structured, analyzed, and labeled,
- Reports on the recall and precision of the method, and
- Applies the PRI labels resulting from the method in a traceability task.

We organize the paper as follows: Section 2 introduces some definitions and the problem. Section 3 describes our method. Section 4 illustrates our method via a case study. Section 5 describes an assessment of the accuracy of the method and its usefulness in a traceability task. Section 6 discusses related work. Section 7 concludes and presents some future work.

2. Definitions and Problem

Our work focuses on recovering and structuring textual PRI. In this section, we discuss mental models and PRI, our PREREQIR method, and specific techniques supporting the method.

2.1. Mental Models and PRI

"[A] domain is used to denote or group a set of systems or functional areas within systems that exhibit similar functionality. [21]" Within a domain, a requirement is a necessary capability or characteristic of a system. A generic requirement is a necessary capability or characteristic of any system in a domain. A generic or domain requirement specification is a collection of requirements that specify a domain.

In a typical development process, domain engineering and application engineering occur in parallel. To produce the requirements for a software system, one first performs domain analysis, which results in a domain model. Analysis is performed on the domain model. During this analysis, stakeholder requirements must be negotiated and consolidated into an agreed upon set of requirement specifications (RS) for evaluation and approval.

As the system evolves, the distance between the current implementation and the high-level documentation increases and negotiated RS may no longer reflect the actual system.

We are interested in the domain model or generic requirements specification, specific to a domain, not to a system, because we believe that stakeholders innately know what constitutes a word processor, a payroll system, etc. Although we cannot directly access the stakeholder mental models, we can access their textual projection or PRI. At any point in time, stakeholders' mental models constitute PRI because they express the user perception of what the system should do.

2.2. Problem Statement

A few years after deployment, the RS may be non-existent, incomplete, and/or outdated. The RS may no longer specify the needs of the stakeholders. In writing, recovering, or revising a RS, it is crucial to ensure that all stakeholders, *i.e.*, programmers, managers, testing team members, marketing personnel, and end users, share a common understanding of the system. Therefore, the RS should reflect the stakeholder PRI to ensure that the implementation, enhancement, or evolution of the system satisfies stakeholder expectations and needs.

Once obtained, PRI can be grouped into common and outlier user needs. PRI from similar stakeholders can be: **grouped**—such as all developers or all end users; **compared within groups**—how similar are the PRI of all end users? of all the testers?; **compared between groups**—how similar are the PRI of the average developer and the average end user?; and **labeled or tagged**. PRI can be used to build a generic or domain-specific RS that can then be used for validation [22, 23] or traceability recovery [2, 6, 8, 9, 10, 11, 16, 20, 30].

3. The PREREQIR Method

The challenge lies in recovering and processing the PRI. We propose the PREREQIR method, consisting of three steps: obtain and vet a list of PRI; structure PRI via cluster analysis; and analyze the clustered PRI.

3.1. Obtaining PRI

Most of the documentation that accompanies large software systems consists of free text documents expressed in a natural language. (PRI in other forms than text could be transcribed as text.) Examples include requirements and design documents, user manuals, logs of errors, maintenance journals, and also annotations of individual programmers and teams. Even when semi-formal models are used, free text is essential to add semantic and context information. Therefore, it is natural to ask stakeholders to provide PRI in the form of free text.

We believe that a minimally-biased and minimally-intrusive way to obtain PRI from stakeholders is through a single inquiry. The inquiry should not be interactive, but rather issued via an anonymous questionnaire/Web form to ensure that the stakeholders are not influenced by the researchers.

3.2. Structuring PRI

Once obtained via questionnaire, the PRI must be analyzed and structured. We map the textual fragments composing the stakeholder PRI into a vector space via stopping, stemming, and dictionary building. Singular text fragments are then compared and grouped based on distance and then by using a similarity threshold. Standard vector space retrieval is used [3] to study the presence or absence of a strong structure among the PRI. If there is a weak separation among the groups, *i.e.*, many clusters that contain few PRI, then we investigate the presence of a hierarchical structure using an agglomerative nesting algorithm.

Decomposing with Vector Space. An individual textual PRI or user need, for example "the user shall be able to hear audio files such as .wav," is viewed as a query for which we search in the collection of all other PRI, also known as the document collection.

We define $V = \{k_1,\ldots,k_N\}$ as the vocabulary or list of keywords of a given document collection. A vector model of document d is a vector $(w_1,\ldots,w_N)$ of keyword weights where w_i is computed as $w_i = tf_i(d) \bullet idf_i$. $tf_i(d)$ is the term frequency or frequency of keyword k_i in the document and idf_i, the inverse document frequency, is computed as $idf_i = \log_2 (n/ df_i)$ where n is the number of documents in the collection and df_i is the number of documents in which keyword k_i occurs. To determine the similarity between a query vector $q = (q_1,\ldots,q_N)$ and a document vector $d = (w_1,\ldots,w_N)$, the similarity is the cosine of the angle between the vectors [3]:

$$sim(d,q) = cos(d,q) = \frac{\sum_{i=1}^{N} w_i q_i}{\sqrt{\sum_{j=1}^{N} w_j^2 \sum_{j=0}^{N} q_j^2}}$$

Most grouping and clustering algorithms deal with dissimilarity rather than with similarity. However, once a similarity measure is defined, dissimilarity can be obtained via a transformation as follows:

$$diss(d,q) = \begin{cases} \dfrac{1}{sim(d,q)} - 1 \text{ if } sim(d,q) \neq 0 \\ \infty \text{ otherwise} \end{cases}$$

Dissimilarity has been used to cluster the textual user needs mapped into vector space using partitioning around medoids to study data structure and to obtain information on the number of clusters present in the data. Details follow.

Partitioning around medoids (PAM). PAM groups PRI using a medoid, which is the PRI that is closest to the center of a cluster composed of k other PRI. The parameter k is selected so that the average dissimilarity, with respect to the medoid, is minimal.

The algorithm works as follows: first, k PRI are randomly selected and promoted as medoids, *i.e.*, representatives of clusters. Each PRI is then assigned to the nearest medoid X. An objective function is calculated as the sum of dissimilarities of all PRI to their nearest medoids, using the squared error criterion. The algorithm then swaps a randomly selected PRI Y with one medoid X, if the swap reduces the objective function. The swapping process is iterated until the objective function can no longer be reduced [26].

Obtaining the optimal number of clusters. To determine the optimal number of clusters, Kaufman and Rousseeuw [13] proposed *silhouette statistics*. For a PRI i, let $a(i)$ be the average distance to the other PRI in its cluster, and $b(i)$ be the average distance to PRI in the nearest cluster. The silhouette statistic is:

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}.$$

Kaufman and Rousseeuw suggested choosing the optimal number of clusters as the value maximizing the function $s(i) = (b(i) - a(i)) / max(a(i), b(i))$ over all the PRI. Traditionally, it is assumed that the error curve shows a knee for the optimal number of clusters [7]. Values of the maximum of the average silhouette statistics above 0.70 indicate that a very strong clustering structure has been found – that is, separated clusters. Values between 0.50 and 0.70 highlight a reasonable structure, while values in the range 0.25 and

0.50 indicate a weak structure. Values below 0.25 indicate the absence of a structure.

Clustering with AGNES. We cluster the PRI using the AGNES clustering algorithm (Agglomerative Nesting) [27] to obtain evidence of a hierarchical structure because several respondents gave close answers and there was no easy way to manually group them. PAM gave a weak cluster structure so we used AGNES, which revealed a strong hierarchical structure, because it has been used in previous reverse engineering research. AGNES uses the dissimilarity between pairs of PRI in a cluster and between pairs of clusters to merge PRI and clusters iteratively. First, AGNES selects the two least dissimilar PRI and merges them into a cluster. It proceeds similarly with every PRI, thus forming clusters of pairs of PRI. Then, AGNES iteratively selects least similar clusters and merges them into larger clusters. AGNES ends when all the PRI are merged into a unique cluster. AGNES selects the closest PRI to be merged using different strategies such as: average dissimilarity between the PRI in the two clusters, or minimum or maximum dissimilarity between any two pairs of clusters. In reported computations, average dissimilarity between PRI was applied; similar results were obtained with the other two strategies. At each step, AGNES provides an Agglomerative Coefficient (AC) measuring the clustering structure of the clusters. AC is measured as the strength of the hierarchical structure discovered; an AC value above 0.9 is an indication of a very strong hierarchical structure. The AGNES strategy is to build a complete tree grouping all PRI together. However, only sub-trees that cluster very similar PRI are of interest. In PREREQIR, we impose a minimum value of similarity between PRIs grouped into a sub-tree node, thus pruning the tree and grouping together only strongly related PRIs. Non-clustered PRIs are outliers with respect to the structured PRI document but not necessarily with respect to the traditional outlier definition. In fact, the imposed similarity threshold can be much higher than 1.5 the inter-quartile range below the lower quartile.

3.3. Analyzing PRI

The result of the structuring of the PRI is a set of clusters that are *merged* PRI, *i.e.*, PRI that embody the stakeholders' understanding of a system or domain. Our analysis of these merged PRI consists of two major activities: understanding the outliers (see Section 4) and automatically labeling the merged PRI.

Automatically Labeling Merged PRI. To generate a useable textual artifact of the merged stakeholder PRI, we label each cluster. First, we parse each PRI of a cluster, remove stop words, and apply a stemmer. We then build a cluster-specific dictionary with associated weights where every word is weighted by its frequency in the cluster. If a word is in all the PRI in a cluster, its weight is 1.00. If a word appears in half of the PRI, its weight is 0.50. Next, for the given stemmed PRI, we sum up the weights of the stems present in the cluster dictionary to obtain a *positive* weight. Then, we count the number of words in the cluster-specific dictionary that are absent in the current PRI and obtain a *negative* weight. We associate a fitness value to the PRI computed as the ratio between the positive and negative weights. Finally, we select the PRI with the highest fitness value in the cluster as its label.

4. Case Study of PREREQIR

We illustrate PREREQIR on a case study of PRI for a Web browser collected from numerous stakeholders.

4.1. Objects of the Study

A Web browser is a ubiquitous system, commonly used in today's society by a large variety of stakeholders. It allows searching for information, surfing on the Internet, posting news, and writing blogs. Moreover, for such a common system, we were able to find generic requirements specifications for comparison with the PREREQIR PRI.

4.2. Subjects of the Study

We used 'convenience sampling' and sent our questionnaire to more than 200 of our colleagues and acquaintances. Among the 200 recipients, 25 sent back their questionnaires, of which we kept only 22 for this study. We omitted any questionnaire that was not totally completed.

The demographics of the respondents of the 22 retained questionnaires follows. On average, the respondents were 36 years of age (σ = 9.55 years). It took the respondents, on average, 29 minutes to write down their PRI or user needs (σ = 10 minutes). Twenty respondents were male, two were female. Eighteen respondents have held a bachelor's degree for an average of ~13 years (σ = ~10 years). Seventeen respondents have held a master's degree for an average of ~11 years (σ = ~8 years). Eleven respondents have held a Ph.D. degree for an average of ~8 years (σ = ~7 years). Among the respondents, there were: 10

researchers, five lecturers/professors, four students, one programmer, and two project managers. All of the respondents reported using Web browsers several times a day. The respondents were from nine different countries, including Italy, Canada, Hungary, the United States, France, and Germany.

4.3. Obtaining the PRI

A key decision for the study was the form and the tool to obtain stakeholder PRI. Our goal was to include a variety of stakeholders: managers, programmers, researchers, students, non-computer experts, etc. Therefore, we developed a questionnaire where stakeholders can report, in free text, a ranked list of what they perceived as the essential features of a Web browser. The questionnaire is available at https://web.soccerlab.polymtl.ca/repos/soccer-lab/web-questionnaire/. We sent the questionnaire to the 200 participants in early 2008 under strict anonymity. That is, we know to whom we sent the questionnaire, but we do not know who answered it because the PRI were received by a dedicated process on our e-mail server and were rendered anonymous before being sent to us.

The 22 retained questionnaires contain 433 PRI or user needs in the form of sentences. Randomly selected examples of such PRI include:

"Tabbed Browsing"

and:

"Offline working mode – The browser should allow the user to browse previously visited web pages while disconnected from the network."

These two PRI highlight the gap existing among the level of abstraction, the wording, and the consideration of the different stakeholders. Some descriptive statistics for the PRI obtained from the respondents follow. The unprocessed PRI range from 1 to 69 words. On average, the unprocessed PRI have 15.53 words. When stemmed and stopped, the average length of PRI is 10 words. The PRI provided by the respondents are, therefore, rather short.

4.4. Structuring the PRI

Following our method, we applied the various IR and clustering techniques on the obtained PRI. A preliminary study of our data set revealed a weak structure with a maximum of the average silhouette statistics of 0.26 in the region between 165 and 168 clusters; this suggests an optimal number of clusters of about 170 and the presence of a weak separation between clusters.

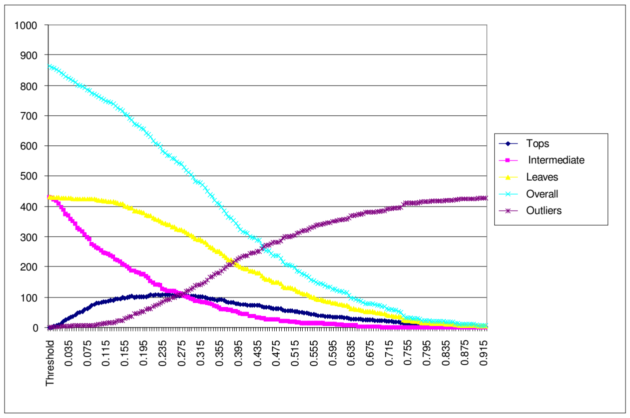

Figure 1. The Structure of the Clusters.

We then applied the Agglomerative Nesting algorithm to the 433 user needs/PRI. When building the complete tree, where all the user needs are contained in the tree structure, the AC was 0.99, providing evidence of a strong hierarchical structure in the PRI. Figure 1 shows relevant AGNES information: an average similarity score is assigned at each level of the built tree. Top clusters, which aggregate lower level clusters, correspond to more general PRI. We observe that the number of top level clusters, the dark curve at the bottom, reaches a plateau in the similarity region close to 30%, where the overall tree contains about 500 nodes and 300 leaves.

4.5. Analyzing the PRI

Once obtained, the clusters represent merged PRI that embody the stakeholders' understanding of a generic Web browser. We briefly discuss outliers and the labels of the merged PRI.

Understanding Outliers. Setting a similarity threshold or choosing any given height in the tree built by AGNES results in the exclusion of a subset of PRI from the merged PRI. These excluded needs, in some sense, represent outliers not captured while automatically merging PRI. When we applied the traditional outlier definition, the percentage of outliers was considerably lower, about 5-15%. We made sure that retained AGNES clusters excluded these PRI. In this process, we discovered two categories of outliers. In the first category, there are very detailed user needs referring to a particular given technology such as Ajax or supported features such as "the browser should pass the ACID and ACID2 test." These are discovered via the traditional definition of outliers. The second category includes outliers that are due to vagueness in the user need formulation (*e.g.*, "it should be easy to use") or due to very different ways of expressing user needs that are already retained in the PRI. For a threshold of 0.36, about 55% of the PRI were common to two or more stakeholders and 42% were outliers.

Labeling Clusters. We labeled the clusters following the method described in Section 3.3. We validated each label manually using the process described in Section 5.1 and found that about 82% of the labels are correct. Some clusters and their labels are:
Cluster I contained these user needs or PRI:
1 – "The browser shall allow searching text in a page."
2 – "**The web browser shall allow users to search text on the page.**"
PRI 1 has positive weight, negative weight, and fitness scores of 6, 2, 3, respectively. PRI 2 has scores 7, 1, 7 and is thus selected as the label. Other clusters include (with labels in bold):

- *Cluster M*: (1) "Show source HTML for current page."; (2) "**Possibility to show the HTML source of the current page.**"
- *Cluster N*: (1) "**The browser shall support the tabbed browsing.**"; (2) "Tabbed browsing."
- *Cluster Z*: (1) "**Should have help system context sensitive help glossary on line help, etc.**"; (2) "Help – The system should provide context sensitive help."

5. Assessment of PREREQIR

The previous section illustrated our method on a case study. We now assess the accuracy of our method in terms of precision and recall [3]. The first assessment concerns the precision and recall of the method with respect to the obtained clusters and answers the question: "How relevant are the merged PRI?" The second assessment concerns the usefulness of the method on a typical traceability problem and answers the question: "How revealing are the merged PRI compared to PRI obtained by another method?"

5.1. Cluster Verification

To assess the relevance of the merged PRI, we manually verified each cluster and then calculated the standard IR measures precision and recall. Recall measures the degree to which all the documents matching a given query in a collection are retrieved. In our method, recall refers to whether or not our method selects the right clusters. Precision measures the quality of the retrieved lists of documents. In our experiment, precision measures the quality of the clusters.

Two of the authors independently verified the clusters, assigning "Yes," "No," or "Maybe" to each cluster. A conservative approach was used to resolve conflicts: "Yes" was assigned if both authors said "Yes," "No" was assigned if one of the authors said "No," and "Maybe" was assigned in the other cases.

We use this ground truth conservatively, not in favor of the method, by only considering "Yes" assignments when computing precision and recall. The following procedure was used to calculate the precision and recall at various threshold levels. For a given similarity threshold, we scan the spreadsheet of the clusters vetted by the researchers. We keep all the clusters, the merged PRI, above the threshold. PRI were then classified as follows: if a PRI is in a cluster marked as "Yes," we gave the PRI the same status, "Yes" - even if the PRI also appears in a cluster marked as "No." We then computed the precision and recall.

As can be seen in Figure 2, recall starts at about 66% when imposing no threshold on the similarity score, *i.e.*, all clusters are considered, and increases to 78% when the threshold is 0.285. It sharply decreases as the threshold increases. Precision starts at about 43% with no threshold and then climbs to about 86% at threshold 0.82, then continues up to 1.0 at threshold 0.92 where we have only one cluster.

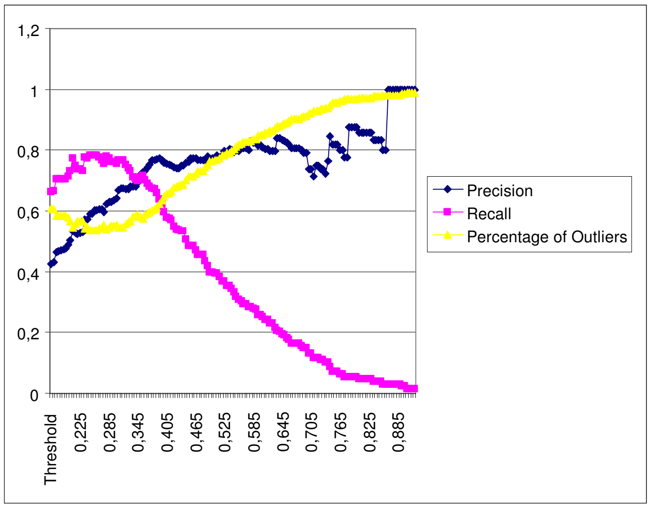

Figure 2. Recall, Precision, and Outliers of Clusters.

The percentage of outliers is a bit high, but a visual examination of these showed that many of the PRI left outside of the clusters are already in some existing clusters, *e.g.*, print page, bookmark page, etc. Our conservative approach also tends to increase the number of outliers.

The optimal number of clusters is also of interest. Figure 1 shows that as the similarity threshold decreases, there is an increase in the number of retained clusters and a loss in accuracy. In particular, in the region close to 50% similarity, having around 200 clusters, the number of clusters judged as "No" by the authors increases and thus reduces accuracy. This is not surprising because PAM predicted around 170 clusters. It can also be seen that top, high level concepts confirm a curve with an inflexion change in concavity and a maximum. If we use the inflexion before the maximum as the threshold, we obtain a similarity threshold close to 50%, which yields about 80% recall and 70% precision. The error rate also means that 70% of the clusters are valid. Recalling that this automated process did not require any manual intervention, 70% appears to be a useful accuracy level. Higher threshold values increase precision at the price of lowered recall. On the other hand, we can see in Figure 2 that precision and recall intersect in the region close to 36% similarity. Overall, a threshold of 0.36 gives rise to 128 retained clusters and precision/recall values of about 70%.

5.2. Traceability Task

To assess the usefulness of the method, we use the labeled PRI, which consists of the 128 common user needs and the 181 outlier user needs, and compare them against the PRI for a Web browser provided by an independent organization on its Web site, www.learnthenet.com. We refer to these as the LtN PRI [12]. This mimics the situation in which an obsolete requirement document is traced into new PRI, to quantify how much of the new functionalities are already implemented and reusable, for example. We used vector space retrieval with *tf-idf* weighting to perform the trace.

There are 20 LtN PRI, textual in nature, ranging from 5 to 73 words, having on average 23.5 words. Some examples of LtN PRI are:

- LtN18: "Should include a Status Bar at the bottom to display the progress of web page transactions, such as the address of the site contacted, whether the host computer has been contacted, and the size and number of the files to be downloaded."
- LtN10: "The toolbar should include a Reload or Refresh button to load the web page again."

Using a similarity threshold of 0.20, two of the co-authors independently marked each LtN PRI as "Yes," "No," or "Maybe." The results were then reconciled using the conservative approach described in Section 5.1. The results show that 14 of the 20 LtN PRI are found in the PRI obtained with certainty from our 22 respondents. The 14 PRI were all marked as "Yes" by both authors. If we also include the two marked as "Maybe," there are 16 LtN PRI out of 20 that are traced in the respondents' PRI. An example of LtN PRI not found follows:

- LtN6: "Should include a toolbar: a row of buttons at the top of the browser that helps travel through the web of possibilities, keeping track of where the user has been."

Seventeen of the 128 common PRI are found in the LtN PRI - 19 if we include PRI marked as both "Yes" and "Maybe." Examples of missing PRI include:

- *Cluster O*: "Possibility to show the HTML source of the current page."
- *Cluster P*: "The system shall support automatic updates."

Ten of the 181 outlier PRI are found in the LtN PRI - 14 if we include PRI marked as both "Yes" and "Maybe." Missing PRI include:

- *Cluster Q*: "Should support localization can start with English but all GUI stuff should be externalized"
- *Cluster R*: "Allow saving of portion of the current page in many formats PS, PDF, JPG"

Therefore, it appears that PRI can be used to ensure that all requirements are indeed needed by the users. In our study, between 70% ("Yes" only) and 80% ("Yes" and "Maybe") of the LtN PRI are also found in the PRI obtained from the respondents.

PRI can also be used to ensure that an old requirement specification is still complete. It appears that a number of PRI have not been included in the LtN generic Web browser specification. As a minimum, the common and outlier PRI should be examined to ensure consideration for inclusion. We conclude that our method is useful in a software development project.

5.3. Discussion and Threats to Validity

In our study, we attempted to obtain PRI from a diverse set of stakeholders. The presented results are encouraging, yet are subject to some validity threats.

External validity threats concern the generalization of our findings. We believe that our findings support the evidence that PREREQIR can be applied and that modern IR techniques and tools help reduce the cost of building a PRI document. The scenario presented is realistic and likely to be representative of many real-world situations. However, although we sent our questionnaire to more than 200 colleagues and acquaintances, we essentially obtained PRI from respondents who use Web browsers several times per day and thus can be considered experts in Web browsers. Also, the group is highly educated. We believe that only people with knowledge in Web browsers were intrigued enough by our study to answer the questionnaire in a short time frame. This unwanted homogeneous group of experts decreases the generalization of our study and, consequently, the generalization of our method. In future work, we will apply our method on PRI obtained from more respondents, from various backgrounds.

Moreover, we have not specifically studied the issue of vocabulary. However, our work in Information Retrieval and on many other systems (such as scientific instruments, space telescopes, LEDA, Albergate, Eclipse, and Mozilla) shows that vocabulary is tied to the domain and to the system, and that is why thesauri are useful. Thus, we plan to replicate the study and integrate other tools such as thesauri or ontologies, when available, in the extension of the current work.

Construct validity threats concern the relationship between the theory and the observation. Such threats arise from possible errors introduced by measurement instruments. PRI clustering, indexing, and similarity computation were performed using widely adopted toolsets. For example, we used the Perl stopper and stemmer available from the Virginia Polytechnic Institute and State University. Also, a TDF-IDF implementation is available from the open-source Lucene project. We used R from CRAN for PAM, and our Java AGNES version is the re-implementation of a well known and classic AGNES algorithm. Precision and recall are computed based on the sole agreed decision of our two experts and are thus computed in a conservative way. Nevertheless, we cannot exclude the possibility that another chain of tools, or ranking from different experts, may produce slightly different results, or that different developers would rate functions in different ways. One critical element is the choice of the AGNES similarity threshold. We inspected the precision and recall plots versus the similarity thresholds to validate our choice of a similarity close to the silhouette knee; other researchers may apply different strategies, such as grouping a fixed percentage of PRI or requiring that each cluster contains at least three or more PRIs. These strategies would produce different tree pruning thresholds and thus would structure different PRI documents.

Reliability validity concerns the possibility of replicating the study and obtaining the same results. The questionnaire and documentation are publicly available on the server of the SOCCER Laboratory (https://web.soccerlab.polymtl.ca/repos/soccer-lab/web-questionnaire/); the set of collected requirements are available from the authors upon request.

Internal validity refers to the influence of independent variables on dependent variables and the existence of confounding factors. The main threat to the internal validity of this study is the level of subjectivity introduced by experts. However, as explained in Section 5.1 and 5.2, we attempted to minimize this subjectivity with a conservative approach based on the expert agreement (both experts must agree on a “Yes” for a cluster or a link to be correct).

We do not know if the accuracy of 70% clusters marked as relevant PRI is generally valid as only one study was performed on one domain, but it is still an interesting finding. More studies on different domains will follow in future work.

6. Related Work

Related work is organized into two subsections: (1) mental models and requirements engineering, and (2) application of IR techniques.

6.1. Mental Models and Requirements Engineering

The mental model that people have of a domain is often incomplete [4, 18] which leads to faulty reasoning. Communication issues, such as use of different terminology, may result in misunderstandings of people’s mental models of a to-be-built system [4]. Differing goals of users and analysts can also indicate different mental models. Browne et al. [4] suggest that the most common problems of requirements determination need to be examined, and that the cognitive underpinnings of these problems need to be identified. They further recommend appropriate techniques such as devil’s advocacy, what-if analysis, scenario response, etc. to deal with the cognitive issues. In related work, Pitts and Browne looked at how analysts decide that enough requirements information has been collected. There exist cognitive stopping rules that help explain analyst behavior. Representational stability is such a rule, that analysts will stop asking for additional information when they feel that their mental model is stable [19].

Goldin and Berry applied signal processing methods to collections of natural language text in order to extract abstraction information. They demonstrated their method by abstracting tables of contents from software system requests for proposals [31]. Svetinovic examined the semantic similarity of domain models (DMs) specified by students at the University of Waterloo and determined that the use of his artifact, the unified Use Case statechart, could help improve semantic likeness of the DMs [32].

Kudikyala and Vaughn applied pathfinder networks (PFNETs) from the artificial intelligence field to sets of requirements categorized by two groups – developers and customers. These sets of requirements were deemed as the mental models of each stakeholder group. The PFNETs of these models were then mathematically compared to find the similarity, as well as to identify duplicate or misunderstood requirements [22]. The main differences between the work of Kudikyala and Vaughn and the work presented here are that: our method involves elicitation of requirements; we concentrate on generic or domain requirements, as opposed to system or application specific requirements; we do not ask the stakeholders to categorize the requirements; and we do not use simple correlation to determine similarity.

6.2. Application of IR techniques

Several surveys and overviews of clustering techniques applied to software engineering have been published in the past, for example, by Wiggerts [28] and by Tzerpos and Holt [24]. The latter authors, in [25], defined a metric to evaluate the similarity of different decompositions of software systems. They proposed a novel clustering algorithm specifically conceived to address the peculiarities of program comprehension; they also addressed the issue of stability of software clustering algorithms [26].

Applications of clustering to reengineering were suggested in [1] where Anquetil and Lethbridge devised a method for decomposing complex software systems into independent subsystems. Source files were clustered according to file names and their name decomposition. Mancoridis et al. [15] presented an approach relying on inter-module and intra-module dependency graphs to refactor software systems.

Cleland-Huang et al. [11] examined the application of non-functional requirements (NFR) to aspects. They mined and then classified NFR from a collection of textual requirements by using indicator terms specific to the NFR. For example, they used the terms confidentiality, integrity, and completeness along with other terms to detect security NFR. It seems that such a method may also prove useful for eliciting functional requirements, although a training set must be obtained. Such a method may also be applicable to obtaining PRI, especially if the PRI are non-functional such as reliability, performance, security, etc.

There has been work on mining aspects or non-functional requirements from computer applications by performing pattern matching against the Abstract Syntax Tree or by analyzing dynamic execution traces, but each of these methods require that source code or executable exist [14, 17]. It seems reasonable that such approaches may work for mining PRI also. An advantage of our method is that it does not require that code, design, or even requirements exist.

7. Conclusions and Future Work

In this paper, we presented a method, PREREQIR, to recover and structure the pre-requirements information (PRI) obtained from stakeholders, which are projection of the stakeholders' mental models. We applied information retrieval (IR) techniques to cluster the PRI, to analyze their hierarchical structure, to identity unique PRI, and to automatically label the clusters.

We report the results of applying PREREQIR on the PRI obtained from 22 respondents. We show that using the agglomerative nesting (AGNES) algorithm, we cluster PRI with an accuracy of 70%. We show that, for a similarity threshold of about 0.36, about 55% of the PRI were common to two or more stakeholders and 42% were outliers. We automatically label the common and outlier PRI with 82% of the labels being correct. Also, the method achieves roughly 70% recall and 70% precision when compared to a ground truth. Bearing in mind that IR methods tend to work well on large datasets, our method achieves a decent precision and recall on our relatively small dataset - 433 total PRI. Thus, it seems reasonable that our method could be applied to a larger set of PRI with at least comparable results. Although our method may not be generally applicable, the accuracy is high enough to warrant further investigation.

In future work, we will survey a broader, more diverse set of stakeholders for multiple domains. We will improve the questionnaire used to collect information. We will study the use of different clustering mechanisms. We will also study the use of more sophisticated methods for selecting labels.

8. Acknowledgments

Our thanks to Stephanie Ferguson, Jane Cleland-Huang, and Lisa Montgomery. This work is sponsored by NASA under grant NAG5-11732, NSERC Canada Research Chair Tier I in Software Change and Evolution, NSERC Discovery grant, and NSF under grant CCF-0811140.

9. References

[1] Anquetil N., Lethbridge T., "Extracting concepts from file names – A new file clustering criterion," Proceedings of the International Conference on Software Engineering, IEEE Computer Society Press, 1998, pp. 84-93.
[2] Antoniol, G., Canfora, G., Casazza, G., De Lucia, A., and Merlo, E. "Recovering Traceability Links between Code and

Documentation." IEEE Transactions on Software Engineering, Volume 28, No. 10, October 2002, 970-983.
[3] Baeza-Yates, R., B. Ribeiro-Neto. Modern Information Retrieval, Addison-Wesley, 1999.
[4] G.J. Browne, V. Ramesh, "Improving information requirements determination: a cognitive perspective," *Information & Management*, Elsevier, Volume 39, Issue 8, September 2002, Pages 625-645.
[5] Dhar, V. Jarke, M. "Dependence directed reasoning and learning in systems maintenance support." IEEE Transactions on Software Engineering, Volume 14, No. 2, February 1988, 211-227.
[6] Egyed, A. "A Scenario-Driven Approach to Trace Dependency Analysis," IEEE Transactions on Software Engineering (TSE), Volume 29, Number 2, February 2003, pp. 116-132.
[7] Gordon, A. D., "Classification," Chapman & Hall, Boca Raton, 2nd edition, 1999.
[8] O.C.Z. Gotel and A.C.W. Finkelstein. "An analysis of the requirements traceability problem." In 1st International Conference on Requirements Engineering, pages 94--101, 1994.
[9] Hayes, J.H.; Dekhtyar, A.; Sundaram, S.K., "Advancing candidate link generation for requirements tracing: the study of methods," Transactions on Software Engineering, Volume 32, Issue 1, Jan. 2006 Page(s): 4 – 19.
[10] Jane Cleland-Huang, Carl K. Chang, Mark J. Christensen: "Event-Based Traceability for Managing Evolutionary Change." IEEE Trans. Software Eng. 29(9): 796-810 (2003).
[11] Jane Cleland-Huang and Raffaella Settimi and Xuchang Zou and Peter Solc, "The Detection and Classification of Non-Functional Requirements with Application to Early Aspects," in Proceedings of the IEEE International Conference on Requirements Engineering (RE) 2006.
[12] Learn the Net, Michael Lerner Productions, www.learnthenet.com/.
[13] Kaufman, L. and Rousseeuw, P.J. (1990). Finding Groups in Data: An Introduction to Cluster Analysis. Wiley, New York
[14] Magiel, B., A.v. Deursen, R.v. Engelen and T. Tourw´e, 2004. "An evaluation of clone detection techniques for identifying crosscutting concerns." In Proc. Intl. Conf. Software Maintenance (ICSM). IEEE Computer Society.
[15] Mancoridis S., Mitchell B. S., Rorres C., Chen Y., Gansner E. R., "Using automatic clustering to produce high-level system organizations of source code," Proceedings of the International Workshop on Program Comprehension, IEEE Computer Society Press, 1998.
[16] Marcus, A.; Maletic, J. "Recovering Documentation-to-Source Code Traceability Links using Latent Semantic Indexing," Proceedings of the Twenty-Fifth International Conference on Software Engineering 2003, 3 – 10 May 2003, pp. 125 – 135.
[17] B. Nora, G. Said, and A. Fadila. "A comparative classification of aspect mining approaches." Journal of Computer Science, 2(4):322--325, 2006.
[18] D.N. Perkins, R. Allen, J. Hafner, "Difficulties in everyday reasoning, in: W. Maxwell (Ed.), Thinking: The Expanding Frontier, Franklin Institute Press, 1983.
[19] Mitzi G. Pitts and Glenn J. Browne. "Stopping Behavior of Systems Analysts During Information Requirements Elicitation." *Journal of Management Information Systems,* M.E. Sharpe, Inc., Volume 21, 2004, pp. 203-226.
[20] B. Ramesh, "Toward Reference Models of Requirements Traceability. " IEEE Trans. Software Eng. 27(1): 58-93 (2001).
[21] Domain Engineering and Domain Analysis, SEI Software Technology Roadmap, http://www.sei.cmu.edu/str/descriptions/deda.html.
[22] K. Kudikyala and R. Vaughn, "Understanding Software Requirements Using Pathfinder. Networks", *CROSSTALK, The Journal of Defense Software Engineering*, May 2004, pp. 21 – 25.
[23] A.G. Sutcliffe and N. A. Maiden, "Use of Domain Knowledge for Requirements Validation," In Proceedings of the IFIP Wg8.1 Working Conference on information System Development Process (September 01 - 03, 1993). N. Prakash, C. Rolland, and B. Pernici, Eds. IFIP Transactions, vol. A-30. North-Holland Publishing Co., Amsterdam, The Netherlands, 99-115.
[24] Tzerpos V., Holt R. C., "Software Botryology: Automatic clustering of software systems," DEXA Workshop, IEEE Computer Society Press, 1998, pp. 811-818.
[25] Tzerpos, V., Holt, R. C., "MoJo: A distance metric for software clusterings," Proceedings of Working Conference on Reverse Engineering, IEEE Computer Society Press, 1999, pp. 187-195.
[26] Partitioning Around medoids (PAM). www.unesco.org/webworld/idams/advguide/Chapt7_1_1.htm
[27] Agglomerative Nested Clustering, http://www.unesco.org/webworld/idams/advguide/Chapt7_1_4.htm
[28] T. A. Wiggerts "Using clustering algorithms in legacy systems remodularization," in: Proceedings of IEEE Working Conference on Reverse Engineering, IEEE Computer Society Press, 1997.
[29] Andrea Zisman, George Spanoudakis, Elena Pérez-Miñana, Paul Krause: "Tracing Software Requirements Artifacts." Software Engineering Research and Practice 2003:448-455
[30] Mohammad El-Ramly, Eleni Stroulia and Paul Sorenson "Recovering software requirements from system-user interaction traces" Proceedings of the 14th international conference on Software engineering and knowledge engineering, 2002, pages 447-454
[31] Goldin, L. and Berry, D. M. 1997. AbstFinder, A Prototype Natural Language Text Abstraction Finder for Use in Requirements Elicitation. Automated Software Engg. 4, 4 (Oct. 1997), 375-412.
[32] Svetinovic, D. 2006 Increasing the Semantic Similarity of Object-Oriented Domain Models by Performing Behavioral Analysis First. Doctoral Thesis. UMI Order Number: AAINR23541., University of Waterloo.