Concept Mapping as a Means of Requirements Tracing

Leonid Kof
Fakultät für Informatik, Technische Universität München, Boltzmannstr. 3, D-85748, Garching bei München, Germany, kof@in.tum.de

Ricardo Gacitua, Mark Rouncefield, and Pete Sawyer
Computing Department, InfoLab21, South Drive, Lancaster University, Lancaster, UK, LA1 4WA, {r.gacitua | m.rouncefield | p.sawyer}@lancaster.ac.uk

***Abstract*—Requirements documents often describe the system on different abstraction levels. This results in the fact that the same issues may be described in different documents and with different vocabulary. For analysts who are new to the application domain, this poses a major orientation problem, as they cannot link different concepts or documents with each other.**

In the presented paper, we propose an approach to map concepts extracted from different documents to each other. This, in turn, allows us to find related passages in different documents, even though the documents represent different levels of abstraction. Practical applicability of the approach was proven in a case study with real-world requirements documents.

I. Different Abstraction Levels in Requirements Documents

At the beginning of every software project, some kind of requirements document is usually written. There are many different modeling notations that support the precise description of requirements and which support reasoning to help achieve completeness and consistency in the specified requirements. However, use of these notations is only feasible if they are intelligible to the documents' authors and to the stakeholders who are required to approve the documents' contents. In most cases, they are not and therefore, as the survey by Mich et al. shows [1], the great majority of requirements documents are written in natural language. As a consequence, most requirements documents are imprecise, incomplete, and inconsistent, because precision, completeness and consistency are extremely difficult to achieve using natural language as the main presentation means.

These problems become even worse when it comes to different abstraction levels. Typically, requirements documentation contains both high-level documents, describing the application domain and general goals of the system to be developed, and low-level documents, focusing on application scenarios, user interfaces, etc. The links between these documents mostly remain unspecified, as they are perfectly obvious for the document authors. However, for an analyst new to the application domain or to the project, it often remains unclear how the different documents are linked to each other. The situation is further complicated by the fact that documents written at different abstraction levels can contain different vocabulary. This renders recently developed techniques for automatic tracing [2], [3], relying on common vocabulary in different documents, infeasible.

This situation is typical to software projects and results in a vicious circle: the experienced domain expert has the necessary knowledge, but is either too busy to express it, or does not know what should be written down. An analyst new to the domain sees a lot of gaps in the specification, but does not know which of the gaps are genuine and which of them are due to missing links between documents.

In the presented paper, we propose an approach to concept mapping and tracing that caters for different abstraction levels. The generated mappings and traces can then be used to guide an interview with a domain expert. It turned out in our case study that the domain expert, simply by arguing about correctness or incorrectness of the provided mappings and traces, revealed a lot of relevant information that seemed too obvious when writing the document; an example of taken-for-granted knowledge [4]. This additional information definitely contributes to the exchange of domain-specific knowledge.

The presented approach consists of three phases: In the first phase, we extract domain specific concepts from requirements documents. Then, in the second phase, we map every low-level concept onto the most similar high-level concept. Finally, in the third phase, we find for every low level concept the occurrences of its corresponding high-level concept in the high-level documents. This allows us to link low-level and high-level documents with each other, despite different vocabulary used in the documents.

Outline: The remainder of the paper is organised as follows: Section II presents the case study used to evaluate our approach. Sections III and IV are the technical core of the paper: Section III presents the proposed method of concept mapping and tracing, and Section IV the evaluation of the method. Then, Section V presents the lessons learned in the case study. Finally, Sections VI, VII, and VIII present the related work, some directions for future work, and the summary of the paper.

This work was supported by a fellowship within the Postdoc-Programme of the German Academic Exchange Service (DAAD) and by EPSRC grant EP/F069227/1 MaTREx.

II. Case Study: Postgraduate Admission Portal

A specification of a postgraduate admissions portal for a UK university was used as a case study in the presented work. The requirements for the admission portal consist of several documents: one high-level document (approximately 100 pages), describing the general problems that the system should solve, and five low-level documents (2 to 4 pages each), specifying concrete application scenarios. The documents were written by several authors, but there was at least one author who participated in all the documents. All documents are written in grammatically correct English, but no language restrictions in the sense of controlled languages (cf. [5]) or special language or document structure were enforced.

The problem description in the high-level document states informally, what improvements to the currently existing process should be achieved with the new system:

Electronic workflow:

- *enabling work with actions to be moved electronically through (to and from) the system.*
- *enabling applications to be processed and audited electronically via the system.*
- *enables context sensitive distribution e.g., acceptance of offer triggers links to information related to a department or forms to download depending on fee status etc.*
- ...

The scenarios documents are more concrete and specify the desired system behaviour in exemplary situations, like this:

- *Miss X, having completed her undergraduate studies in the PRC, is interested in studying a professionally-accredited postgraduate finance course in the UK.*
- *Mrs Y, a management school (MS) faculty admissions officer (FAO), logs on to the system and sees an enquiry from Miss X about local transport options.*
- *The enquiry also requests that Miss X can book a chat session with somebody about student accommodation and funding options.*
- ...

The difficulty in linking the documents written at different abstraction levels lies in the fact that the vocabulary used in these documents is different, which renders recently developed techniques for automatic tracing infeasible. For example, the second sentence of the above application scenario, stating that "Mrs Y ... logs on to the system and sees an enquiry..." is definitely relevant for the general goal of electronic workflow, but, due to differences in vocabulary, the existing tracing methods would not identify a relationship between this sentence and the phrase "electronic communication". This link is tacit, in the sense that its existence is obvious for a domain expert but hidden for a novice. Identification of related low-level and high-level concepts and passages, despite differences in vocabulary, and in this way providing hints about potential tacit knowledge in the documents, is the goal of our approach presented in this paper.

III. Tracing and Concept Mapping

The aim of the approach presented in this paper is to find concepts and document passages that are close to each other, even though they use different words to represent similar issues. The proposed mapping and tracing procedure consists of three phases:

1) **Extraction of domain-specific concepts:** Before looking for similar concepts and passages, it is necessary to know what concepts are really relevant. The extraction of domain-specific concepts is presented in Section III-A.
2) **Concept mapping:** To solve the problem of different concepts used to describe the same issue, we define a similarity metric with which to map similar concepts to each other. The mapping algorithm is presented in Section III-B.
3) **Tracing:** When the concept mapping is known, we can find related text passages. We assume that two passages are related if they contain concepts that are mapped to each other. Details of the related passage tracing are presented in Section III-C.

A. Extraction of Domain-Specific Concepts

Extraction of domain-specific concepts is a prerequisite for our tracing technique, as every concept from a low-level document is mapped to a high-level concept before tracing. Two techniques are used for concept extraction: frequency profiling and relevance-driven abstraction identification. Both techniques are presented below.

- **Frequency profiling:** There exist linguistic corpora, like the British National Corpus [6] that document the everyday usage of language. If the frequency of some term in a domain specific document significantly deviates from the frequency of the same term in everyday usage, this can be an indicator that this particular term is domain specific [7]. In addition to pure frequency profiling, we can use part-of-speech tagging[1] to filter the results. In this case we obtain the specified part of speech only (nouns, adjectives, ...) as suggested terms. OntoLancs [8], the tool used in the presented work, provides a user interface to sort all the extracted terms by their relevance (=frequency deviation) and to help the analyst decide which of the terms represent domain-specific concepts.

[1] http://en.wikipedia.org/wiki/Part-of-speech_tagging

- **Relevance-driven abstraction identification:** The corpus-based frequency profiling technique works well for concepts that are signified by single words, but not for compound terms. In order to adapt frequency profiling to compound terms, a new technique called RAI (Relevance-driven abstraction identification) was recently added to OntoLancs [9]. The RAI algorithm comprises the following steps:
  1) Each word in the domain text is annotated with a part-of-speech tag.
  2) The set of words is filtered to remove stop words.
  3) The remaining words are lemmatized to reduce them to their dictionary form, i.e. to collapse inflected forms of words to a base form or lemma.
  4) Each word is assigned a log-likelihood value by applying the corpus-based frequency profiling approach.
  5) Syntactic patterns are applied to the text to identify multi-words term.
  6) A significance score is derived from the frequency profiling.
  7) The terms are ranked according to their significance score.

  As in the case of frequency profiling, the user decides which of the suggested terms represent domain-specific concepts.

B. Concept Mapping

The problem of linking different requirements documents with each other results from the fact that different lexical forms can be used for the same concept. For example, applicants are referred to both as "applicant" and as "prospective student" in our case study. Furthermore, different but related concepts can be used to illustrate the same idea. For example, in our case study, it was stated in the high-level document that communication between the applicants and the admission officers should be automated. In the low-level documents, however, some examples were given where the applicants and the admission officers send and receive e-mails to/from each other, but there were no explicit links to the automated communication in the high-level document.

The main idea of the proposed concept mapping is to find concepts that are similar to each other. When looking for similar concepts, we are interested not only in similar lexical forms, but in similar meanings. This motivates the usage of WordNet [10], a comprehensive taxonomy of English nouns and verbs: When looking for related concepts, we are looking not for similar lexical forms, but for word proximity in WordNet. The WordNet-based search can become computationally expensive, so we perform the search not for every word set or every sentence, but for significant concepts, as extracted by relevance-driven abstraction identification (cf. Section III-A).

For the purposes of concept mapping, we consider every concept as a set of words. As every word can have several meanings, there is no 1:1 mapping between lexical forms and WordNet nodes, called synsets[2]: Every synset characterises one meaning, thus every lexical form can be relevant for several synsets. Analogously, every synset can be relevant for several lexical forms. This idea is illustrated in Figure 1: "mail" and "post" share many meanings, as they both can refer to different aspects of postal services. For example, according to WordNet, they share the meanings "the system whereby messages are transmitted via the post office" (Synset 2 in Figure 1) and "any particular collection of letters or packages that is delivered" (Synset 3 in Figure 1). Additionally, "mail" has the meaning "the bags of letters and packages that are transported by the postal service" (Synset 1), not shared with "post". Analogously, "post" has the meaning "military installation at which a body of troops is stationed", not shared with "mail" (Synset 4). Both words have further meanings as well.

Our main assumption for concept mapping is that, if three words are used in the same application domain to denote related concepts, then there must exist close synsets for them. Conversely, if we can find close synsets for two words, we assume that they denote related concepts. For example, the hypernym (parent node) of "e-mail" is "electronic communication", which, in turn, has "transmission" and "communication" as its hypernyms. Thus, we can infer that "e-mail", used in specific application scenarios, is related to the "communication" concept, used in the generic domain document. In general, given two words, we list all synsets for every word, and then use breadth-first search to find synsets closest to each other. This allows us to define the distance between two words:

Def. 1: The distance between two synsets is the length of the shortest path in the WordNet-graph between these synsets. The distance between two words is the shortest distance between their synsets.

Based on the above definition, we can define distance between two sets of words:

Def. 2: The distance between two sets of words is the shortest distance between their constituting words.

This definition can be illustrated as follows. Let us assume that we have two sets of words:

$$\begin{aligned} s_1 &= \{w_{1.1}, w_{1.2}\} \\ s_2 &= \{w_{2.1}, w_{2.2}\} \end{aligned}$$

Furthermore, let us assume the following relationships for distances between single words, according to Definition 1:

$$d(w_{1.1}, w_{2.1}) < d(w_{1.1}, w_{2.2}) < d(w_{1.2}, w_{2.2}) < \\ < d(w_{1.2}, w_{2.1})$$

[2]see also http://en.wikipedia.org/wiki/Synonym_ring

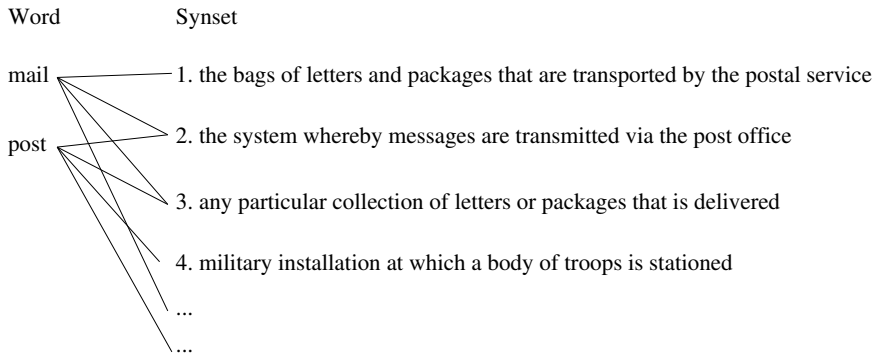

Figure 1. WordNet entries: words and synsets

Table I
MEASURING CONCEPTS SIMILARITY

Consider every multiword concept as a set of words
$sum := 0$
$counter := 0$
while both sets non-empty **do**
  $distance := distance\ between\ word\ sets\ according\ to\ Def.2$
  $sum := sum + \frac{1}{2^{distance}}$
  $counter := counter + 1$
  remove concepts that result in the shortest path from the sets
end while
$WordNet\ similarity := \frac{sum}{counter}$

Then, the distance between s_1 and s_2 is defined as $d(w_{1.1}, w_{2.1})$.

Definition 2 allows us to calculate a similarity metric for multiword concepts. As two multiword concepts can contain several words that are similar to each other, the shortest distance between word sets is not sufficient. However, we can use the shortest distance for an iteratively defined similarity metric.

This iterative algorithm, measuring similarity between two sets of words, is presented in Table I. Given two sets of words, it finds word pairs in multiword concepts that are closest to each other. Every found pair contributes its share to the similarity metric. The weighting of the found word pair with $\frac{1}{2^{distance}}$ emphasises that we are not interested in distant relations between words.

Apart from words that are contained in WordNet, we have to cater for specific words without WordNet entries. For example, in our case study, there was an important abbreviation "DSO"[3]. Automatic expansion of abbreviations is not yet possible. As "DSO" is not contained in WordNet, it does not contribute to the WordNet similarity, as defined above. Nevertheless, it does contribute to concept similarity: as "DSO" is a domain-specific abbreviation, it is likely that two concepts referring to the same abbreviation "DSO" are related with each other. Thus, we introduced an additional similarity metric, based on words that are not contained in WordNet. Let $shared$ be the number of common non-WordNet words in two concepts, and $length_{shorter}$ be the number of words in the shorter concept. Then, the non-WordNet similarity is defined as follows:

$$Non{-}WordNet\ similarity = \frac{shared}{length_{shorter}}$$

The total similarity is defined as

$$\begin{aligned} similarity = &\alpha \times WordNet\ similarity + \\ &(1-\alpha) \times non{-}WordNet\ similarity \end{aligned} \quad (1)$$

By varying α, it is possible to assign higher significance either to words not contained in WordNet (e.g., domain specific acronyms), or to words contained in WordNet.

For example, if we assume $\alpha = 0.8$ and measure the similarity between "DSO communication record" and "mail sent to DSO", we will get:

- WordNet similarity: "communication" and "mail" provide a pair of words most close to each other in WordNet: "communication" is the direct hypernym for one of the "mail" synsets. Thus, the distance is equal to 1 and $WordNet\ similarity = 0.5$. "Sent" is a verb and is not considered in our WordNet similarity metric.
- non-WordNet similarity: one common non-WordNet word ("DSO"), the shorter concept consists of three words $\Rightarrow$ $non{-}WordNet\ similarity = \frac{1}{3}$. For non-WordNet similarity, we consider all words as relevant, in order to obtain non-zero concept length even for concepts consisting of acronyms only.
- $similarity = 0.5 \times 0.8 + \frac{1}{3} \times 0.2 \approx 0.46667$

The above definition of concept similarity allows us to perform mapping between low-level and high-level concepts: For every low-level concept, we look for the high-level concept most similar to it, according to the above metric. This concept mapping allows us, in turn, to find related passages in low-level and high-level documents: We consider two passages as related, if they contain concepts that were mapped to each other.

[3]"Disabled Support Officer"

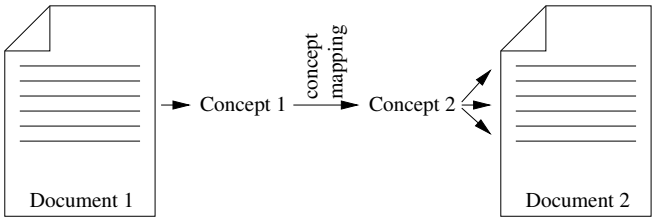

Figure 2. Concept mapping, as applied to document tracing

Table II
ITERATIVE GREP ALGORITHM

result := whole text
consider concept to search for as a set of words
for every *word* in set of words **do**
  restrict the *result* to sentences in the *result* containing *word*
end for

C. Tracing

When the concept mappings are available, we can link documents using different vocabulary with each other. The main idea, illustrated in Figure 2, is to look for concepts that occur in one document, and for every concept, to look for its mapping counterpart in the other document. As the concept mapping algorithm, presented in Section III-B, does not consider the internal grammatical structure of the concepts, any set of words occurring in Document 1 (resp. Document 2) can be considered as Concept 1 (resp. Concept 2) in the sense of Figure 2. In particular, it is possible to declare whole sentences as concepts and then to look for the most similar sentences. However, due to the size of WordNet, the computational effort grows when the concepts become too long, so it is not advisable to skip concept extraction (as presented in Section III-A) and to consider arbitrary sets of words as concepts.

When searching for concept occurrences in a document, we have to take into account that the same concept can occur not only as an uninterrupted word sequence, but its words may be permuted or mixed with other words. For example, "on line application" occurs in the following contexts in our case study:

- *This applies to on-line and manual applications; in fact, on-line applications are printed before processing begins.*
- *Applications can either be on-line or by paper applications.*
- *Paper applications (unlike on line ones) are not automatically acknowledged.*

In order that such occurrences of a concept can be found, we apply the iterative `grep` algorithm, presented in Table II.

This algorithm allows us to find every concept occurrence, at the small price of potential identification of irrelevant passages as relevant (false positives). This strategy was proven to be useful in the evaluation, as it highlights not only the exact occurrences of the sought concept, but also similar phrases.

On the tool side, every found trace is represented as a pair of sentences (sentence from the low-level document and the traced sentence in the high-level document) with several sentences of context information. The context information is provided in order to facilitate the reconstruction of the relevant parts of the documents, as a remedy for the missing GUI.

IV. EVALUATION

For evaluation, we focused on the concept mapping and the actual tracing, as the concept extraction is already evaluated [9]. To evaluate our technique, we interviewed a domain expert about the quality of the generated mappings and traces. We considered traces from application scenarios to the generic domain document. Due to the relatively small volume of scenarios we treated every scenario sentence as a concept. A similar strategy was used in our previous work where we mapped scenario sentences to elements of executable models [11]. Thus, by combining the traces from scenario sentences to models, and from scenarios sentences to the generic domain document, we can bridge the gap between executable models and the generic document.

We converted the generic domain document to plain text and then, using relevance-driven abstraction identification (cf. Section III-A), extracted the 150 most significant concepts from it. As there were not too many acronyms (non-WordNet terms) extracted, we set α for Equation (1) to 0.9, giving higher weight to the WordNet-part of the similarity metric. Then, for every scenario sentence, we calculated the similarity metric and determined the most similar generic concept for every sentence. The concept mappings were presented to the domain expert as pairs of strings, one string representing the scenario sentence, and one string representing the mapping result (most similar generic concept). We evaluated the resulting mapping by interviewing a domain expert on two questions:

1) Is the mapping correct?
2) Is the produced mapping the best one for the given sentence?

We evaluated every "yes" answer as "1" and every "no" answer as "0", and measured the statistical correlation[4] between the expert's answers and automatically calculated similarity metric. We got the correlation of 0.38 on the correct mappings (first question), and 0.21 on best mappings (second question). Given our sample size of approx. 160 sentences to be mapped, both correlation values are statistically significant at the confidence level of 95% and

[4]http://en.wikipedia.org/wiki/Pearson_product-moment_correlation_coefficient

show that the calculated similarity metrics correlate with the relevance/irrelevance decisions made by the human analyst.

In order to evaluate the traces between the documents that result from concept mappings, we picked at random 17 traces and presented them to the domain expert. Larger sample size was not possible due to the availability of the expert. He evaluated 5 of these traces as correct, which corresponds to approx. 29% precision. Given that our technique provides by construction 100% recall, we conclude from high precision (as compared to other tracing approaches) that tracing based on concept mapping can be applied in real-life projects.

V. Lessons Learned

The most important lesson learned in the case study was that our approach helps elicit tacit knowledge, although not directly. When the obtained concept mappings and traces were presented to the domain expert, he explained why the obtained mapping was correct or not, and in this way provided a lot of information about the application domain. This information was not explicitly stated in the written documents, but nevertheless important. Eliciting of this information without concept mappings or traces would have required highly directed questions, and coming up with such questions requires profound domain knowledge.

Conducting the case study has revealed a number of further important lessons that we shall incorporate into our future work, and which, we believe, have wider relevance for concept mapping and taxonomy-based tracing. The bottom line is that it is not always possible to grasp the whole application domain just by analysing the domain-specific vocabulary. The main implication of this lesson is that there are more indicators for traces between documents than the vocabulary. In particular, the overall goals of the prospective system turned out to be an important trace indicator. More concretely, our domain expert pointed out the following issues that should be taken into consideration:

- Explicit goal modelling
- Definition of relevant concepts
- Mapping of one set of words to several concepts

These points are discussed below in detail.

Explicit goal modelling. When evaluating the automatically produced concept mappings, the domain expert explained every answer to us. The explanations were mostly motivated by system goals: for every concept, he decided first, to which goal the concept in question was most relevant. If two concepts that were mapped to each other were relevant for the same goal, he agreed with the automatically produced mapping.

This observation gives an important insight into the process of decision making: the most important part seems to be not the lexical or even semantic similarity of the concepts, but their pragmatic usage. Automatic capture of pragmatics is far beyond the capabilities of the state-of-the-art in computational linguistics [12]. Nevertheless, explicit goal models can help model pragmatics and should be produced and used, inter alia, for tracing.

Definition of relevant concepts. Several of the produced concept mappings were wrong due to the fact that the most relevant concept, that would provide the best mapping, was not in the set of 150 most significant concepts. As a consequence, we missed some important concepts as potential mapping targets. One of the sources of the problem can be that, in order to save the domain expert's time, we reviewed the set of extracted concept without the domain expert and decided ourselves, which concepts were relevant and which were not.

To mitigate this problem in a real world setting, we would need to perform a thorough review of the extracted concepts with an application domain expert. As the review of the long concept list can be tedious and tiring, a mixed mode might make sense:

- The domain expert names the concepts that are, in her opinion, most relevant for the application domain (brainstorming).
- Additionally, abstraction identification is used to extract concepts.
- Finally, both lists are reviewed and joined to the final concept list.

This would allow us to avoid the problem that concepts important for the domain expert are neglected.

Mapping of one set of words to several concepts. When evaluating the correctness of the produced concept mappings, the domain expert often gave comments like "yes, this mapping is correct, but mappings X, Y, and Z would be correct too". This implies that there is no gold standard in concept mapping, at least in our case study. The consequence for concept mapping and document tracing is that we should investigate how close the similarity metric for different concept pairs are. One research question would be whether all mappings that the domain expert evaluated as possible get close similarity scores, and conversely, non-related concepts get distinctly different scores. A follow-up research question would relate to the influence of additional mappings on tracing: with more than one map for a set of words, we can provide more traces between documents. The research question would be how this extension influences the precision of the method.

To summarise, our results show that concept mapping and document tracing can not only be used in software development projects to obtain unspecified information, but also give rise to a set of further interesting research questions.

VI. Related Work

On the technical side, work on WordNet-based similarity metrics is most close to the presented paper. Such metric were introduced, for example, by Leacock and

Chodorow [13], Resnik [14], Lin [15], and Jiang and Conrath [16] The existing similarity metrics are, however, focused on defining similarities between two single words, whereas we are interested in multiword concepts.

In the requirements engineering context, requirements tracing is related to our work. Requirements tracing attracted significant attention in the 1990s, spurred by the seminal work of Gotel and Finkelstein whose empirical research [17] highlighted its impact on practice. Their work on *contribution structures* [18] was conceived to make explicit how the actors contribute to the requirements' formulation. Pohl's contribution [19] was to highlight the critical issue of *agreement* in the formulation process, representing it, along with *specification* and *representation*, as one of the three dimensions of requirements engineering. Pohl also argued that tracing needed to be, as far as possible, automated, while recognizing that the nature of the information that had to be manipulated made full automation impossible.

Tracing is often viewed primarily as a means to trace artifacts to each other in order to track how derived requirements and components contribute to satisfaction of the user requirements. Ramesh and Jarke [20], [21] call this basic tracing utility *low-end* tracing. They note that richer information about relationships between artifacts can be captured, which they term *high-end* tracing. The activity of deriving traces between a rich set of documents that might be used by (e.g. domain descriptions), if not necessarily all generated by (e.g. the requirements specification) the requirements process, is an example of high-end tracing. Being able to trace between heterogeneous documents can have a variety of benefits such as the discovery and recording of requirements' rationale.

However, even low-end tracing is commonly neglected because "requirements tracing is at best a mundane, mind numbing activity" [22] the benefits of which only appear later in the development process. Consequently, even if mandated by process standards, tracing is vulnerable to neglect. Even if the relationships between requirements and downstream artifacts are recorded, they may not be properly maintained as the requirements and other artifacts evolve. Effective high-end tracing is even more of a problem. These difficulties in effecting good requirements tracing practice have motivated several researchers [23], [24], [25], [26], [27] to investigate automatic tracing as a way to mitigate prevalent poor practice. Automatic tracing is the post-hoc discovery of relationships between artifacts, sometimes called *link generation*. Automatic tracing, without the overhead associated with the on-going manual recording and maintenance of trace information, has clear appeal.

In recent years, several researchers have applied information retrieval (IR) techniques to automatic tracing. IR is concerned with the identification of information in natural language documents for applications that include automatic abstracting and search engines. IR has developed the key concept of *document similarity*. Several robust techniques have been devised for calculating document similarity. In a typical usage scenario, the document similarity is calculated between a *query*, which may be anything from a string of keywords to a large document, and a set of target documents to see which document(s) best match the query. Applying IR techniques to automatic tracing means that artifacts such as requirements are treated as documents, allowing similarity measures to be calculated between requirements, and between requirements and other downstream artifacts. Vector-space models, term frequency-inverse document frequency (TF-IDF) and cosine similarity have all proven effective for low-end tracing, by inferring the existence of *derives* relationships between requirements [2], and between requirements and other artifacts such as test cases [26].

Experiments using requirements data sets benchmarked against manually-derived *gold standard* sets of trace relationships, [2], [28], [3] show that IR-based trace-recovery tools are capable of discovering up to 90% of derives relationships. Precision is typically significantly poorer but this is justified by the fact that human analysts are better at identifying errors of commission than errors of omission [3]. The practical effect of this is to value recall above precision.

The performance that is achievable by IR-based tools is sensitive to properties of the requirements documents to which the tools are applied. In experiments on different data sets, Cleland-Huang *et al* [2] note that their *Poirot* tool achieved performance that varied from 90% recall with 30% precision to around 60% recall with only 4% precision. This variation in performance appears to correlate with properties of the requirements documents.

One desirable document property is consistency across all requirements documents and across all levels of abstraction. However, this is commonly not achieved. In particular, high-end tracing may seek to establish trace relationships between the requirements formulated for the system-to-be and the structural, behavioural and organisational properties described in (e.g.) general domain descriptions, process descriptions and concepts of operation. There may also be 'raw' elicitation material such as verbatim reports of elicitation interviews. This heterogeneity of requirements documents reflects the complexity of the requirements formulation process itself; a complex combination of Pohl's agreement and specification dimensions. Requirements are seldom elicited ready-formed from the stakeholders. Rather, requirements are formulated from information about what Alexander and Robertson [29] term *our system*, the *containing system* and the *wider environment*, the needs of the stakeholders in each of these systems, the factors that constrain the possible solutions and the analyst's knowledge and experience. Moreover, the roles of the analyst and stakeholders in requirements formulation may not be clear-cut. The requirements formulation process is further complicated by the fact that some sources are intangible. As noted by

Ryan "neither informal speech nor natural language text is capable of expressing unambiguously the myriad facts and behaviours that are included in large scale systems" [30]. In particular, the assumptions used to fill the inevitable gaps in the available knowledge and the judgement and creativity that is so vital to the requirements formulation process [31] may have no physical, textual or other representation. In combination, the nature of requirements sources and the complexity of the requirements formulation process make it both difficult and tedious to identify the user requirements' sources.

One tangible result of the complexity of requirements formulation is that it is unrealistic to expect complete consistency of terminology across all the documents that form part of the wider set of requirements artifacts typically used in a project. For example, there may be several terms that are used to signify a common underlying concept. While a project dictionary may be definable for the requirements specification document, it will typically be derived only after the process of formulating the user requirements, and no assumptions about adherence to the terms in the dictionary can be made on behalf of many actors, particularly those involved earlier in the requirements process. Although inconsistent use of terminology is only one of the problems posed by high-end tracing, synonymy (where different words represent the same underlying concept) significantly complicates automatic tracing for the reasons described above.

Latent semantic analysis (LSA) is capable of tolerating a degree of synonymy (and polysemy) and has been used for tracing requirements provenance [32]; a high-end tracing task that seeks to identify the sources of user requirements formulation by generating links between the user requirements and (e.g.) elicitation interview statements. However, LSA is computationally expensive and does not work well where the requirements statements are terse. Automatic tracing, particularly in support of high-end tracing, is therefore a tough research problem that currently has few solutions. Our work presents a contribution in this direction.

VII. Future Work

One of the lessons learned in the case study was that, even though the automatically calculated similarity metric correlate with the answers of the human analyst, the best concept mappings are sometimes not contained in the set of extracted abstractions. This means that even the concept with the highest similarity would not provide the best mapping, as the best mapping is not in the set of mapping targets.

For this reason, manual concept mapping should be available too. This would involve a graphical interface, currently under development, that will provide support for the identification of relationships between abstractions manually. The graphical interface will provide the user with a set of sentences extracted from the text. The significant abstractions occurring in the sentences will be highlighted.

A screenshot of the graphical interface prototype for relationships identification is presented in Figure 3. In the left panel, documents are shown sentence by sentence. The user selects one or more sentences, and the tool presents the set of abstractions that occur in the selected sentences in the right panel at the bottom. At the top of the right panel, a description of a relationship is presented. The user can modify both abstractions and the relationship. Based on the similarity metric presented in this paper, we could go beyond purely manual identification of relationships, and suggest concepts with high similarity to the user. This would not preclude, however, manual identification of relationships.

With the prototype presented in Figure 3 the user is given a possibility to select two concepts in order to set a relationship between them. Then, the user will have several options to set a relationship:

- Selecting a relationship from a predefined set of relationships, such as "is-a", and "is-part-of", in case that a generic relation is identified.
- Using the verbs that occur between both abstractions found in the text document, which describe an explicit relationship between them. For instance, in the sentence: "Mr B has already used the portal to check his appointment details.", two abstractions are identified: "Mr B", and "portal". In this case, the verb "use" is identified between them and stated as the relationships between the concepts, as follows: "Mr B uses the portal".
- Providing a new relationship defined by the user, in case that a new relationship is inferred but not explicit in the text document. For instance, the sentence "Miss X notices an email from the Programme Director for the course she is applying for thanking her for the application". signifies a relationship "Miss X's application for the postgraduate program has been received". Identification of such relationships cannot yet be automated, so we give the user the possibility to set arbitrary relationships manually.

In terms of future research, we should compare the performance of the manual and the automatic concept mapping. The comparison should take place not only on the case study presented in the paper but on larger case studies too.

VIII. Conclusion

Requirements knowledge is manyfold, which may be one of the reasons that some knowledge is forgotten when requirements are written down. Furthermore, some knowledge may be simply considered by document authors not important enough to be documented. As errors of omission are more difficult to identify than errors of commission, detection of undocumented knowledge can be tedious.

Our approach, presented in this paper, detects undocumented knowledge pertinent to two aspects:

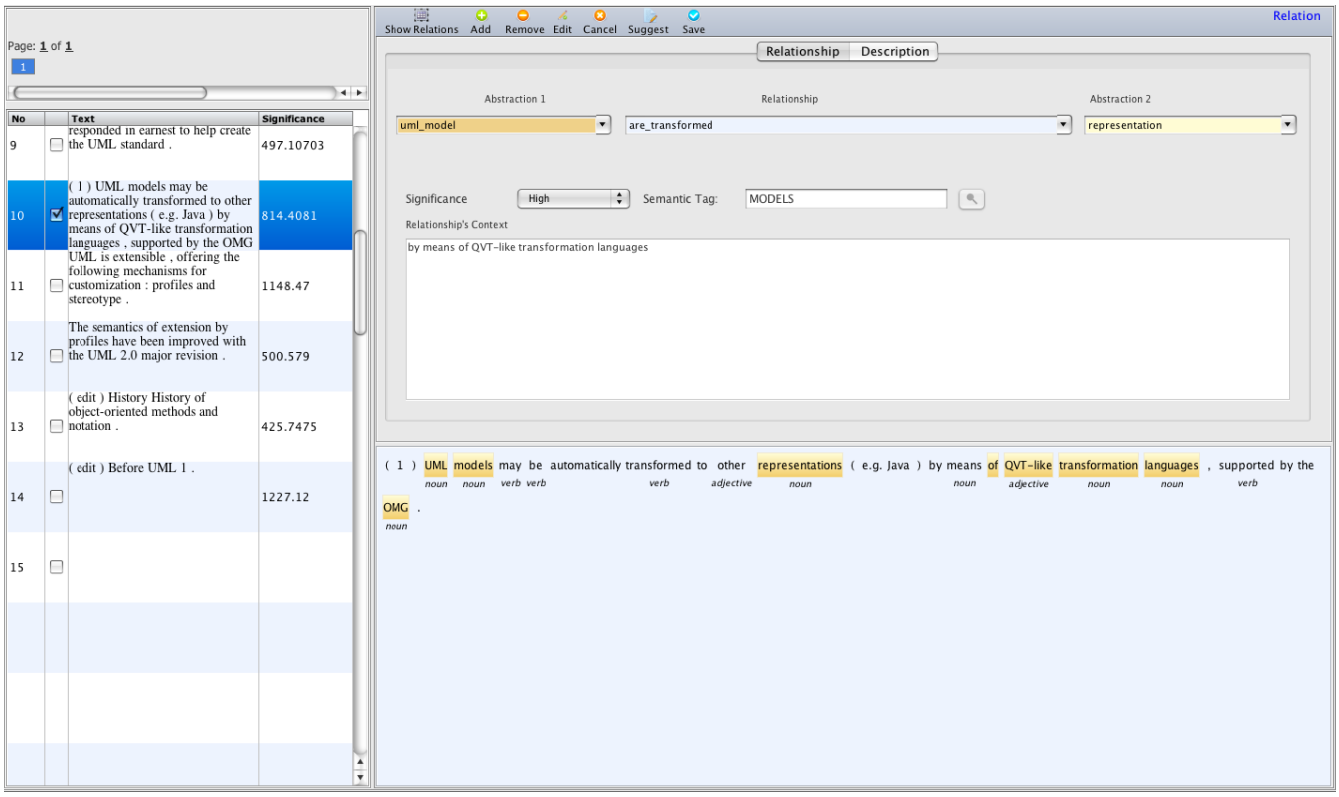

Figure 3. Manual identification of concept mappings

- The approach shows which application domain concepts are potentially related to each other. Even though our case study was limited to two abstraction levels, the presented technique is able to cope with pairwise mappings on many different abstraction levels.
- The approach detects links between documents or their passages.

This knowledge can be important on its own, just in order to give the analyst more orientation in the application domain than she gets from explicit specifications. More importantly, this knowledge can guide an interview with a domain expert. In this respect, it provides an important contribution in two directions: both by showing the domain expert what else should be documented, and by giving the analyst explicit additional knowledge about the application domain. Just by this fact of fostering communication our approach provides an important contribution to managing requirements knowledge.

ACKNOWLEDGMENTS

We are grateful to Paul Holland for his help in the evaluation and for his valuable feedback on our method.

REFERENCES

[1] L. Mich, M. Franch, and P. Novi Inverardi, "Market research on requirements analysis using linguistic tools," *Requirements Engineering*, vol. 9, no. 1, pp. 40–56, 2004.

[2] J. Cleland-Huang, R. Settimi, E. Romanova, B. Berenbach, and S. Clark, "Best practices for automated traceability," *IEEE Computer*, vol. 40, no. 6, pp. 27–35, 2007.

[3] J. Huffman-Hayes, A. Dekhtyar, and S. K. Sundaram, "Advancing candidate link generation for requirements tracing: The study of methods," *IEEE Transactions on Software Engineering*, vol. 32, no. 1, pp. 4–19, 2006.

[4] N. Maiden and G. Rugg, "ACRE: selecting methods for requirements acquisition," *Software Engineering Journal*, vol. 11, no. 3, 1996.

[5] N. E. Fuchs, "Attempto - controlled english as a specification language," http://www.ifi.unizh.ch/attempto/, accessed 30.06.2008.

[6] G. Aston and L. Burnard, *The BNC Handbook: Exploring the British National Corpus with SARA*. Edinburgh University Press, 1998.

[7] P. Sawyer, P. Rayson, and K. Cosh, "Shallow knowledge as an aid to deep understanding in early phase requirements engineering," *IEEE Trans. Softw. Eng.*, vol. 31, no. 11, pp. 969–981, 2005.

[8] R. Gacitua, P. Sawyer, and P. Rayson, "A flexible framework to experiment with ontology learning techniques," *Knowledge-Based Systems*, vol. 21, no. 3, pp. 192 – 199, 2008, aI 2007, The 27th SGAI

International Conference on Artificial Intelligence. [Online]. Available: http://www.sciencedirect.com/science/article/B6V0P-4R6B2HC-F/2/549a3ec9b4320c97e5a572e30be5fa97

[9] R. Gacitua, P. Sawyer, and V. Gervasi, "On the effectiveness of abstractions identification in requirement engineering," in *RE'10: Proceedings of the 18th IEEE International Conference on Requirements Engineering (RE'10)*. Washington, DC, USA: IEEE Computer Society, 2010, to appear.

[10] G. A. Miller, C. Fellbaum, R. Tengi, S. Wolff, P. Wakefield, H. Langone, and B. Haskell, "WordNet," 2005, http://wordnet.princeton.edu/, accessed 12.06.2010.

[11] L. Kof, R. Gacitua, P. Sawyer, and M. Rouncefield, "Ontology and model alignment as a means for requirements validation," in *ICSC 2010, Fourth IEEE International Conference on Semantic Computing*. Washington, DC, USA: IEEE Computer Society, 2010, submitted.

[12] L. Kof, "On the identification of goals in stakeholders' dialogs," in *Innovations for Requirement Analysis. From Stakeholders' Needs to Formal Designs: 14th Monterey Workshop 2007, Monterey, CA, USA, September 10-13, 2007. Revised Selected Papers*, ser. LNCS, vol. 5320. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 161–181.

[13] C. Leacock and M. Chodorow, *Combining Local Context and WordNet Similarity for Word Sense Identification*. The MIT Press, May 1998, ch. 11, pp. 265–283.

[14] P. Resnik, "Using information content to evaluate semantic similarity in a taxonomy," in *IJCAI'95: Proceedings of the 14th international joint conference on Artificial intelligence*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1995, pp. 448–453.

[15] D. Lin, "An information-theoretic definition of similarity," in *ICML '98: Proceedings of the Fifteenth International Conference on Machine Learning*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1998, pp. 296–304.

[16] J. J. Jiang and D. W. Conrath, "Semantic similarity based on corpus statistics and lexical taxonomy," *CoRR*, vol. cmp-lg/9709008, 1997.

[17] O. Gotel and A. C. W. Finkelstein, "An analysis of the requirements traceability problem," in *First International Conference on Requirements Engineering (ICRE)*. IEEE Computer Society Press, April 1994, pp. 94–101. [Online]. Available: citeseer.ifi.unizh.ch/article/gotel94analysis.html

[18] O. Gotel and A. Finkelstein, "Contribution structures," in *Proceedings: 2nd IEEE International Symposium on Requirements Engineering*. IEEE Computer Society Press, 1995, pp. 100–107.

[19] K. Pohl, "PRO-ART: Enabling requirements pre-traceability," in *Proceedings: 2nd International Conference on Requirements Engineering*. IEEE Computer Society Press, 1996, pp. 76–84.

[20] B. Ramesh and M. Jarke, "Toward reference models for requirements traceability," *IEEE Trans. Software Engineering*, vol. 27, no. 1, pp. 58–93, 2001.

[21] B. Ramesh, C. Stubbs, T. Powers, and M. Edwards, "Requirements traceability: Theory and practice," *Annals of Software Engineering*, vol. 3, pp. 397–415, 1997.

[22] J. Huffman-Hayes, A. Dekhtyar, and J. Osborne, "Improving requirements tracing via information retrieval," in *Procerdings of 11th IEEE International Requierments Engineering Conference*, 2003, p. 138.

[23] I. Alexander, "Towards automatic traceability in industrial practice," in *1st International Workshop on Traceability*, Edinburgh, UK, 2002, pp. 26–31.

[24] J. Cleland-Huang, C. K. Chang, G. Sethi, K. Javvaji, H. Hu, and J. Xia, "Automating speculative queries through event-based requirements traceability," in *RE '02: Proceedings of the 10th Anniversary IEEE Joint International Conference on Requirements Engineering*. Washington, DC, USA: IEEE Computer Society, 2002, pp. 289–298.

[25] J. Lin, C. C. Lin, J. Cleland-Huang, R. Settimi, J. Amaya, G. Bedford, B. Berenbach, O. B. Khadra, C. Duan, and X. Zou, "Poirot: A distributed tool supporting enterprise-wide automated traceability," in *Proceedings of the 14th IEEE International Requirements Engineering Conference (RE'06)*. Washington, DC, USA: IEEE Computer Society, 2006, pp. 356–357.

[26] M. Lormans and A. van Deursen, "Reconstructing requirements coverage views from design and test using traceability recovery via lsi," in *TEFSE '05: Proceedings of the 3rd international workshop on Traceability in emerging forms of software engineering*. New York, NY, USA: ACM Press, 2005, pp. 37–42.

[27] A. D. Lucia, F. Fasano, R. Oliveto, and G. Tortora, "Can information retrieval techniques effectively support traceability link recovery?" in *Proceedings of the 14th IEEE International Conference on Program Comprehension (ICPC'06)*. Washington, DC, USA: IEEE Computer Society, 2006, pp. 307–316.

[28] J. Natt och Dag, V. Gervasi, S. Brinkkemper, and B. Regnell, "A linguistic-engineering approach to large-scale requirements management," *IEEE Software*, vol. 22, no. 1, pp. 32–39, Jan./Feb. 2005.

[29] S. R. Ian Alexander, "Understanding project sociology by modeling stakeholders," *IEEE Software*, vol. 21, no. 1, pp. 23–27, 2004.

[30] K. Ryan, "The role of natural language in requirements engineering," in *Proceedings of the IEEE Int. Symposium on RE*, 1993, pp. 80–82. [Online]. Available: citeseer.ifi.unizh.ch/ryan93role.html

[31] N. Maiden, A. Gizikis, and S. Robertson, "Provoking creativity: Imagine what your requirements could be like," *IEEE Software*, vol. 21, no. 5, pp. 68–75, Sep./Oct. 2004.

[32] A. Stone and P. Sawyer, "Identifying tacit knowledge-based requirements," *IEE Proceedings - Software*, vol. 153, no. 6, pp. 211–218, 2006. [Online]. Available: http://link.aip.org/link/?IPS/153/211/1