Enhancing Candidate Link Generation for Requirements Tracing: The Cluster Hypothesis Revisited

Nan Niu Anas Mahmoud
Department of Computer Science and Engineering, Mississippi State University, USA
niu@cse.msstate.edu, amm560@msstate.edu

***Abstract*—Modern requirements tracing tools employ information retrieval methods to automatically generate candidate links. Due to the inherent trade-off between recall and precision, such methods cannot achieve a high coverage without also retrieving a great number of false positives, causing a significant drop in result accuracy. In this paper, we propose an approach to improving the quality of candidate link generation for the requirements tracing process. We base our research on the cluster hypothesis which suggests that correct and incorrect links can be grouped in high-quality and low-quality clusters respectively. Result accuracy can thus be enhanced by identifying and filtering out low-quality clusters. We describe our approach by investigating three open-source datasets, and further evaluate our work through an industrial study. The results show that our approach outperforms a baseline pruning strategy and that improvements are still possible.**

***Keywords*-traceability; requirements tracing; clustering**

I. Introduction

In their seminal paper [1], Gotel and Finkelstein defined *traceability* as the "ability to describe and follow the life of a requirement, in both a backward and forward direction". Traceability has since been shown to be critical for a wide variety of software engineering activities, including verification and validation (V&V), risk assessment, and change impact analysis [2]. However, practitioners often fail to implement consistent and effective traceability processes if the traces are maintained manually [1, 3].

One way to overcome the problem is to automatically generate traceability links by exploiting information retrieval (IR) methods, e.g., the vector space model (VSM) [2], probabilistic network model (PN) [3, 4], and latent semantic indexing (LSI) [5]. These methods compute similarity scores between pairs of artifacts, and rank the retrieved traceability links based on the similarity to the query requirement. Extensive empirical studies have been conducted to evaluate the effectiveness of different IR-based automated tracing techniques. Converging evidence indicates that all the exploited methods so far are almost equivalent in that they are able to capture almost the same traceability information [6]. In most cases, a recall of 90% is achievable at precision levels of 5-30% [2, 3, 4, 5, 6, 7], where recall measures coverage and is defined as the percentage of correct links that are retrieved, and precision measures accuracy and is defined as the percentage of retrieved links that are correct [8].

Currently, IR-based tracing tools favor recall over precision. This is mainly because commission errors (false positives) are easier to deal with than omission errors (false negatives) [2]. However, retrieving an excessive number of links can seriously affect the practicality of such tools. A tool that always returns all possible links is guaranteed to have a 100% recall, but is practically useless. Researchers have therefore focused on retrieving and ranking the correct traceability links in the upper part of the result list, so that the human analyst can save effort by vetting only the top-ranked subset [4, 9]. The links with similarity scores above a certain threshold (cutoff) value are called *candidate links* [2, 10]. For example, a threshold of 0.3 was used in [11]; other approaches (e.g., [2, 4, 9]) have evaluated different cutoff values.

Determining threshold in practice can be challenging: Using a low threshold value retrieves a larger number of correct links than using a high value, but more incorrect links are captured at the same time [11]. Among the many performance enhancement techniques (e.g., [5, 12]), no approaches to date can largely decrease false positives at low cutoff points and significantly increase correct links at high cutoff points [11].

In this paper, we aim to enhance IR-based candidate link generation by examining the cluster hypothesis [8], which states that relevant documents tend to be more similar to each other than to irrelevant documents. When adapted to traceability, the hypothesis suggests that correct and incorrect links can be grouped in high-quality and low-quality clusters respectively. Thus, the performance of IR-based tracing can be enhanced by selecting candidate links from high-quality clusters. It is our conjecture that discarding the links from low-quality clusters helps tackle the threshold determination problem.

Prior work in this area has employed clustering as an alternative to the ranked-list presentation to increase the understandability of the candidate links [10]; however, recall and precision are unchanged before and after clustering. In contrast, we leverage clustering to directly improve precision by reordering the retrieved traceability links. Some efforts have also combined clustering with other enhancement techniques [4, 11, 12]; however, the clustering effect is often implicit in these integrated approaches. In contrast,

978-1-4673-2785-5/12/$31.00 © 2012 IEEE

RE 2012, Chicago, Illinois, USA

we perform thorough analysis of the underlying clustering algorithms, adopt novel metrics to explore the interplay of key clustering parameters, and conduct rigorous evaluations on both open-source and proprietary projects.

The contributions of our work lie in the development of a set of detailed procedures to examine the cluster hypothesis in the context of IR-based requirements tracing. Our work not only advances the fundamental understanding about the role clustering plays in traceability, but also enables principled ways to increase the practicality of automated tracing tools. In what follows, we discuss how clustering has been employed in the related literature in Section II. We then present our central hypothesis and specific research questions in Section III. We use three open-source traceability datasets from different domains to answer the research questions in Section IV., and further evaluate the benefits of our approach through an industrial study in Section V. We conclude the paper with a summary and some directions for future work in Section VI.

II. Background and Related Work

A. Clustering in Information Retrieval

Clustering is an unsupervised learning method which automatically divides data into natural groups based on similarity [13]. There has been extensive research on using clustering to improve IR systems. The basic tenet is known as the *cluster hypothesis*, stating that relevant documents tend to cluster near other relevant documents and farther away from irrelevant ones [8]. We discuss related work in IR from three perspectives: document, query, and search results.

On the document side, clustering is mainly used to increase retrieval efficiency. As the number of documents increases, matching the query to all documents can degrade the system performance. A solution is to build a clustering of the entire collection and then match the query to the cluster centroids [14]. In this way, clustering is done in advance so as to enhance the performance at search time. This is referred to as *static clustering* as only one fixed clustering is made to accommodate all user queries. Hearst and Pedersen [15], in designing the Scatter/Gather system, performed *dynamic clustering*, in which document clusters are formed dynamically depending on the user query. The evaluations demonstrated that Scatter/Gather consistently improved retrieval results and that users were able to distinguish high-quality clusters from low-quality ones [15].

On the query side, clustering is particularly useful for overcoming the well-known "short query" problem where the user input provides insufficient information. In this context, previously encountered queries are collected and placed into groups. For example, Yi and Maghoul [16] presented an algorithm to compute equivalent classes of queries (i.e., query clusters) and tested the algorithm on a Web search dataset consisting of over 16M unique queries. For a new incoming query, its equivalent class helps to expand the query with terms that reflect similar information needs [16]. In cases where the new query is not similar to any of the document cluster centroids, it may instead be similar to one of the query groups, which in turn can be used to match document clusters [14].

On the result side, clustering is widely applied to support user's interactive browsing [15, 17, 18]. In a ranked list presentation, the search results are isolated from their context. Organizing and displaying the retrieved artifacts in topic-coherent clusters can facilitate the comprehension and evaluation of the search results. However, in order to effectively provide the contextual information (e.g., generating cluster labels and determining the ordering among the result clusters), human intervention is still required. In fact, a series of experiments showed that result clustering could be as effective as the interactive relevance feedback based on query expansion [18]. Clustering's primary benefit, then, arises from the sense of control it offers to the user over the relevance feedback process [18].

B. Clustering in Traceability

Despite the broad role clustering plays in supporting general IR systems, the application of clustering in traceability has emerged only recently. We discuss here first the seminal work by Duan and Cleland-Huang [10] on using clustering to improve the comprehension and evaluation of the retrieved traceability links, followed by the hybrid approaches [4, 11, 12] that synthesize clustering and other trace retrieval enhancement strategies.

The work in [10] is among the first to systematically investigate the application of clustering in traceability. The authors identified the characteristics of automated trace generation that are significantly different from those of document retrieval.

- On the document side, the searchable artifacts in tracing consist of individual requirements, classes, test cases, etc. Such a collection tends to be significantly smaller than the document collection targeted in a typical Web search or online library search [10]. Therefore, the need of reducing the search space via document clustering is less pressing in requirements tracing.
- On the query side, a trace query is composed from the text of a requirement or other software artifact. Although ideally every requirement is concise, primitive, and unambiguous, the reality is that requirements can often be relatively long and may also contain superfluous information [10]. This, on one hand, reduces the precision of the retrieved results, but on the other hand, mitigates the "short query" problem in IR. Thus, query-side clustering is less useful in requirements tracing.

In addition, it is worth emphasizing that it is essential for automated traceability to strive for the highest possible recall, whereas for many general IR queries, it is sufficient to

locate and return a good subset of relevant documents [10], partly due to the challenge associated with searching through the whole document collection and the opportunity offered by query expansion in IR.

For these reasons, Duan and Cleland-Huang [10] focused on the result-side clustering. The main goal was to increase understandability and reduce the human effort needed to evaluate a set of candidate links. They compared a set of representative clustering algorithms on three traceability datasets, proposed heuristics to determine optimal clustering granularity, and developed metrics to rank the trace clusters. The evaluation on tracing five business requirements showed that clustering traceability links led to less number of decision points than presenting links in an ordered list, thereby saving the effort needed to evaluate the candidate links [10].

While result-side clustering improves comprehensibility, it is the same set of candidate links that is retrieved by non-clustering techniques and the clustering-enabled method described in [10]. In other words, recall and precision are unaffected before and after clustering. In attempts to retrieve higher quality links than the baseline IR method, clustering has also been studied. For instance, Cleland-Huang *et al.* [4] introduced three enhancement strategies (hierarchical modeling, logical clustering of artifacts, and semi-automated pruning) to improve PN, Chen and Grundy [11] described three supporting techniques (regular expression, key phrases, and clustering) to improve VSM, and Wang *et al.* [12] presented four strategies (source code clustering, identifier classifying, similarity thesaurus, and hierarchical structure enhancement) to improve LSI.

Although the integrated strategies are promising, the studies showed mixed results, e.g., the approach in [12] resulted in higher precision but lower recall than LSI. However, perhaps the more important question unanswered in these studies is the enhancement effect resulted from clustering alone rather than from the combined approaches. Lack of such knowledge limits our understanding of clustering's strengths and weaknesses, which in turn impedes the selection of appropriate clustering techniques to be complemented with other types of strategies. Understanding the role clustering plays in enhancing candidate link generation requires rigorous empirical inquires. This is precisely the focus of our research.

III. Research Methodology

A. Central Hypothesis

The cluster hypothesis has been shown true on multiple occasions [4, 10, 11, 12, 14, 15, 17, 18]. When applied in requirements tracing, the cluster hypothesis suggests that correct links tend to be more similar to each other than to incorrect links. This motivates us to formulate our *central hypothesis* — clustering algorithms and optimal clustering granularity exist in grouping correct and incorrect links into high-quality and low-quality clusters respectively. The overarching goal of our research, which is to capture all potential correct traceability links and few incorrect ones, can be accomplished by discovering the appropriate clustering mechanisms, distinguishing between high- and low-quality clusters, and filtering out the links in low-quality clusters.

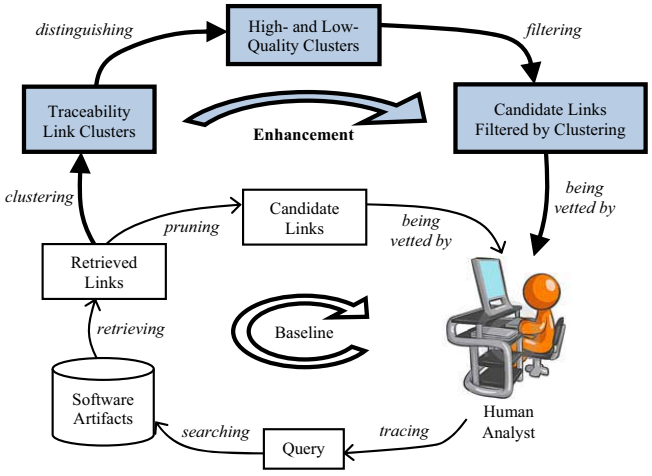

Figure 1. A clustering-based approach to enhancing candidate link generation for requirements tracing. The bottom "baseline" cycle illustrates a conventional IR-based tracing process. The three shaded boxes on the top introduce the "enhancement" steps derived from the cluster hypothesis.

Fig. 1 shows our clustering-based enhancement approach in the context of a baseline tracing process. The baseline process is commonly adopted in state-of-the-art tracing tools like RETRO [2] and Poirot [3]. The human analyst selects some requirements artifact to trace, and the IR algorithm retrieves the traceability links by computing the similarity between the query and the software artifacts in the repository. Pruning aims to (semi-)automatically discard the portion of the retrieved results with a low density of correct links. If this portion were presented to the analyst, then the effort required to discard false positives would become much higher than the effort to validate correct links. For this reason, most approaches use some method to cut (or filter) the result list, thus presenting the human analyst only the subset of retrieved links (or candidate links). Typical pruning strategies include: 1) an absolute threshold on the similarity value, e.g., links whose similarity scores are greater than or equal to 0.3 are chosen as candidate links in [11]; and 2) a relative cutoff point for the proportion of retrieved results, e.g., 70% of the most similar links are selected in [19].

The clustering-based enhancement that we propose is highlighted using shaded boxes in Fig. 1. We discuss the key aspects of each step as follows.

- We perform clustering *after* the initial search is completed. This makes our clustering dynamic rather than static, i.e., we do *not* assume that if two artifacts L_1 and L_2 are both correct or incorrect links for requirement R_A, they must *both* be correct or incorrect links for R_B. The link clusters produced in our approach are query-dependent, and therefore have the potential to

be closely tailored to the characteristics of the specific requirement being traced.

- Upon the identification of a clustering that divides correct and incorrect links into separate groups, it is crucial to research automated ways to differentiate between high- and low-quality clusters. We propose several heuristics and test their performance empirically.
- Filtering in our approach does not rely solely on a link's similarity to the query, but takes into account the cluster the link belongs to as well. In other words, a link's neighbors also define its relevance. Our filtering is thus performed on a cluster basis, which is fundamentally different from the baseline pruning strategy of acting on individual links according to their similarity scores or rankings.

B. Research Questions

The overall goal of our research is to assess the extent to which the cluster hypothesis can be leveraged to enhance automated traceability. We refine our general goal with a set of specific research questions (cf. Fig. 1).

Clustering. What is the clustering algorithm that best separates correct and incorrect links into different groups? What is the optimal granularity (e.g., the optimal number of resulting clusters) to do so?

Distinguishing. What is the most adequate heuristic in uncovering the quality of traceability link clusters?

Filtering. As the links in low-quality clusters are removed, what is the most appropriate way to arrange the remaining candidate links?

The answers to the research questions will enable the discovery of the underlying clustering mechanisms, along with the supporting strategies, for improving candidate link generation. Next, we present an experimental study that seeks to answer these research questions.

IV. Experimental Investigation

In most cases, identifying the best settings for effectively integrating clustering in a working application is viewed as an NP-complete optimization problem [20]. For this reason, we are compelled to make some assumptions about the general clustering features in order to achieve an acceptable approximation. In particular, we restrict our discussion to mutually exclusive and collectively exhaustive clustering, which allows each of the retrieved traceability links to be assigned to one and only one cluster.

Three datasets are used in our investigation. iTrust[1] is a medical application developed by the students from North Carolina State University (USA). eTour[2] is an electronic tour guide built by the final year students at the University of Salerno (Italy). CM-1[3] contains a complete set of requirements (high-level) and design (low-level) documents for a NASA scientific instrument. Table I shows the characteristics of these datasets. While all projects are of medium size (about 20K lines of code), they are drawn from different application domains and are written in different languages with different coding conventions. The traceability information also exhibits important variability: both requirements-to-source-code (iTrust and eTour) and requirements-to-design (CM-1) traces are examined. We use these datasets to investigate the 3 enhancement procedures in our approach. For each procedure, we state the experimental setting, describe the evaluation method, and present the result analysis. We discuss the threats to validity in Section IV-D.

Table I
Characteristics of the experimental datasets

Data -set	General Information			Traceability Information*		
	Domain	Prog. Lang.	Lines of Code	|Source|	|Target|	|Correct Links|
iTrust	Healthcare	Java	18.3K	38	226	314
eTour	Tourism	Java	17.5K	58	116	394
CM-1	Aerospace	C	20.0K	235	220	361

* Requirements-to-source-code traces (usecases-to-classes) are studied in iTrust and eTour, whereas requirements-to-design traces are studied in CM-1.

A. Clustering Traceability Links

Experimental Setting. In one of the most systematic comparisons of traceability link clustering [10], three algorithms, namely hierarchical agglomerative clustering (HAC), k-means, and bisecting divisive clustering ("bisecting" for short), are evaluated. These algorithms cover a wide spectrum of clustering methods available in the literature. They are also known to perform well in documents clustering [13]. We test a similar set of algorithms in our study, including k-means, bisecting, and 3 variants of HAC:

- SL (single-link): $M(A,B)=min\{d(a,b) : a \in A, b \in B\}$.
- CL (complete-link):$M(A,B)=max\{d(a,b) : a \in A, b \in B\}$.
- AL (average-link): $M(A,B)=\frac{1}{|A|\cdot|B|}\sum_{a\in A}\sum_{b\in B} d(a,b)$.

where $d(a,b)$ is the distance between data objects a and b, and $M(A,B)$ defines the linkage (merging) criteria for clusters A and B. For example, SL merges the two clusters with the smallest minimum pairwise distance.

All the five clustering algorithms in our study rely on the terms contained in the software artifacts to compute similarity. For our experiments, the similarity scores are computed using tf-idf cosine [8], a popular scheme in VSM which has been validated through numerous traceability studies [2, 6, 7]. Such a choice, at the same time, defines the baseline tracing mechanism (cf. Fig. 1) in our study to be tf-idf in VSM. We extend our indexer [19] to preprocess both source code and (requirements & design) documents. Three steps are involved: tokenizing, filtering, and stemming. We further extend our TraCter tool [21] by adding and

[1] http://agile.csc.ncsu.edu/iTrust

[2] http://www.cs.wm.edu/semeru/tefse2011

[3] http://promise.site.uottawa.ca/SERepository/datasets/cm1.desc

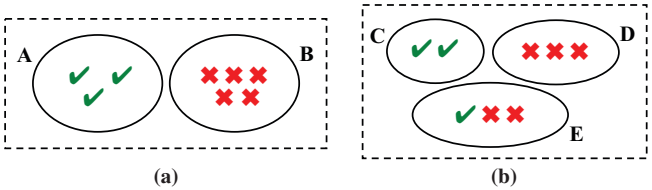

Figure 2. **(a)** Dividing 8 retrieved traceability links into two clusters: **A** contains 3 correct links and **B** contains 5 incorrect ones. **(b)** The same 8 links grouped into three clusters: **C**, **D**, and **E**.

refining the k-means, bisecting, SL, CL, and AL clustering implementations.

When exploring the optimal clustering granularity, two interdependent parameters are usually calibrated: the number of clusters k and the cluster size $|C|$. In our case, the primary driving factor shared by all the five algorithms is k, which we use to gauge the clustering granularity. In terms of $|C|$, especially for k-means and bisecting, we adopt the rule of thumb to generate balanced clusters, i.e., relatively uniformly sized clusters [10].

Evaluation Method. In general, clustering quality can be evaluated either internally or externally [13]. Internal evaluation does not refer to external knowledge (e.g., an authoritative decomposition prepared by human experts) and therefore cannot assess the "goodness" of a clustering method. Rather, internal evaluation compares how close the clusterings are to each other. In [10], for instance, a pairwise correlation analysis was performed between the {HAC, k-means, bisecting} algorithms. The result showed that at reasonable clustering granularity (of 5-6 clusters or higher) no significant difference was observed between the clusters produced by the three algorithms. Internal evaluation is particularly suited for exploratory studies where clustering is conducted to discover patterns in the input data [13].

When evaluated externally, clustering results are compared with a gold standard or classification labels [13]. These ground-truth clusters, also known as authoritative figures, are usually produced by experts or provided based on a natural decomposition of the data. A clustering algorithm can then be evaluated based on how much of the known structure it can recover [13]. External evaluation fits our purpose since our objective is to assess the "goodness" of the clusterings, as well as to determine the optimal clustering granularity.

We devise a gold standard (not *the* gold standard) in our study by directly applying the cluster hypothesis to the automated tracing problem. In an ideal situation, only two clusters should be present: one cluster has a 100% recall and precision (i.e., it contains only and all the correct traceability links), and the other cluster has a 0% recall and precision (i.e., it contains only and all the false positives). Fig. 2a illustrates this ideal situation.

In order to calibrate the clustering granularity k while externally evaluating the clustering results, we adopt the MoJo distance measure [22], the *de facto* metric embraced by the software clustering community. MoJo measures the distance

```
1.  For each c_i ∈ Clustering_Algorithms
2.    Loop (initialize k=2) AND (increment k by 1)
3.      For each d_j ∈ Datasets
4.        For each q_m ∈ Trace_Query(d_j)
5.          RTL_m ← Retrieve_Traceability_Links(q_m)
6.          GS_m ← Produce_Gold_Standard(RTL_m)
7.          CR_{m,i,k} ← Cluster(RTL_m, c_i, k)
8.          MoJo_{m,i,k} ← Computer_MoJo(CR_{m,i,k}, GS_m)
9.        Average_MoJo_{i,k} ← (1/m) Σ_m MoJo_{m,i,k}
10.       BREAK if local_minimum(Average_MoJo_{i,k})
11.   Return min(Average_MoJo_{i,k})
```

Figure 3. Identifying optimal clustering settings.

between two decompositions of the same software system by computing the number of Move and Join operations to transform one to the other. Intuitively, the smaller the MoJo distance, the closer the two clustering results. Take Fig. 2 as an example, the MoJo value of transforming Fig. 2b to Fig. 2a is 2: moving the correct link from **E** to **C** and then joining **D** and the revised **E** together. We integrate MoJo's optimal implementation described in [22] (*release 2.0*[4]) into our current analysis.

Result Analysis. Before discussing the results, it is important to comment on the intrinsic intractability of clustering with constraints [23] so as to better understand our analysis procedure shown in Fig. 3. The problem of separating n data points into k disjoint sets such that pairwise distances within sets are bounded by a constant is known to be NP-complete [20]. Therefore, most formulations of the clustering problem are NP-hard, which means that there cannot be an efficient clustering algorithm that satisfies all constraints for all datasets [23]. However, acceptable approximations can be achieved by using procedures that optimize one or more fitness measures. The main goal is to choose from a large set of possible solutions the one that gives the best value for the selected measures.

Fig. 3 outlines our procedure for finding optimal clustering settings. The measure that we choose to optimize is the MoJo distance averaged over all the trace queries in a given dataset (lines 9-10). That is, if a local minimum of Average_MoJo$_{i,k}$ is reached for the clustering algorithm c_i at granularity level k, then the optimal clustering setting is found and min(Average_MoJo$_{i,k}$) is returned. The procedure will repeat for the next evaluation cycle with a new clustering algorithm c_{i+1}. It is necessary to point out that k is looped from 2. This is because the gold standard that we use consists of 2 clusters (cf. Fig. 2a). Thus, a local minimum near the lower bound of k is considered to be an acceptable solution to our particular optimization problem.

Fig. 4 plots the average MoJo values for all the three traceability datasets. For iTrust, Average_MoJo hits a minimum when using SL to produce 7 clusters. For eTour, SL achieves the best performance at 7-8 clusters. For CM-1, a local minimum of Average_MoJo is obtained when 9 clusters are generated by either CL or SL. From our analysis,

[4]http://www.cse.yorku.ca/~bil/downloads/mojo.tar

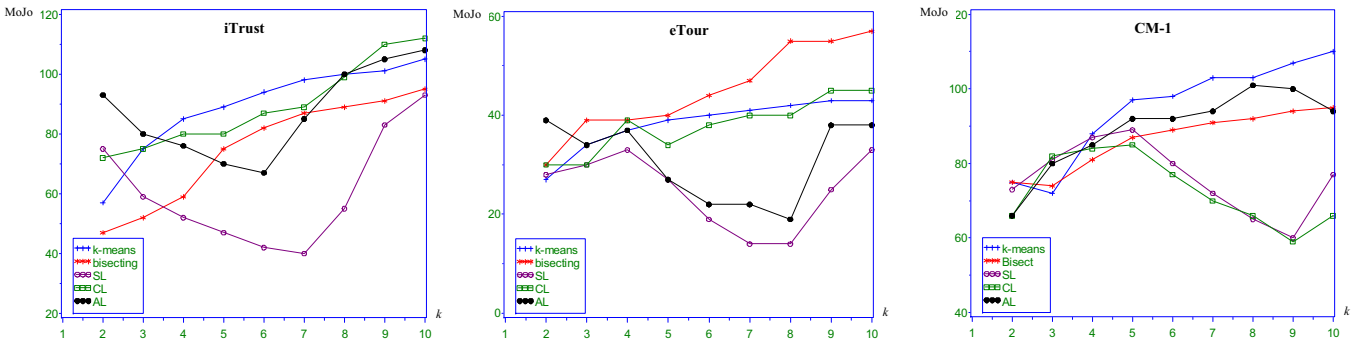

Figure 4. MoJo analysis for comparing the clustering algorithms and exploring the optimal clustering granularity in terms of k.

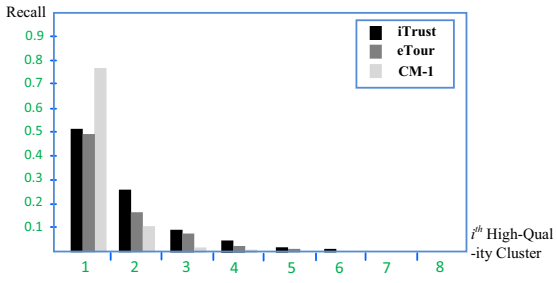

Figure 5. Traceability link clusters arranged based on recall.

SL turns out to be the best candidate for realizing the cluster hypothesis in traceability. The optimal clustering granularity of SL is found to be $k \in [7, 9]$ across the three datasets. These findings differ from the previous observations that k-means, bisecting, and HAC result in similar clusterings when k equals to 5-6 or higher [10]. We speculate this difference is attributed to the different evaluation methods employed: while we externally assess how close the clusterings are to a gold standard, the study in [10] internally evaluates how close the clusterings are to each other.

To further validate the procedure in Fig. 3, we apply the optimal clustering settings (i.e., SL with k=8) to trace all the requirements from each dataset. For every trace query, the resulting 8 SL clusters are arranged in a descending order based on their recall values (the percentage of correct links they contain). The recall values of all the clusters at the same rank (1^{st}, 2^{nd}, etc.) are then averaged over all the trace queries. Fig. 5 shows the results. It is apparent that, when clustered using the optimal settings, the retrieved traceability links are effectively separated in high-quality (high-recall) and low-quality (low-recall) groups. For example, the top-3 clusters in CM-1 contain almost all the correct links. These results provide evidence supporting the findings previously reported in [13, 15], stating that there existed an optimal clustering mechanism such that if the IR system were able to optimize the clustering settings, the mechanism would always perform better than baseline document retrieval.

B. Determining the Quality of Link Clusters

Experimental Setting. In Fig. 5, the cluster quality is judged by using the answer set available for each dataset. However, under non-experimental settings where no answer set is available, a new strategy for determining cluster quality is needed. Automated support for this determination is challenging, e.g., the quality of clusters were determined manually in [15].

We investigate three heuristics to distinguish between high- and low-quality clusters, given the cluster hypothesis holds[5]. These heuristics are motivated by dynamic clustering [15], which emphasizes that the clusterings should be query-specific. Since traceability link clustering in our approach is dynamic, each cluster can be characterized via certain link based on the similarity to the trace query.

- MAX (maximum similarity): The link with the maximum tf-idf similarity to the trace query is selected as a representative for the cluster. Clusters are then arranged based on their MAX representatives.
- AVE (average similarity): Clusters are sorted based on the average tf-idf similarity of their links to the query.
- MED (median similarity): Clusters are sorted based on the median tf-idf similarity of the links to the query.

Evaluation Method. The link clusters' quality hinges on their recall values since automated traceability strives to achieve the highest possible recall. Similar to the iterative procedure listed in Fig. 3, we evaluate the heuristics {MAX, AVE, MED} by closely monitoring the drop rate in recall each time the links from a low-quality cluster are discarded. The heuristic that maintains a consistently high recall during the removal of lowest-quality clusters is then considered to be an effective strategy.

Result Analysis. Fig. 6 compares how recall drops when different heuristics are applied. In all three datasets, MAX is the most successful strategy for determining the quality of link clusters. In other words, it is adequate to use each cluster's link that is the most similar to the query to distinguish high- and low-quality clusters. The results in Fig. 6 also show that the number of lowest-quality clusters to remove varies from dataset to dataset. When the clusters are arranged using MAX, discarding the links from the bottom 3 clusters has little effect on recall in iTrust and eTour. For CM-1, a considerable drop in recall is observed if the 6^{th}

[5]Not surprisingly, we use SL with k=8 in this analysis.

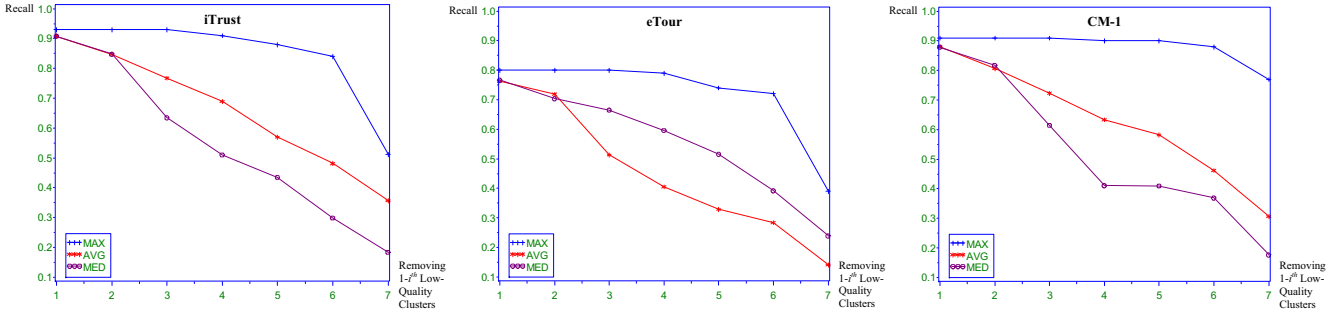

Figure 6. Using different representatives to determine the quality of clusters.

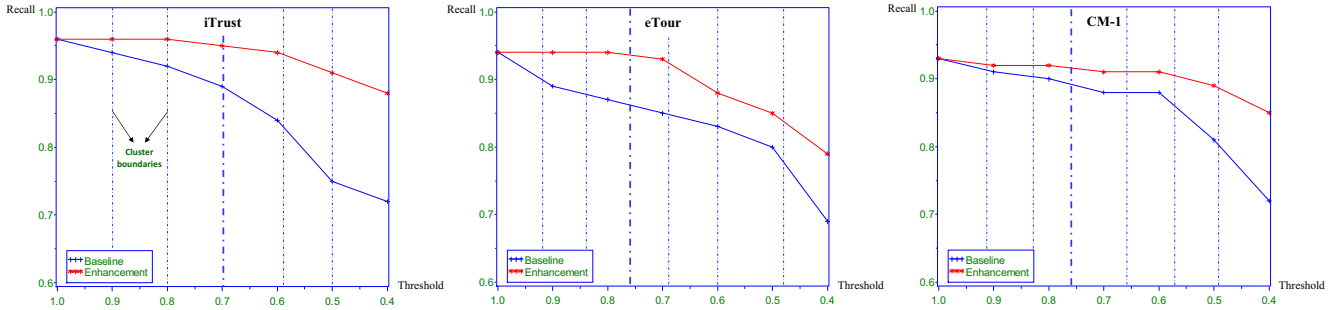

Figure 7. Comparing recall: the threshold (x-axis) follows the baseline pruning unit showing the portion of retrieved links being kept as candidate links.

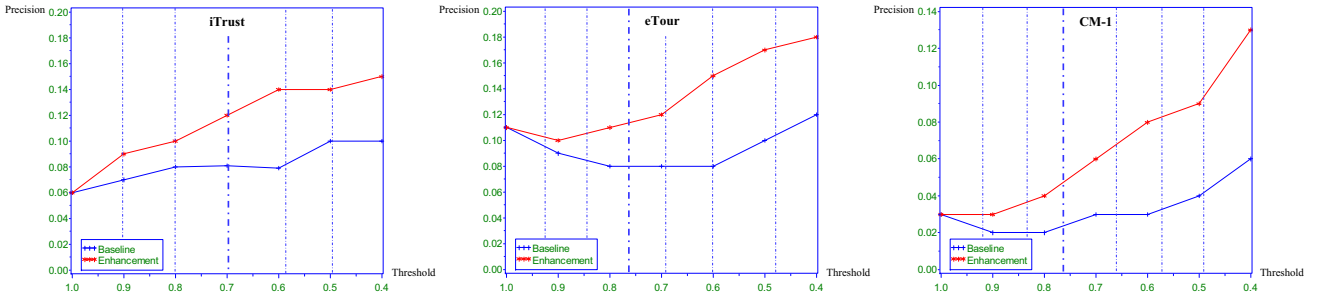

Figure 8. Comparing precision: the threshold (x-axis) follows the baseline pruning unit showing the portion of retrieved links being kept as candidate links.

bottom-ranked cluster is removed. Although CM-1's MAX curve represents a best case scenario, we believe removing 3 lowest-quality clusters is a satisfactory answer to our research question. The links in these clusters are mostly false positives, so discarding them could significantly improve candidate link generation.

C. Generating Candidate Links

Experimental Setting. Two tests are performed at this stage. First, it is crucial to compare the candidate links generated via our clustering-based approach with those produced by the baseline pruning strategy. This test evaluates the *enhancement* effect (cf. Fig. 1). Second, two styles of *presenting* the newly generated candidate links are assessed:

- ABS (using absolute similarity scores): The candidate links are ordered based on the similarity to the trace query regardless of their clusters.
- MAX (preserving cluster boundaries): The candidate links are displayed in their respective clusters, which in turn are arranged by their MAX representatives. Inside each cluster, the links are ranked according to the tf-idf similarity to the trace query.

Evaluation Method. For the first test on enhancement effect, standard recall and precision metrics are used. For the second test, these primary metrics are inappropriate because the two styles are applied to display the same set of candidate links (i.e., the links contained in the 5 highest-quality SL clusters judged by the MAX heuristic). We therefore resort to MAP (mean average precision), a secondary measure useful in evaluating the performance of tracing methods [24]. MAP assesses the quality across the recall levels [8]. Mathematically, let $S = \{s_1, \ldots s_n\}$ and $T = \{t_1, \ldots t_m\}$ be sets of software artifacts. $L = \{(s,t) \mid sim(s,t)\}$ is the set of candidate traceability links, and $L_T \subseteq L$ is the set of correct links. Then, MAP $= \frac{1}{|Q|} \sum_{j=1}^{|Q|} \frac{1}{m_j} \sum_{k=1}^{m_j} \text{Precision}(L_{jT})$. Intuitively, the higher the MAP, the closer the correct links are to the top of the candidate link presentation. The presentation with a higher MAP is superior as the links are easier to browse [24].

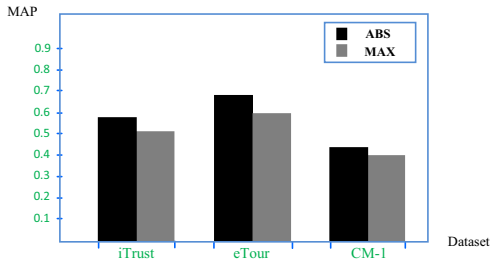

Figure 9. Assessing the browsability of the presentation styles.

Result Analysis. Fig.s 7 and 8 show the recall and precision comparisons respectively. Note that the baseline and our clustering-based enhancement use different units in filtering out false positives. In order to reconcile this difference, the thresholds in Fig.s 7 and 8 are specified by following the baseline pruning strategy, namely, removing the bottom $(x\times100)\%$ of the least similar links in the ranking defined by the links' tf-idf scores, where $x = \{1.0, 0.9, 0.8, \ldots 0.4\}$. To accommodate the analysis of cluster-based filtering proposed in our approach, the size of each cluster is also depicted through the dotted vertical "cluster boundaries" line in Fig.s 7 and 8. The bold dotted line shows the cutoff point for selecting candidate links in our approach, i.e., discarding the links from the 3 lowest-quality clusters and keeping the remaining as the candidate links.

The recall and precision comparisons clearly show that our approach outperforms the baseline. To examine whether the difference is statistically significant, we use Mann-Whitney test (α=.05). Mann-Whitney is a non-parametric test. It does not make any assumption about the distribution of the data [25]. The test results show that, in all three datasets, our clustering-based approach performs significantly better than the baseline pruning method, with <recall, precision> *p-values* of $<0.043, 0.018>$, $<0.036, 0.046>$, and $<0.018, 0.034>$ for iTrust, eTour, and CM-1 respectively. This leads us to conclude that our approach significantly enhances candidate link generation for automated requirements traceability.

Fig. 9 shows the MAP values when the two presentation styles, ABS and MAX, are compared. The results imply that ABS slightly outperforms MAX in all three datasets; however, the improvement is significant only in eTour (p=0.046). This finding raises some interesting issues in how to best present the candidate links to the human analyst. While our results seem to suggest that the traditional ranked-list display (ABS) contains more correct links on the top, preserving cluster boundaries (MAX) has shown to be valuable in improving the understandability and usability of candidate traceability links [10]. We are currently carrying out pilot studies with our TraCter tool [21] to further investigate the presentation factor and its impact on assisted requirements tracing [26].

The MAP analysis completes our experimental inquiry of seeking answers to the set of questions posed in Section III-B. The results can be summarized as follows.

- The cluster hypothesis holds in traceability.
- Single-link (SL), at the k=8 clustering granularity, represents a good candidate mechanism for fulfilling the potential suggested by the cluster hypothesis.
- The quality of clusters can be adequately inferred by their maximum similarity (MAX) to the trace query, and the 3 lowest-quality clusters contain such a high density of false positives that discarding them significantly improves the overall quality of the candidate link generation.
- Displaying the candidate links in an absolute similarity (ABS) manner facilitates the browsability aspect of the automated tracing results.

D. Threats to Validity

As is the case for most controlled experiments, our investigation into the cluster hypothesis in traceability is performed in a restricted and synthetic context. We discuss here some of the most important factors that must be considered when interpreting the results.

The *construct validity* [27] of our study can be affected by the use of a gold standard to operationally measure the "goodness" of clustering results. The gold standard that we devise assumes that an optimal decomposition of the traceability link space should have only two clusters, thus effectively splitting the correct links and incorrect ones into perfect-quality and zero-quality groups respectively. This assumption is based on an extreme leverage of the cluster hypothesis to the automated tracing problem. At the other extreme, every retrieved link can be its own cluster or only a couple of correct links can form a cluster which is separated from other links; however, such decompositions offer little value to automated tracing solutions. Our operationalization of "goodness" represents only one of the gauges to search for acceptable approximations to the clustering with constraints problem, but a practically useful one. In cases that other clustering solutions are desired (e.g., the cluster size should be adjusted to 7 ± 2 in order to take human usability into consideration [10]), we argue that the procedure in Fig. 3 may be applied incrementally to help find other optimal clustering settings.

One of the internal validity [27] threats relates to the sequential examination of the three enhancement steps in our approach (cf. Fig. 1). The test of the cluster-quality determination heuristics, for example, is executed by applying the optimal clustering mechanisms discovered in the previous step. While we feel that our execution configurations are logical, caution must be taken in interpreting our findings as a whole rather than separately.

The results of our experiment may not generalize to other datasets — a threat to the *external validity* [27]. Two of the projects are developed by students and are not necessarily

representative of all systems. In particular, proprietary software products are likely to exhibit different characteristics. We also note that our chosen traceability datasets are of medium size, which may raise some scalability concerns. Nevertheless, we believe the use of three datasets from different domains while incorporating both requirements-to-source-code and requirements-to-design traces helps mitigate related threats. We present an industrial case study in the next section in order to triangulate our findings. Other threats might stem from specific design decisions, such as the preprocessing indexer used, the tf-idf similarity measure computed, and so on.

Finally, in terms of *reliability* [27], we conduct all the experiments on open-source projects using procedures, algorithms, and measures completely described in this paper. Moreover, our implementations are available upon request. We therefore believe it is possible to independently replicate our results.

V. An Industrial Study

The experimental study presented in Section IV uses open-source projects to answer our research questions (cf. Section III-B) in a quantitative and precise way. To further investigate the benefits of our approach, we conduct an exploratory case study [27] by applying the results to an industrial proprietary software system deployed in the workforce development domain. The objective is to assess the usefulness and the scope of applicability of our approach, and more importantly, to identify areas for improvement.

A. Background

The subject system of our study is a software-intensive platform that provides technological solutions for service delivery and workforce development in a specific region of the United States. In order to honor confidentiality agreements, we use the pseudonym "WDS" to refer to the system. WDS has been deployed for almost a decade and is developed using Java technologies. It intends to meet the goals of multiple stakeholders, including job seekers, employers, educational institutions, and government agencies. The development team of WDS describes its requirements practice as a goal-oriented and agile process. While the high-level business goals shall be fulfilled, the changes to low-level design and implementation are also embraced.

Currently, WDS employs a commercial state-of-the-practice issue tracking system to manage the traceability information. Although there is no immediate need to invest in new techniques, the WDS team is very interested in exploring how automated tracing methods like ours can help improve their practice, especially for handling volatile requirements. For this reason, the team shared with us 6 business requirements and WDS's Java code base containing over 500 classes. The correct requirements-to-source-code links were identified by WDS developers.

Table II
Results of applying our approach to business requirements

Requirement	|Correct Links|	Recall			Precision		
		Retrieved	Baseline	Enhancement	Retrieved	Baseline	Enhancement
Req_1	25	.84	.68	.84	.06	.08	.10
Req_2	330	.92	.80	.85	.77	.84	.89
Req_3	14	.93	.50	.93	.08	.07	.13
Req_4	22	1.00	.73	.95	.06	.08	.11
Req_5	17	.94	.82	.82	.11	.20	.20
Req_6	20	.95	.80	.95	.08	.10	.12

B. Results

Table II shows the results obtained by applying our approach to the 6 WDS requirements. For each requirement, the table compares the performance of three tracing methods: term-based retrieval, baseline pruning, and our clustering-based enhancement. Note that these 6 requirements are processed by using the same procedures, algorithms, and configurations as described in Section IV. It is encouraging to realize that our approach greatly outperforms the baseline in generating the candidate traceability links for three requirements: Req_3, Req_4, and Req_6. Not only is recall maintained at a high level ($\geq$.93) by using our approach, but precision is markedly increased. This provides further evidence confirming the validity of the cluster hypothesis in traceability. Meanwhile, the result increases our confidence in the generalizability of our optimal clustering findings.

For Req_1, even the initial IR-based tracing results in a relatively low recall (.84). Although the clustering-based enhancement completely matches the recall level, this situation does exploit one of the limitations of our approach. Being query-specific, our approach does not provide much support on the recall side. Requirements like Req_1 contributes to the "hard-to-retrieve traces" problem [28], which needs solutions beyond the basic term-matching mechanism. For example, replacing the original query with a new set of query terms is proposed in [28]. Even though this shows that clustering on the query side can be useful, the scope of applicability of our approach is currently limited on the retrieval and filtering side.

Among the WDS requirements under study, Req_2 has the greatest number of correct traceability links, meaning that its traces are spread all over the link space. It is therefore difficult for any pruning or filtering mechanisms not to throw out some of the correct links. However, this does illuminate the value of our choosing a 2-cluster extreme (cf. Fig. 2a) as a gold standard. In practice, when Req_2 is traced, a successful dividing of the traceability links into perfect-quality and zero-quality clusters can be of great help. A closer look at Req_2 reveals that it is a non-functional requirement (NFR) describing security-related concerns which are addressed by a large number of artifacts in WDS's code base. Tracing NFRs has received growing research attention in recent years [29]. It is therefore interesting to exploit the special nature of NFRs to further improve our approach.

As far as Req_5 is concerned, clustering has no effect on candidate link generation. This is clearly an exception to the general trend in Table II, which shows that our approach improves both recall and precision over the baseline. It turns out that the two Java classes that are incorrectly filtered out implement general utility functions — DateUtil.java and EmailUtil.java. In the software clustering literature, these are known as omnipresent objects [30] and need to be handled separately from "regular" data objects. Thus, a potential improvement to our approach is to take into account other types of information, such as structural [30] and semantic [31] information, to produce more reliable and robust clusterings.

VI. Conclusions

In this paper, we have proposed an approach to improving the performance of IR-based automated tracing by examining the cluster hypothesis. The approach is presented through a set of detailed procedures. Three open-source datasets from different application domains are investigated to discover optimal settings for these procedures. We further evaluate our approach through an industrial study that helps identify the limitations of our approach and the avenues for future research. The study results show that our approach outperforms the baseline, but still has more room for improvement.

It is imperative to mention that as long as the challenge of achieving a 100% recall and precision is still standing, the problem of automated tracing remains unsolved. More research is required on different aspects of the problem to achieve the desired quality levels for automated tracing tools to be successfully deployed in industrial settings.

Research in software traceability has its root in Gotel and Finkelstein's seminal work [1]. In a recent paper [32], Gotel and Morris highlight the challenges specific to requirements traceability by revisiting a wide range of disciplines, from metrology to epidemiology, in which traceability has played a vital role over millennium. The essence is to illustrate how existing practices could be leveraged. Our work reported here has revisited, and tried to leverage, an underlying tenet in Computer Science, namely the cluster hypothesis, to improve automated requirements traceability. It is hoped that our work contributes a step towards realizing the vision of "a newer tracing discipline" [32].

Acknowledgement

We would like to thank the partner company for the generous support of our research. We also thank Gary Bradshaw and Tanmay Bhowmik for careful comments on earlier drafts of this paper. The work is in part supported by the U.S. NSF (National Science Foundation) Grant CCF-1238336.

References

[1] O. Gotel and A. Finkelstein, "An analysis of the requirements traceability problem," in *ICRE*, 1994, pp. 94–101.

[2] J. H. Hayes, A. Dekhtyar, and S. K. Sundaram, "Advancing candidate link generation for requirements tracing: the study of methods," *IEEE TSE*, vol. 32(1), pp. 4–19, 2006.

[3] J. Cleland-Huang, B. Berenbach, S. Clark, R. Settimi, and E. Romanova, "Best practices for automated traceability," *IEEE Computer*, vol. 40(6), pp. 27–35, 2007.

[4] J. Cleland-Huang, R. Settimi, C. Duan, and X. Zou, "Utilizing supporting evidence to improve dynamic requirements traceability," in *RE*, 2005, pp. 135–144.

[5] A. Marcus and J. I. Maletic, "Recovering documentation-to-source-code traceability links using latent semantic indexing," in *ICSE*, 2003, pp. 125–137.

[6] R. Oliveto, M. Gethers, D. Poshyvanyk, and A. De Lucia, "On the equivalence of information retrieval methods for automated traceability link recovery," in *ICPC*, 2010, pp. 68–71.

[7] J. H. Hayes and A. Dekhtyar, "A framework for comparing requirements tracing experiments," *IJSEKE*, vol. 15(5), pp. 751–782, 2005.

[8] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[9] A. De Lucia, R. Oliveto, and G. Tortora, "IR-based traceability recovery processes: an empirical comparison of "one-shot" and incremental processes," in *ASE*, 2008, pp. 39–48.

[10] C. Duan and J. Cleland-Huang, "Clustering support for automated tracing," in *ASE*, 2007, pp. 244–253.

[11] X. Chen and J. Grundy, "Improving automated documentation to code traceability by combining retrieval techniques," in *ASE*, 2011, pp. 223–232.

[12] X. Wang, G. Lai, and C. Liu, "Recovering relationships between documentation and source code based on the characteristics of software engineering," *Electronic Notes in Theoretical Computer Science*, vol. 243, pp. 121–137, 2009.

[13] L. Kaufman and P. J. Rousseeuw, *Finding Groups in Data: An Introduction to Cluster Analysis*. John Wiley & Sons Ltd., 1990.

[14] P. Willett, "Recent trends in hierarchic document clustering: a critical review," *Info. Processing & Mgmt.*, vol. 24(5), pp. 577–597, 1988.

[15] M. A. Hearst and J. O. Pedersen, "Reexamining the cluster hypothesis: Scatter/Gather on retrieval results," in *SIGIR*, 1996, pp. 76–84.

[16] J. Yi and F. Maghoul, "Query clustering using click-through graph," in *WWW*, 2009, pp. 1055–1056.

[17] O. Zamir and O. Etzioni, "Grouper: a dynamic clustering interface to Web search results," in *WWW*, 1999, pp. 1361–1374.

[18] A. Leuski, "Evaluating document clustering for interactive information retrieval," in *CIKM*, 2001, pp. 223–232.

[19] A. Mahmoud and N. Niu, "Source code indexing for automated tracing," in *TEFSE*, 2011, pp. 3–9.

[20] E. L. Lawler, J. K. Lenstra, A. H. G. R. Kan, and D. B. Shmoys, *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization*. John Wiley & Sons Ltd., 1985.

[21] A. Mahmoud and N. Niu, "TraCter: a tool for candidate traceability link clustering," in *RE*, 2011, pp. 335–336.

[22] Z. Wen and V. Tzerpos, "An optimal algorithm for MoJo distance," in *IWPC*, 2003, pp. 227–235.

[23] I. Davidson and S. S. Ravi, "Intractability and clustering with constraints," in *ICML*, 2007, pp. 201–208.

[24] H. Sultanov, J. H. Hayes, and W.-K. Kong, "Application of swarm techniques to requirements tracing," *REJ*, vol. 16(3), pp. 209–226, 2011.

[25] W. J. Conover, *Practical Nonparametric Statistics*. Wiley, 1999.

[26] A. Dekhtyar, O. Dekhtyar, J. Holden, J. H. Hayes, D. Cuddeback, and W.-K. Kong, "On human analyst performance in assisted requirements tracing: statistical analysis," in *RE*, 2011, pp. 111–120.

[27] R. K. Yin, *Case Study Research: Design and Methods*. Sage Publications, 2003.

[28] M. Gibiec, A. Czauderna, and J. Cleland-Huang, "Towards mining replacement queries for hard-to-retrieve traces," in *ASE*, 2010, pp. 245–254.

[29] J. Cleland-Huang, "Toward improved traceability of non-functional requirements," in *TEFSE*, 2005, pp. 14–19.

[30] Z. Wen and V. Tzerpos, "Software clustering based on omnipresent object detection," in *IWPC*, 2005, pp. 269–278.

[31] A. Mahmoud, N. Niu, and S. Xu, "A semantic relatedness approach for traceability link recovery," in *ICPC*, 2012, pp. 183–192.

[32] O. Gotel and S. J. Morris, "Out of the labyrinth: leveraging other disciplines for requirements traceability," in *RE*, 2011, pp. 121–130.