Text Mining Support for Software Requirements: Traceability Assurance

Abstract

Requirements assurance aims to increase confidence in the quality of requirements through independent audit and review. One important and effort intensive activity is assurance of the traceability matrix (TM). In this, determining the correctness and completeness of the many-to-many relationships between functional and non-functional requirements (NFRs) is a particularly tedious and error prone activity for assurance personnel to peform manually. We introduce a practical to use method that applies well-established text-mining and statistical methods to reduce this effort and increase TM assurance. The method is novel in that it utilizes both requirements similarity (likelihood that requirements trace to each other) and dissimilarity (or anti-trace, likelihood that requirements do not trace to each other) to generate investigation sets that significantly reduce the complexity of the traceability assurance task and help personnel focus on likely problem areas. The method automatically adjusts to the quality of the requirements specification and TM. Requirements assurance experiences from the SQA group at NASA's Jet Propulsion Laboratory provide motivation for the need and practicality of the method. Results of using the method are verifiably promising based on an extensive evaluation of the NFR data set from the publicly accessible PROMISE repository.

Keywords- traceability; software assurance; completeness; non-functional requirements

1. INTRODUCTION

Though it is well known that traceability information can assist with a number of vital software engineering and software assurance activities, the capture and maintenance of such information is still not commonplace. This is largely due to the fact that building requirements traceability matrices (RTMs), even with automated support, is very time consuming and error prone. RTMs are thus not maintained and hence cannot be used to support activities such as change impact analysis, regression testing, criticality analysis, etc. Researchers have sought to improve automated techniques for generating and maintaining RTMs. However, there has been very little done to address the assessment, either manually or with automated support, of RTMs that are built. In fact, a recent survey of traceability in the requirements engineering community as well as in the model-driven development community [1] notably lacks any discussion of the assessment of RTMs. As a point of terminology, when a party not directly involved in generating the requirements performs assessment, it is generally called *assurance.*

The Software Quality Assurance (SQA) group at NASA's Jet Propulsion Laboratory (JPL) is one of many organizations developing or assuring large mission- and/or safety-critical systems. For them, the development of RTMs is a mandated activity. Because of the high-risk typically associated with JPL projects, assurance of RTMs is also a mandated activity, albeit one that is not being accomplished as effectively or efficiently as desired. Interviews with SQA personnel indicate that "Spot checking" and "completeness by expectation" are phrases that best describe the heuristics currently applied when assessing traces.

As if these tracing challenges weren't daunting enough, a further challenge confronts assurance personnel – that of non-functional requirements or NFRs. Assurance staff are responsible for ensuring that all NFRs trace to all appropriate functional requirements (or FRs), and that there are no inappropriate or spurious traces (i.e., requirements that with certainty do not trace, or *anti-traces*). This is also referred to informally as ensuring the *completeness* and *correctness* of the NFR traces. What makes this difficult is that the degree or strength of a trace is not considered; it is a binary relationship, each NFR either traces or anti-traces to an FR. What exactly determines an appropriate degree is generally unspecified, but the intent is to only trace an NFR to an FR upon which it has an observable effect. We refer to the RTM sub-matrix of traces from NFRs to FRs simply as the NFR-FR matrix.

Because there may be missing traces, this is a highly effort-intensive activity as assurance personnel must check all possible traces, not just those already indicated in a traceability matrix. In addition, traceability assurance is a highly detail-oriented and information-intensive activity, unguided and tedious for humans to perform. Clearly automated traceability techniques could greatly assist in guiding assurance efforts. Unfortunately, approaches do not currently exist to assist with such completeness or correctness assessments.

We introduce a pragmatic text-mining based approach currently under study within JPL's SQA group to assessing the completeness and correctness of NFR-FR traces that reduces the effort required of assurance personnel while increasing quality and confidence. Specifically, we introduce a method akin to graph-based semi-supervised classification [2] utilizing a mixed model of requirements similarity and dissimilarity to automatically generate investigation sets that help identify low-risk and possible problem areas in the NFR-FR RTM. The approach also supports automatic adjustment to the quality of the trace under assessment. The approach was empirically validated through a series of controlled experiments and a comprehensive case study using data from the PROMISE repository [9].

Because the method is being developed for use within a practicing assurance organization, our approach is more pragmatic than theoretical. In this, success is determined by

how well the method meets certain practical key objectives for application. These will be detailed later. Pragmatics dictates that we try to make use of established techniques and technology. Notwithstanding, there are a number of novelties in our approach: a) It uses information contained in a given RTM to address difficult assurance problems; b) It leverages information generated independently from both similarity **and dissimilarity** between requirements; c) It integrates deterministic classification rules with statistically generated classifications; and d) It considers both traces and anti-traces as first class artifacts.

The paper is organized as follows. Section 2 discusses requirements assurance. Manual performance of NFR-FR tracing is discussed in Section 3. Related work is presented in Section 4. Section 5 presents text-mining support for trace assurance while Section 6 discusses the investigation set generator. A case study is presented in Section 7. Evaluation follows in 8. Section 9 presents validation of the method and Section 10 concludes.

2. REQUIREMENTS ASSURANCE AND TRACING

What is requirements tracing assurance ?

Requirements assurance is concerned with independently (from requirements developers) assessing and ensuring the quality of the requirements. Chiefly this includes assessing the correctness and completeness of the requirements for which tracing is key. This tracing encompasses high and low level requirements as well as their relationships *with* other artifacts (relationships between requirements and test cases, e.g.) and *within* artifacts (tracing between NFRs and FRs). Tracing assurance is typically performed by assurance professionals within an SQA engagement, while conducting verification and validation (V&V) of requirements, or within an independent V&V (IV&V) assessment.

Why assure traces?

In 1999, the Mars Climate Orbiter (MCO) was lost due to the incorrect use of English measures by the development contractor (not in keeping with the interface specifications which called for metric units). In addition to the loss of the $125 million Orbiter, further issues were caused for its partner mission Mars Polar Lander which was to receive data from the climate orbiter. While official reports discussed breakdowns in the V&V of the system (including lack of end to end testing for the trajectory tracking software), a careful NFR-FR trace assurance may have avoided the problem.

Software assurance is a risk management technique. As with defects, it is well known that the earlier that risks can be identified and mitigated, the better. Specifically, it is important to ensure that all FRs have considered the appropriate NFR qualities and to ensure that adequate tests are planned for execution and quality specifications are satisfied. Toward that end, JPL SQA primarily performs requirements tracing assurance early in the lifecycle, generally during requirement definition or before appropriate milestone reviews. They have found that this avoids more costly rework later in the lifecycle (at/after testing).

The NFR-FR completeness of tracing problem

In NFR-FR traceability assurance, correctness is important but completeness is a more significant focus as it generally presents more risk. For example, a missing NFR-FR trace may lead to a later omission of a test cover trace, and subsequently to omission of a test for the quality of a critical function (e.g., metric units for output of propulsion system monitor). Incorrect NFR-FR tracings are more likely to be identified and resolved at various stages (e.g., reviews, test) whereas a missing trace can more easily go undetected and thus become a latent defect (i.e., lies dormant in the system or is detected only when it presents a failure in operation). Traceability research has also shown that humans are better at discovering errors of commission in traces than at discovering errors of omission [11]. In addition, NFRs present a particularly challenging problem in that they tend to be broadly relevant to many FRs. We discuss the challenge of NFR-FR trace assurance next.

3. CHALLENGE OF MANUAL NFR-FR TRACE ASSURANCE

There are three major challenges to the manual assurance of NFR to FR traces: size, complexity, and effort/cost. We discuss each below.

Size

Imagine a very small software requirements specification consisting of just 50 requirements, 20 of which are FRs and 30 of which are NFRs. There are 20 x 30 = 600 possible traces between the FRs and NFRs that may have to be assured. It is not optimal for analysts to assess 600 traces manually, but it is possible. In contrast, the MCO (mentioned earlier) had over 7500 requirements, a portion of which were NFRs. If we assume 7300 FRs and 200 NFRs, there could still be 1,460,000 traces to assure.

Complexity

NFR to FR trace assurance is a matching problem, central to graph theory, which can be modeled as a bi-partite graph (the tracing graph) on the two sets of requirements NFR and FR where an edge indicates that a given non-functional requirement affects the related functional requirement. Such bi-partite graphs are equivalent informationally to an NFR-FR matrix. The matching problem is relevant due to the fact that every NFR must trace to at least one FR and that the focus is on validating a "tracing" from the many "valid" combinations of tracings possible (that is, not all valid traces of an NFR-FR are expected to be relevant or of interest).

Even though the tracing graph is expected to be relatively sparse (generally each NFR traces only to a small percentage of FRs), assuring completeness requires examination to ensure that edges are valid and no edges are missing. This implies that the complete bi-partite graph K(NF,F) with

|NF|*|F| edges must be reviewed to verify the trace/anti-trace relevancy. This requires O(|F|^2) number of steps [1].

Part of the assurance process is determining the risky and non-risky areas, thus we cannot reduce the complexity by prioritizing or reducing the set of requirements to investigate (the "investigation set") based solely on external risk or cost.

We have taken a somewhat simplified view of the problem. At JPL, requirements tend to be hierarchical: there is "flow-down" from one level to another (hence the terms upwards/downwards tracing). Thus if there is a trace at one level, this trace will flow-down to the requirements below it. This can significantly reduce the number of traces to be verified. However, in order to "depend" on this hierarchy, one must first validate that the trace is at the appropriate level. Hence, "layering" the requirements does not totally circumnavigate the complexity of the assurance problem.

Effort/cost

NFR to FR tracing assurance is a costly and effort consuming activity. Consider the MCO project with 7500 requirements. If we assume that there are 1,460,000 traces to assure and assume an average of 1 minute per trace audit (a highly optimistic estimate), then we expect |7300|*|200|/60 = 24,333 person-hours of effort. With a 40-hr work week, it would take 11.7 people an entire year to complete the work. As a result, assurance personnel rarely perform exhaustive analysis. Rather, they become "familiar" with the requirements and use a variety of approaches to approximate a completeness check. A common approach is to "spot check" to rapidly identify potential problem areas and then to focus on these. Another popular approach is to only validate the existing traces and then prune and expand these. Assurance personnel will augment these approaches by considering related groups of requirements. For example, if there is a trace from a particular NFR to a FR, then it is often fruitful to look at the requirements that are similar or strongly related to that FR for traces.

All these approaches assume a sufficient familiarity with the entire set of requirements and rely heavily on the experience and domain knowledge of assurance personnel. Given this assumption, completeness is addressed by comparing a given trace to what traces are "expected" relative "not expected" in the particular system. Gaps in domain knowledge are unavoidable (a person cannot keep all knowledge of a system in their mind at one time), thus making it difficult to gauge the believability of a completeness audit.

It is clear that the above challenges point to the need for automated tool support. However, such support must also be in alignment with assurance practices as indicated above.

4. RELATED WORK

Related work is addressed in the subsections below.

[1] assuming |F| > |NF|.

Challenges in Requirements Traceability Research and Practice

There has been relatively little work on the assessment or quality assurance of traces. The only work on assessing traces is that of Dekhtyar, et al. [5] where a committee of automated methods was executed and each voted on the accuracy of a given RTM link. A number of different voting schemes were used. The approach succeeded at finding and rejecting false positives (pf) in RTMs created by automated methods. Our paper meets these traceability challenges by reducing the human effort required for and by increasing the confidence in assurance activities performed for the RTM.

In addition to the shortcoming of research on assessing traceability matrices, there is also a lack of work on non-functional requirements (NFRs). Next, we address research to date on NFR traceability.

Tracing Non-functional to Functional Requirements

Our work examines the satisfaction or completeness of NFRs by FRs (each NFR minimally needs to map to at least one FR in order to be 'satisfied'). Holbrook, Hayes, and Dekhtyar examined the use of RTMs to assist with performing satisfaction assessment determining if requirements were satisfied by design, e.g. [6]. Such a technique could be used to examine each NFR and see if it is satisfied by one or more FRs. It should be noted that this technique requires that each FR/NFR be chunked (parsed into phrases) as well as tagged with parts of speech. Our technique does not require this pre-processing. Another unique aspect of our work is the use of bi-partite graphs. Though all tracing work directly or indirectly represents RTMs as graphs, traceability research does not discuss the assessment of the RTMs based on this structure. Next, we examine the use of clustering to support tracing.

Clustering Support for Automated Requirement Tracing

Cleland-Huang, Settimi, Zou, and Solc examined a technique for automating the detection and classification of NFRs based on stakeholders' quality concerns across requirements specifications containing scattered and non-categorized requirements, and also across freeform documents [7]. In fact, our validation uses their datasets and classifications. Compared to the Huang work, a unique contribution of our clustering support is to make use of both requirements similarity and dissimilarity to generate two sets of clusters tracing NFRs to FRs.

Goldberg, Zhu, and Wright present a semi-supervised classification algorithm that learns from dissimilarity and similarity information on labeled and unlabeled data to handle both binary and multi- class classification [2]. This work provides a theoretical support for our clustering methods.

5. TEXT-MINING SUPPORT FOR TRACE ASSURANCE

We now describe the approach for using text mining to support trace assurance. For clarity, we emphasize a few things up front. First, the method does not aim to generate an

RTM. Indeed, the method requires an existing RTM as input i.e., the RTM to be assured. Second, the aim of the method is not to automate the detection of or assure FRs or NFRs. However, a by-product of tracing assurance can help with this. Last, the method is not designed to detect vague or poorly stated requirements.

With the above in mind, we state that a successful method for automated support of trace assurance at JPL would meet the following vital objectives: 1) Must be compatible with the way assurance personnel address trace assurance (e.g., "expected" and "unexpected" traces based on prior experience, domain knowledge, and familiarity with the requirements); 2) Must be empirically driven, adjusting to the quality of the requirements specification (e.g., vaguely specified requirements should result in more conservative automated results) and adjusting to the quality of a given RTM; 3) Must be easily implemented and integrate with existing requirements managers (e.g., DOORS, RequisitePro, etc.); 4) Must have an established theoretical foundation; Must be practical to use (e.g., low-learning curve) and provide meaningful guidance; 5) Must be based on open methods and technologies (assurance cannot be based on black-box solutions); 6) Must reduce overall effort, increase efficiency of effort, and increase confidence in results. Note that Hayes, Dekhtyar, Sundaram, and Howard have posited essential requirements for any requirements tracing tool as examined from the user's perspective. Objective (5) ties to their Usability sub requirement (of Believability) [7].

We now describe an approach to meet the above objectives in a series of concepts and examples given below.

Trace investigation sets

The fundamental challenge for trace assurance is effectively managing the verification of a large number of traces and anti-traces. A natural means of addressing this is to employ a divide and conquer strategy that partitions these sets into more manageable *investigation subsets* based on meaningful rules and empirical properties of the requirements. For example, an obvious rule is "each NFR must trace to at least one FR" and the resulting investigation set (a subset of the traceability graph that is under assessment) would simply be all those NFRs without traces. Determining rules and properties and making them actionable (e.g., if an NFR has no traces then it must be removed or be reported as having missing traces) helps address objective (5).

Aligning rules and empirical properties with assurance personnel's a priori knowledge helps meet objective (1). Partitioning the assurance tasks into investigation sets greatly reduces the complexity and narrows the focus of the assurance effort. Furthermore, each investigation set implies particular assurance activities (e.g., look for a missing trace), "guiding" the effort to be more efficient and effective thereby helping to meet objective (7). If some investigation sets have a low risk (when appropriately defined) of its elements being incorrectly determined (as being in the set, for example), then such sets can be eliminated or "lightly" assured further helping to satisfy objective (7).

To illustrate, assume that as we examine the traceability matrix to be assured, we notice an observable property between a pair of requirements called "high-similarity" (perhaps each requirement contains many of the same words, e.g.) which we believe is highly correlated (but this is not certain) with requirements that have been associated to each other in the traceability matrix (meaning that it is highly correlated with our notion of ***trace***). We note that absence of this property between two requirements does not imply that they ***anti-trace***. The absence of "high-similarity" provides no information, whereas the presence of "high-similarity" appears to provide evidence of ***trace***.

With this idea in mind, let us examine the notion of trace investigation sets further. In the previous example, we discussed the ***trace*** set or T. Fig. 1 shows T in the top left; all NFRs trace to at least one FR, but do not trace to every FR. By simply examining the edges that do not exist in T, we obtain the ***anti-trace*** or AT (shown in the top middle section of Fig. 1). Based on T and AT and our notion of "high similarity" (called HT), we can generate four trace investigation sets L, M, F, N (see Fig. 1).

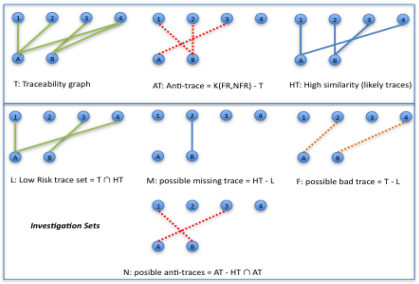

Figure 1. Illustration of partitioning into investigation sets.

We consider traces in the investigation set L (T∩HT, as shown in Fig. 1) to be *low risk* as they have two independent sources corroborating the trace (they were in the RTM under assessment and we observed the high-similarity property). Items in M are at high-risk of being *possible omissions* from T as we expect requirement pairs with high-similarity to trace (but not the converse). Items in M need to be carefully checked to see if they are indeed traces. We have little information about items in F, but as they did not have high-similarity, they should be checked first as possible bad traces (also called false positives or errors of c*ommission*). Last, there is little to say about items in N other than that they do not have high-similarity and they did not trace, so we first try to verify that they are anti-traces.

The example just presented, while simplified and overly generic, is in essence our method. The complexity reduction, work avoided (assuming we do not check the low-risk set L), and increased assurance efficiency is self-evident. Increased

confidence in the assurance results is in part self-evident, but also depends greatly on our confidence in the correlation of the high-similarity property and requirements that trace.

Finding properties that are practical to observe and in which experienced assurance personnel have high confidence is a key component of our method. Also, finding properties that determine both *inclusion* and *exclusion* of elements into an investigation set is essential to effectively addressing the trace completeness problem; this is discussed next.

Similarity and Dissimilarity of requirement pairs

Several studies have suggested that requirements that trace have a high degree of "similarity" based on the terms they use [11] and in the semantic meaning or context in which they are used [12]. This is intuitive as a trace indicates a relationship between a pair of requirements that are typically expressed by using common or highly related terminology (e.g., NFR: "The Xs shall have Y," FR: "This X shall do Z", the trace is the implied relation on all Xs). Similarly, but with subtle differences, requirements that anti-trace have a high degree of dissimilarity. This is intuitive as we do not expect to see similar terms used in a similar context for requirements with little or no relationship. Rather, we expect them to have a high degree of independence.

Our observations of trace assurance in practice and interviews with JPL SQA personnel indicate that, indeed, similarity and dissimilarity form the fundamental basis for verifying traces and anti-traces. Hence, similarity and dissimilarity are good candidates for empirical properties to generate investigation sets that satisfy our objective (1). However, "high degree of similarity/dissimilarity" are generally subjective assessments which are somewhat ambiguous, arbitrary, or inconsistent, and perhaps difficult to observe in an automated manner. It is unclear what confidence we can have in such assessments, especially when results vary between assessors and between assessments performed by the same assessor; this runs contrary to our objectives (3), (4), and, in part, (7).

Fortunately, text-mining research has produced a number of well-established approaches for reliably automating the assessment of similarity/dissimilarity between documents (requirements, in our case). One such approach is Latent Semantic Analysis (LSA) [10]. Briefly, LSA is a theory and method for extracting and representing the contextual-usage meaning of words by statistical computations applied to a large corpus of text. The underlying idea is that the totality of information about all the word contexts in which a given word does and does not appear provides a set of mutual constraints that largely determines the similarity of the meaning of words and sets of words to each other.

We know that LSA produces measures of word-word, word-passage, and passage-passage relations that are reasonably well correlated with several human cognitive phenomena involving association or semantic similarity. LSA allows us to approximate human judgments of overall meaning similarity, estimates of which often figure prominently in research on discourse processing. It is important to note from the start, however, that the similarity estimates derived by LSA are not simple contiguity frequencies or co-occurrence contingencies. Rather, the estimates depend on a deeper statistical analysis (thus the term "Latent Semantic") that is capable of correctly inferring relations beyond first order co-occurrence. As a consequence, LSA is often a much better predictor of human meaning-based judgments and performance than contiguity and co-occurrence counts.

Approximating human judgment of dissimilarity is a related, but fundamentally different, problem than similarity. Research has shown that we cannot define dissimilarity as "not similar" because natural language interpretation does not follow the law of the excluded middle. Thus, similarity and dissimilarity between requirements provide different information; a fact that our method exploits. Fortunately, LSA is also capable of characterizing dissimilarity.

Correlation and likelihood of trace/anti-trace

Given a pair of requirements (NFR_i, FR_j), the LSA characterization for similarity results in a similarity measure $S(NFR_i, FR_j) \in [-1,1]$ and for a dissimilarity measure $D(NFR_i, FR_j) \in [0,1]$, where higher values indicate greater similarity/dissimilarity. Typically, similarity is expressed as a symmetric matrix $\mathbf{S}[i,j]=S(NFR_i, FR_j)$ with $\mathbf{D}[i,j]=D(NFR_i, FR_j)$ expressing dissimilarity. We have performed a number of studies using simulated data and real project data (from the NFR PROMISE data set), which consistently support the desired correlations of LSA similarity/dissimilarity with trace/anti-trace: 1) likelihood of trace increases with LSA similarity (**S**), and 2) likelihood of anti-trace increases with LSA dissimilarity (**D**). Curiously, low LSA similarity does not increase likelihood of anti-trace, and the same for low dissimilarity and trace.

There are a variety of ways to interpret the above statements. For example, in one study we fit a probit model:

$$P[(NFR_i, FR_j) \text{ is a trace} \mid S(NFR_i, FR_j)] = \Phi[\beta\, S(NFR_i, FR_j)+\alpha].$$

to both simulated and real project data (NFR P10). We found that the parameter β was statistically significant. We do not detail these studies further here. Rather, note that while we have high confidence in the correlation of the LSA measures in general, in practice such correlations must be verified for each particular case.

Having performed the more general correlation studies, such as the probit model, for a particular RTM, it suffices to perform a simple hypothesis test to indicate if the estimator (e.g., the median or mean) for the set of observations in **S** (or **D**) for a set of requirement pairs that trace (or anti-trace) is significantly different than the estimator for the set of pairs that do not. Our general studies show that if there is significance, the correlation will be expected. We will show examples of this in the case study given in section 7.

However, correlation does not address the problem of sufficient *degree* of similarity or dissimilarity. As discussed earlier, assurance personnel determine subjectively what degree of similarity/dissimilarity indicates a likely trace/anti-trace. This is highly variable and is often performed on a case-by-case basis. In particular, there may not be a single constant "threshold" by which requirement pairs with greater similarity are considered traces (noting again that low similarity does not necessarily imply an anti-trace). One approach to this is to interpret the degree of similarity as an observed conditional on the likelihood function of the *indicator* that Trace(NFR_i, FR_j) = 1 (i.e., if the requirements trace, 0 anti-trace). In this context, a natural model is to seek a threshold value S_t such that when $S(NFR_i, FR_j)] > S_t$, it is "highly-likely" that Trace(NFR_i, FR_j) = 1. We represent this by the concept of a *likelihood function* that we denote by $L[Trace(NFR_i, FR_j)=1 \mid S(NFR_i, FR_j)] > S_t]$ quantifying how likely the event Trace(NFR_i, FR_j)=1 would occur given the observation $S(NFR_i, FR_j)] > S_t$. Likelihoods are similar to probabilities but with a different perspective. We are interested in finding the value of the parameter S_t that makes Trace(NFR_i, FR_j)=1 "highly-likely" (or most probable) given the similarity data **S** and trace data Trace(NFR_i, FR_j). From likelihood theory (also from Bayes Rule), for constant α:

$$L[Trace(NFR_i, FR_j)=1 \mid S(NFR_i, FR_j)] > S_t] = \alpha\, P[S(NFR_i, FR_j)] > S_t \mid Trace(NFR_i, FR_j)=1].$$

The situation is completely analogous for the dissimilarity matrix **D** where we are interested in the threshold value D_t for the likelihood function $L[Trace(NFR_i, FR_j)=1 \mid D(NFR_i, FR_j)] > D_t]$. For simplicity, we will limit our subsequent discussion to similarity (dissimilarity is analogous).

A nice feature of the likelihood functions of interest is that they are determined from the particular requirements and the corresponding RTM under assessment (in our case, the sub-matrix of NFR ***traces*** FR). Thus, they "automatically adjust" to the given quality of the requirements and traceability matrix as desired by objective (2). Here we do not assume or expect universal or constant threshold values S_t and D_t. With such values at hand, we have a precise meaning for "highly-similar" and "highly-dissimilar" and can proceed to generate investigation sets as illustrated in Fig. 1. Determination of the threshold values S_t and D_t will be discussed next.

Empirical Maximum Likelihood estimates

Generating useful trace investigation sets is predicated on the ability to determine, from a set of requirements and NFR-FR trace matrix, meaningful values for the similarity trace threshold S_t and dissimilarity anti-trace threshold D_t (understanding likelihood functions described previously). Two challenges exist. First, we cannot assume particular probability distributions for the collections of data at hand – i.e., the $S(NFR_i, FR_j)$s or Trace(NFR_i, FR_j)s. They can be, from our studies, wholly arbitrary. Second, there are multiple significant sources of error in the data at hand: an unknown number of incorrect traces and anti-traces, inherent error and inaccuracy in the text-mining that determines the LSA similarity and dissimilarity values, and error in the correlation between traces and LSA similarity. Furthermore, we do not have insight into what the possible distributions of the above errors might be.

Owing to these errors, we must contend with "noise" in the data when determining the threshold values we seek. Our approach must be robust and make minimal assumptions about the data. With this in mind we will assume:

1. likelihood of trace increases with similarity,
2. likelihood of anti-trace increases with dissimilarity, and
3. NFR-FR traces/anti-traces are mutually independent.

However we do allow traces and anti-traces to depend on the similarities and dissimilarity between *all* requirements.

Consider the ratio of likelihoods:

$$LR = L[Trace(NFR_i, FR_j)=1 \mid S(NFR_i, FR_j)]>S_t] \,/\, L[Trace(NFR_i, FR_j)=0 \mid S(NFR_i, FR_j)] > S_t].$$

LR represents the relative likelihood that when its similarity exceeds the threshold S_t, the pair (NFR_i, FR_j) is actually a trace or "true positive" versus not being a trace or "false positive." The larger the value of LR, the greater confidence we have in the statement "it is highly likely that (NFR_i, FR_j) is a trace when $S(NFR_i, FR_j)]>S_t$." Thus, we want to find S_t such that LR is maximized. Using the definition of the likelihood functions we have:

$$LR = \Omega\, P[S(NFR_i, FR_j)] > S_t \mid Trace(NFR_i, FR_j)=1] \,/\, P[S(NFR_i, FR_j)] > S_t \mid Trace(NFR_i, FR_j)=0]$$

for some constant Ω. While we do not know the distributions for the probabilities above (nor the constant Ω), we will generally have enough data so that they can be approximated reasonably well by their marginal empirical cumulative distribution functions (edcf). Our studies indicate that the approximation errors involved make estimates of Ω unreliable. Fortunately, Ω only affects the value of LR at the maximum, not its location, so in determining the value S_t we can ignore this constant. What we sacrifice is our ability to estimate the level of confidence (or significance) we can have in determining a trace when its similarity exceeds S_t. As demonstrated in section 7, graphical optimization techniques are generally sufficient for estimating S_t. Numerical optimization methods must be supervised carefully due to the inherent "noise" in the LR function that may cause spurious "spikes" and lead to false maximums as exemplified in our case study.

Rules and their interpretations

The tables below list rules we have observed in trace assurance with respect to properties S_t, D_t, and Trace(). TABLE I. lists rules related to the structural of the overall NFR-FR matrix and do not create a partition. These structural rules detect inherently problematic trace patterns.

TABLE II. list rules used to partition the TM into investigation sets. Each set represents "likely" trace properties.

TABLE I. STRUCTURAL RULES

rule	Investigation Set
R1: Trace(NFR_i, FR_j)=0 all j	NFR_i must trace to at least one FR_j
R2: Trace(NFR_i, FR_j)=1 all j	Too broad
R3: D(NFR_i, FR_j)=1 all j	Irrelevant
R4: S(NFR_i, FR_j)=1 all j	Too broad
R5: S(NFR_i, FR_j) = S(NFR_k, FR_j) some k	Duplicates NFR_i, NFR_k
R6: S(NFR_i, FR_j) = S(NFR_i, FR_k) some k	Duplicates FR_i, NFR_k

TABLE II. NFR-FR partition rules

	$S>S_t$, $D>D_t$	$S>S_t$, $D<D_t$	$S<S_t$, $D>D_t$	$S<S_t$, $D<D_t$
Trace=0	**R7**: Over-specialized	**R9**: False anti-trace	**R11**: Low risk	**R13**: No info
Trace=1	**R8**: Over-general	**R10**: Low risk	**R12**: False trace	**R14**: No info

Integration with requirements management systems

Our prototype implementation of the method was performed entirely within the open-source statistics system R without any customization and utilizing the publicly available tm (text-mining) and proxy (Distance and Similarity Measures) packages. This was satisfactory for our evaluation and proof of concept study. For practice, we envision an assurance tool that integrates with JPL's requirement management system (IBM-Rational DOORS) that utilizes the R system API to perform statistical functionality. An assurance person would use the tool to acquire an RTM for a project, the tool would process the RTM generating the investigation sets and report these in an RTM annotated with codes (or colors) indicating possible concerns and low-risk elements. A confidence level and effort estimate report would also be generated.

6. GENERATING INVESTIGATION SETS

The process for generating the investigation sets in practice has six steps:

1. Acquire and prepare requirements and RTM data
2. Generate similarity and dissimilarity matrices
3. Verify similarity/dissimilarity correlations
4. Generate empirical likelihood ratio functions
5. Determine MLE S_t, D_t threshold values
6. Apply rule sets to generate investigation sets

Assurance proceeds by validating the individual elements in each investigation set according to their interpretations.

Step1: Acquire and prepare requirements and RTM data

There is a certain amount of pre-processing needed on the requirements text and RTM in order to perform the text-mining and statistical operations. First, the requirements must be converted into a "corpus" where each individual requirement is a "document." We assume that each requirement has been classified as NFR or FR and we label these documents NFR_1, NFR_2,.,NFR_m and FR_1, FR_2, …, FR_n. Next, we pre-process the requirements corpus by removing punctuation, extra whitespace, stop-words (e.g. "the", "of", "shall") and performing stemming (see [11] for details). This is necessary to reduce "noise" and nuisance factors when computing similarity and dissimilarity.

Requirements are a little different than general documents in that they tend to make frequent use of generic terms such as "system" and "project" that provide little information. Frequent terms that appear in many documents are automatically down-weighted by the LSA algorithm, however they still contribute noise. This is not sufficient because use of such terms can vary greatly and we want to avoid creating artificial similarities between requirements simply because both use the term "system" while another requirement did not use the term. So we do our best to identify and remove such words. This is relatively easily done by first generating a term-frequency list and looking at the most frequent terms that are judged generic. These words are then removed from all documents in the corpus. If there is doubt about a particular word being generic, we leave it in.

Step2: Generate similarity and dissimilarity matrices

Having pre-processed the requirements corpus we generate the similarity matrix **S** and dissimilarity matrix **D**. We use LSA with the "cosine" similarity and distance measures. There are a variety of alternative analyzers and measures, some of which we have experimented with, but none that were particularly superior. What is most important is that **D** is generated independently from **S** via an actual dissimilarity measure. That is, something like setting **D** = **I** – **S** (where **I** is the identity matrix) would *not* be independently generated. The matrix **S** should not be used at all in the generation of **D**. At this point we should now have available as entries from the matricies S, D, and RTM, the values for S(NFR_i, FR_j), D(NFR_i, FR_j), and Trace(NFR_i, FR_j) respectively.

Step 3: Verify similarity/dissimilarity correlations

Before proceeding, it is good practice to check that the errors present do not overwhelm the information we may extract from the data given. In particular, the fidelity of the data should be such that we are confident that the assumptions 1-3 given in Section 5 are satisfied. Here we discuss what is sufficient to test that the set of similarity values for traces and anti-traces (as determined in the NFR-FR matrix) have significantly different similarities. Because we cannot assume a particular distribution for the similarity measures between requirements, we suggest using boxplots and the Wilcoxon Rank Sum significance test. Specifically we compare boxplots of the set of values T = {S(NFR_i, FR_j) | Trace(NFR_i, FR_j)=1} with the set NT = {S(NFR_i, FR_j) | Trace(NFR_i, FR_j)=0} and verify that there is an observable difference. Then, we perform a two-sided Wilcox test with the null-hypothesis median(T) = median(NT) and verify that the p-value is less than 0.05 (or a selected confidence level).

The lower the p-value the more confident we can be there is sufficient information in the data because the median values of the two sets differ significantly. We repeat the above for the analogous sets of dissimilarity taken from **D**.

Assessing independence from data is non-trivial. We want to avoid high co-linearity between **S** and **D**, we expect some degree of negative correlation because ideally a pair of requirements should not be both highly similar and dissimilar ("highly" being defined as exceeding S_t and D_t). For this, we compute the Pearson correlation coefficient between the sets of values $\{S(NFR_i, FR_j)\}$ and $\{D(NFR_i, FR_j)\}$ and verify that the value is in the range [-0.8, -0.2]. We provide examples of using boxplots, Wilcox tests, and negative correlation in the case study and validation sections.

Step 4: Generate empirical likelihood ratio functions

In Section 5 the likelihood ratio LR is defined by conditional distributions like $P[S(NFR_i, FR_j)] > S_t \mid Trace(NFR_i, FR_j)=1] = 1 - P[S(NFR_i, FR_j)] \leq S_t \mid Trace(NFR_i, FR_j)=1]$. The right hand side distribution function $P[S(NFR_i, FR_j)] \leq S_t \mid Trace(NFR_i, FR_j)=1]$ can be approximated by creating the empirical cumulative distribution function from the set T defined in Step 3 above (this function is straightforward to compute and packages such as R have excellent support). We call this function T_edcf(x) and similarly create NT_edcf(x) from the set NT. Now the similarity LR will be approximately proportional to
slr(x) = (1-T_ecdf(x))/(1-NT_ecdf(x)). Then, the analogous empirical dissimilarity LR function is generated, dlr(x).

Step 5: Determine MLE S_t, D_t threshold values

We plot slr(x) and visually estimate a range for the location of the maximum value. We take care to ignore "spikes" and look for a true maximum. Spikes occur from noise and approximation gaps in the data. What we look for is a so-called "stable" maximum where a small shift to the left or right does not result in a large drop in value. Once we have an estimated range, we use a numerical optimization function to help narrow down this range. We may need to adjust the search range or tolerance to avoid spikes. With a range in hand, we set S_t to the maximum value in this range (if one wishes to be conservative, or to the minimum value if more tolerant of false positive trace detection). We repeat the above with the function dlr(x) to determine D_t.

Step 6: Apply rule sets to generate investigation sets

We now have all the values needed to apply the rules listed in Tables II,III. For each rule R_i, we create an investigation set by filtering entries in NFR-FR (WRT R_i). Note that each element can only satisfy one partition rule from Table III.

7. CASE STUDY: NFR PROMISE PROJECT 10

We present a case study performing the method steps 1-6 to generate trace assurance investigation sets. Subsequently we will discuss how effective the method was compared to a manual trace assurance. The case study is taken from the PROMISE NFR data set [9]. We selected this data for several reasons – it is publically available (unlike JPL requirements data), our analysis results can be posted to the PROMISE repository for others to verify or replicate, and the requirements require no specialized domain knowledge.

Project 10 (P10) specifies requirements for an online version of a game like "Battleship" and we will not list them here as they are easily accessed online. We selected P10 more or less arbitrarily from the 15 projects in the NFR data set. The only consideration was to ensure both NFRs and FRs were listed and that the project seemed reasonably representative of the requirements data found in NFR. P10 has 15 NFRs and 38 FRs and a manual NFR-FR requirements trace was generated and is illustrated in Fig. 2 to provide an initial feel for the complexity of the trace assurance task at hand.

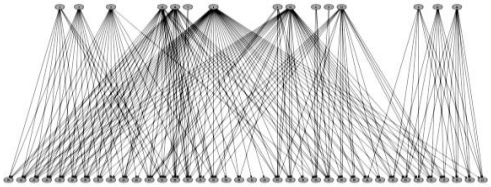

Figure 2. P10 NFR-FR tracability graph.

Step1: P10 is contained in the text file nfr.arff, which is conveniently loadable into a spreadsheet and edited. After cropping out all non-P10 text, we exported the file as a CSV and loaded it as a List object into R and then coerced it into a corpus object. From here the tm package supplies all the functions needed to perform the pre-processing desired (e.g. stemming, stop-words, term-frequencies, etc.). An example original and processed requirement is given below:

NFR1: The product shall simulate the look of ships at sea.

NFR1: simul look ship sea (processed NFR1)

From the frequency-term analysis we found the words "system" and "game" to be generic and these were removed.

Step 2: The pre-processed corpus resulting from step 1 was used to generate a term-document matrix (tdm) from which the dissimilarity matrix

```
D<-tm.dissimilarity(tdm,method="cosine")
```

and similarity matrix

```
S<-similarity(tdm, method="cosine")
```

are generated. These are available on the PROMISE website.

Step 3: The boxplots for the T and NT similarity and dissimilarity sets are shown in Fig. 3.

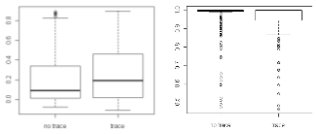

Figure 3. T, NT boxplots for simmilarty (left) and dissimilarty (right).

The two-sided Wilcox p-value between the similarity sets is p=0.04599 indicating that we reject the null hypothesis that

median(T) = median(NT). Performing a one-sided Wilcox test for the null hypothesis median(T) > median(NT) resulted in p=.977 indicating the data is consistent with this hypothesis. For the dissimilarity sets, the two-sided test had p=0.0161 and the one-sided test had p=0.008 indicting that we reject the null hypothesis median(T) > median(NT). This is what we expect as anti-traces should be more dissimilar than traces. The Pearson correlation coefficient between the similarity and dissimilarity value pairs is -0.55, comfortably negative and within the desired range. We are confident that the data is not too noisy to extract meaningful information.

Step 4: The T and NT similarity and dissimilarity data sets were used to generate the LR's using R's ecdf()function:
slr(x)=(1-ecdf(T)(x))/(1-ecdf(NT)(x))
dlr(x)=(1-ecdf(T)(x))/(1-ecdf(NT)(x))
Note that the sets T and NT are generated from **S** for slr(x) and from **D** for dlr(x).

Step 5: Figure 4 shows the LR graphs for slr(x) and dlr(x). Visually we estimate the maximum of slr(x) lies within the 0.6< x < 0.7 range. The maximum around x=7.5 is clearly a noise spike. Using R's optimze() function we were able to determine that the maximum within our estimated range starts to drop after x=0.64 so we conservatively select S_t=0.64 for our similarity threshold. A similar analysis of dlr(x) provides D_t=0.98 for the dissimilarity threshold.

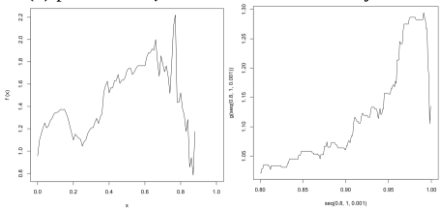

Figure 4. LR graphs for `slr(x)` and `dlr(x)`

Step 6: We apply the rules in Table III to generate investigation sets (sets for Table II were empty or small, so were skipped). It is straightforward to express the rules as list (matrix) index selectors in R. Fig. 5 shows snippets of two different ways to report the investigation sets (you are not expected to read these tables, they are illustrative only). The report on the left provides a compact view while the report on the right uses the investigation sets to annotate the RTM with color to help alert assurance staff of potential issues.

R7	R8	R9	R10	R11	R12	R13	R14
11 25	14 52	4 40	3 42	1 16	2 26	1 31	1 26
11 50		4 41	4 43	1 17	7 39	3 25	1 27
14 22		4 42	5 36	1 18	8 16	3 38	1 28
		5 38	5 37	1 19	8 17	3 39	1 29
		5 39	7 40	1 20	8 18	3 40	1 30
		5 45	11 18	1 21	8 19	4 24	1 40
		5 46	11 20	1 22	8 20	4 25	1 41
		7 41	11 52	1 23	8 23	4 26	1 42
		7 42	11 53	1 24	8 24	4 27	1 43
		7 43		1 25	8 25	4 28	1 48
		11 22		1 32	8 26	4 29	2 23

…	11	12	13	…
21	*R14*	*R11*	*R11*	…
22	*R9*	*R12*	*R13*	…
23	*R11*	*R12*	*R11*	…
24	*R10*	*R12*	*R11*	…
…				

Figure 5. P10 NFR-FR partition investigation sets report examples

More results and trace sets are available through the PROMISE repository [9].

8. EVALUATION

We begin with an independent assessment of the investigation set accuracy. One author went through each set element-by-element assessing the veracity for being in that set (except "no info" sets which make no claims about the requirements). Results are listed in Table IV where each entry *x**y* is read, "*x* were found correct from *y* elements." An (*a*~*b*) entry means the assessor was unsure about *b*-*a* of the elements. These could be correct, but there is some doubt.

TABLE III. Accuracy of investigation sets

R5	R7	R8	**R9**	**R10**	**R11**	**R12**
1\2	3\3	1\1	(8~11)\11	(7~9)\9	330\330	(91~101)\109

Our independent verification gives the investigation sets 95%-98% accuracy. The verification effort took 188 minutes. This is not surprising given that the assessor had to review all but 107 of the 507 potential traces and anti-traces. Next, we had a JPL assurance staff member perform a fully manual P10 trace assurance by means usual to them. Table V compares the results of this effort with the author's assessment guided by the investigation sets generated.

TABLE IV. Comparison of manual and investigation set

	Effort	Missing Traces	False **Traces**	Duplicates	Verified **Traces**	Verified **Anti-Traces**
Manual	227 mins	5	39	2	131	395
Inv-set	94 mins	11	99	2	159	301

Verified trace/anti-trace means that a trace/anti-trace was reviewed and found correct. For the investigation set based assurance, elements in the "low risk" sets R10 and R11 were only "lightly" reviewed to achieve the verification. Here very few elements in the low risk sets were found to be incorrect. In comparison with the manual trace assurance, the set based assurance effort took 58% less effort, found 120% more "high risk" missing traces, and 154% more spurious traces (not so risky, but resource wasteful). The verification rates were comparable, but since any problem found reduces the number of verified elements, it makes little sense to compare the increase or decrease of these. The author's experience in performing the set guided assurance felt more focused and less tedious than the manual approach. While this is wholly subjective, consider if the elements in the low-risk sets were not reviewed at all. This would remove 59% of the trace/anti-trace review size, and assuming a constant effort per trace/anti-trace review, would result in a decrease in 41% of the effort. Given that in this evaluation we saw a 58% decrease in effort, there is likely further efficiencies present than only reducing the number of items to review (and recall that the author did not entirely eliminate review of the low-

risk elements). Neither the author nor the assessor was familiar with P10 beforehand.

Manual trace assurance was performed on 10 of the 15 projects from the NFR PROMISE data set. These, along with the complete details for the P10 evaluation above will be made available there for review.

9. VALIDATION

For this proof-of-concept stage, study validation consists of demonstrating that the method performs as expected under predictable conditions. Later validation studies would address meeting the seven success objectives listed previously and other organization value-oriented criteria. Here we perturb the data in P10 in some controlled manner and compare our expected results from what is observed.

First, we use a random similarity matrix for P10. Here we expect that data will no longer have sufficient information to generate reliable investigation sets. The result we observed was that the Wilcox test on T, NT sets had p-value=0.76 implying the data is consistent with the null hypothesis median(T)=median(NT) giving the expected result.

Next, we use a random trace matrix for P10 for which we again expect the data will no longer have sufficient information. Here the Wilcox test had p-value=0.48 giving the expected result.

Our last validation is to check that the empirical MLE is able to identify an expected threshold value for S_t. For this we generated random similarity values in the range [0.7-1.0] for 291 traces, and random similarity values in the range [-1.0-1.0] for 279 anti-traces. In this case we expect a highly significant difference between median(T) and median(NT). The Wilcox test had a p-value=2.2e-16 strongly rejecting the null hypothesis, as expected. Now, we expect that the maximum for the LR in this case will occur near 0.7, and indeed this is observed in the graph of slr(x) on the left side of Figure 6. Numerical optimization places the maximum at 0.69, well within tolerance. It is also instructive to consider the likelihood ratio $L[T=1 \mid S < S_t] / L[T=0 \mid S < S_t]$ which represents the ratio of false negatives to true negatives. Here we expect that the ratio will rapidly increase after the threshold value 0.7, by design, because prior to this we know there cannot be any traces and after this point we know there will be more and more traces until at S=1 there are about an equal number of them as all traces have $S \leq 1$.

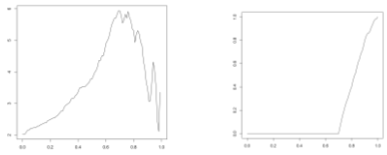

Figure 6. Empirical likelihood ratio function for 0.7 trace set

10. CONCLUSION

Our case study detailed numerous "manual" steps, yet it is clear that much, if not all, of these steps can be automated. Our case study is small, but representative, and observed no inhibitors to scaling the method up to JPL sized projects. If similar results for our case study hold, JPL will save substantial effort, reduce cost, and increase trace assurance effectiveness. As the method shows great promise for meeting the seven objectives vital for success at JPL, we will be initiating a pilot study on select JPL SQA engagements.

Acknowledgment

Our thanks to the experts in assurance from JPL's SQA group (5125) for their participation, feedback, and support. This work is funded in part by the National Science Foundation under NSF grant CCF-0811140.

References

[1] S. Winkler and J. von Pilgrim, "A survey of traceability in requirements engineering and model-driven development," *Software and Systems Modeling*, Dec. 2009.

[2] A. Goldberg, X. Zhu, and S. Wright, "Dissimilarity in graph-based semisupervised classification," *Eleventh International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2007.

[3] J.H. Hayes and A. Dekhtyar, "Humans in the traceability loop: can't live with 'em, can't live without 'em," *Proceedings of the 3rd international workshop on Traceability in emerging forms of software engineering*, Long Beach, California: ACM, 2005, pp. 20-23.

[4] J. Cleland-Huang, A. Dekhtyar, J.H. Hayes, G. Antoniol, B. Berenbach, A. Egyed, S. Ferguson, J. Maletic, and A. Zisman, "Grand challenges in traceability," *TR COET-GCT-06-01-0.9, Center of Excellence for Traceability*, 2006.

[5] A. Dekhtyar, J.H. Hayes, S.K. Sundaram, E.A. Holbrook, and O. Dekhtyar, "Technique Integration for Requirements Assessment," *RE*, 2007, pp. 141-150.

[6] E.A. Holbrook, J.H. Hayes, and A. Dekhtyar, "Toward Automating Requirements Satisfaction Assessment," *Proceedings of the 2009 17th IEEE International Requirements Engineering Conference, RE*, IEEE Computer Society, 2009, pp. 149-158.

[7] J. Cleland-Huang, R. Settimi, X. Zou, and P. Solc, "Automated classification of non-functional requirements," *Requir. Eng.*, vol. 12, 2007, pp. 103-120.

[8] J. Cleland-Huang, R. Settimi, O. BenKhadra, E. Berezhanskaya, and S. Christina, "Goal-centric traceability for managing non-functional requirements," *Proceedings of the 27th international conference on Software engineering*, St. Louis, MO, USA: ACM, 2005, pp. 362-371.

[9] "Predictor Models in Software Engineering (Promise) Software Engineering Repository." http://promise.site.uottawa.ca/SERepository

[10] Landauer, T. K., Foltz, P. W., & Laham, D. (1998). Introduction to Latent Semantic Analysis. Discourse Processes,25, 259-284.

[11] G. Antoniol, G. Canfora, G. Casazza, A.D. Lucia, and E. Merlo, "Recovering Traceability Links between Code and Documentation," IEEE Transactions on Software Engineering/, vol. 28, 2002, pp. 970-983.

[12] A. Marcus and J. Maletic, "Recovering Documentation-to-Source Code Traceability Links using Latent Semantic Indexing," /Proceedings of the Twenty-Fifth International Conference on Software Engineering 2003, 2003, pp. 125-135.

[13] "Recovering Traceability Links between Code and Documentation," IEEE Transactions on Software Engineering/, vol. 28, 2002, pp. 970-983.

[14] A. Marcus and J. Maletic, "Recovering Documentation-to-Source-Code Traceability Links using Latent Semantic Indexing," /Proceedings of the Twenty-Fifth International Conference on Software Engineering 2003, 2003, pp. 125-135.

[15] Landauer, T. K., Foltz, P. W., & Laham, D. (1998). Introduction to Latent Semantic Analysis. Discourse Processes,25, 259-284.

Formatted: Normal