"But it Used to Meet the Requirements!" - Automated Satisfaction Assessment to Support Change

E. Ashlee Holbrook
University of Kentucky
ashlee@uky.edu

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

Alex Dekhtyar
University of Kentucky
dekhtyar@cs.uky.edu

Abstract

When software systems are in use, changes to requirements, design, and code are inevitable. Creating and preserving a requirements traceability matrix (RTM) for a particular software system documents shows how software artifacts are related, and, therefore, supports the impact assessment of potential changes as maintenance proceeds. Suppose that a particular software system completely implements or satisfies all of its original requirements, and then a series of new requirements are levied. Information on which software artifacts help meet the new requirements and which new requirements are not addressed would be very useful to project managers and development teams. Likewise, information about requirement satisfaction can be useful when reusing design or code, such as when determining whether a pre-existing system can meet the requirements of a new project. This paper presents a system that automatically generates information about requirement satisfaction of textual requirements by textual design elements to support software maintenance and reuse. An empirical study was performed to compare two methods for automated satisfaction assessment, resulting in one method performing more accurate assessments that can be more easily reviewed by an analyst.

1. Introduction

In an ideal world, creating software that is easy to maintain and of high quality would be as simple as following the steps of the software development lifecycle. Complete, correct, and consistent requirements would be elicited from the users of the software. These requirements would be translated into design that is also complete and correct. Code would be created to implement the design and would be easily adaptable to future changes. Testing would cover key execution paths and would discover any remaining errors in the code. In addition to being complete and correct, all of the artifacts from the software development lifecycle would be clear and easily understood. Unfortunately, this scenario rarely occurs. Errors are introduced at each stage of development, and omissions often make software maintenance difficult. The sooner these errors are discovered and corrected, however, the less devastating they are in terms of development time and project budget [3].

One method software engineers have developed to ensure consistency and to assist in maintenance is the creation of a requirements traceability matrix (RTM) to trace requirements to design to code and so on. A sample RTM is shown in Figure 1. An RTM consists of a list of requirements mapped to design elements and/or a list of design elements mapped to code segments, test cases, user manual sections, etc. Ideally, an RTM is created and modified along with the elements it maps. While this is the most efficient way to create an RTM, RTMs are rarely prepared this way. Even if an RTM is created as software is developed, it is often too high-level to be useful for maintenance and quality assurance, thus after-the-fact tracing is necessitated [1, 4, 12, 13, 17].

Creating an RTM after-the-fact is a tedious and error-prone process. While automated tracing techniques have been introduced to assist analysts in this task [1, 4, 12, 13, 17], the question remains as to whether elements traced to one another are simply related or exhibit a true satisfaction relationship. ISO 9000:2000 [15] defines software quality as the "degree to which a set of inherent characteristics fulfill requirements," where a requirement is a "need or expectation that is stated, generally implied, or obligatory." This measure of quality is also known as requirement satisfaction. A satisfaction relationship between a requirement and design element is one where the design element either fully addresses and captures the meaning of the requirement or partially does so. In

	DE1	DE2	DE3	DE4
R1		X		
R2	X			X
R3			X	X
R4	X	X		
R5		X		X

Key: R1 – Requirement 1, DE1 – Design Element 1

Figure 1. Sample RTM.

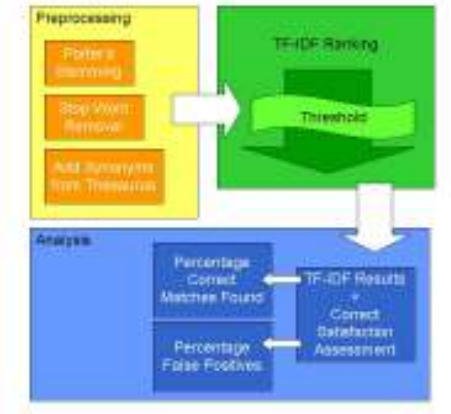

Figure 2. TF-IDF Satisfaction Assessment.

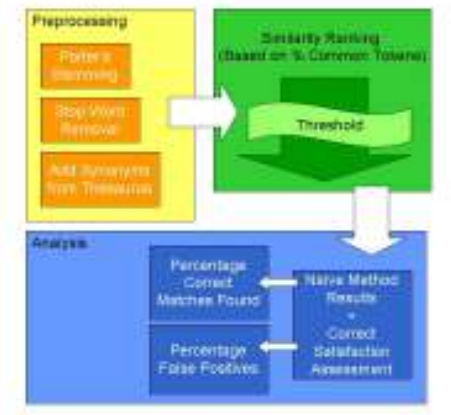

Figure 3. Naïve Satisfaction Assessment.

the latter case, a requirement may be satisfied by a combination of several design elements.

Satisfaction assessment is the process of determining which pieces of a requirement are satisfied by which pieces of a set of design elements. In addition to providing a measure of system quality, knowing the satisfaction assessment of a set of requirements and design elements will allow one to see how potential maintenance efforts will affect a system, how well a system will be suited to reuse on a future project, and how well an existing system meets both preexisting and newly-introduced requirements.

The contributions of this paper are 5-fold: we define the satisfaction assessment problem for textual requirements; we frame this problem as an information retrieval problem; we present two methods for performing satisfaction assessment as implemented in our REequirements SATatisfaction (RESAT) tool; we identify a number of metrics for validating satisfaction assessment; and we undertake an empirical study on the efficacy of RESAT.

An overview of satisfaction assessment will be presented in Section 2. Related work will be described in Section 3. Section 4 will address the empirical study design. Results will be presented in Section 5 and conclusions will be described in Section 6. Finally, future directions will be indicated in Section 7.

2. Satisfaction

Our approach to satisfaction assessment is two-pronged. In this paper, we study the problem of determining matches between portions of individual requirements and design elements. Our next step will be to investigate whether we can reduce the problem of requirements satisfaction to the previously mentioned matching problem.

In particular, we model each requirement and each design element as a collection of phrases, or "chunks," such that each chunk represents a single part of a requirement or design element. In this study, we offer a solution to the problem of establishing satisfaction mappings between individual pairs of requirement and design element chunks. Our assumption is that for a requirement to be completely satisfied, all of its important chunks must be addressed by subsequent chunks in the design element text. .

In this paper, satisfaction assessment is defined as the process of determining the satisfaction mapping of natural language textual requirements to natural language design elements. Suppose we are given a set of requirements, R, broken down into terms ($R = \{t_{r1}, t_{r2}, \ldots\}$) where t_{r1} is the first term in R, or phrases ($R = \{p_{r1}, p_{r2}, \ldots\}$) where p_{r1} is the first phrase in R. Suppose that we also have a set of design element terms, D, ($D = \{t_{d1}, t_{d2}, \ldots\}$) where t_{d1} is the first term in D, or phrases ($D = \{p_{d1}, p_{d2}, \ldots\}$) where p_{d1} is the first phrase in D. A satisfaction mapping is a set of pairs of terms (t_{rn}, t_{dm}) where t_{rn} is a term in a set of requirements, and t_{dm} is a term in the set of design elements where t_{rn} is directly related to t_{dm}. A satisfaction mapping may also occur at the phrase level. In this case it will consist of a series of phrase pairs

a) Original Text:
The DPU CCM shall implement a mechanism whereby large memory loads and dumps can be accomplished incrementally.

b) Stop Word Removal:
dpu ccm implement mechanism whereby memory loads dumps accomplished incrementally

c) Stemming:
dpu ccm implement mechanic whereby memori load dump accomplish increment

d) Thesaurus Tagging:
dpu ccm|command|control|modul|
implement|accomplish|code
routine|call|funct|operat|mechani wherebi
memori|stor|disk load dump accomplish
increment|period

e) Sample Domain-Specific Thesaurus Entries:
collect|gather|accumulate
memory|storage|disk
implement|accomplish|code
dpu_hk|housekeeping
incrementally|periodically
routine|call|function|operation|mechanism
ccm|command|control|module

f) Tokenization:
[The] [DPU] [CCM] [shall] [implement] [a] [mechanism] [whereby] [large] [memory] [loads] [and] [dumps] [can] [be] [accomplished] [incrementally].

g) Chunking:
[The DPU CCM] [shall implement] [a mechanism] [whereby] [large memory loads and dumps] [can be accomplished] [incrementally].

Figure 4. Textual Processing.

(p_{rn}, p_{dm}) with one phrase, p_{rn}, being a phrase in a requirement, and p_{dm} being a phrase in a corresponding design element where p_{dm} directly addresses p_{rn}.

In this work, requirements and design elements are assessed in natural language form. No formal standardization or formatting techniques are applied because specifying a particular format for input would place an additional burden on individuals who write the requirements and design. An independently verified RTM, such as the one shown in Figure 1, is also given as input to show which design elements are traced to each requirement. The satisfaction assessment process considers, for each requirement, only the design elements that were traced to it in an RTM. This provides a basis for the satisfaction assessment and limits satisfaction mappings to only those requirement-design element pairs that are at least minimally related.

In order to determine satisfaction for this work, a four step process was applied. First, each textual requirement and design element was preprocessed. Preprocessing is a two stage process consisting of stemming and stop word removal. These steps are explained in more detail in Section 2.1. Second, a domain specific thesaurus, which contains a set of synonym pairs for domain-specific vocabulary was applied. Next, each requirement and design element was tokenized into chunks based on parts of speech. Each chunk consists of a phrase in a sentence. Chunking took place by parsing sentences and then assigning each grammatical part of the sentence as an individual chunk. Finally, similarity measures were calculated between chunks of requirements and the chunks of design elements they are tied to in the project RTM. For this work, similarity measures were calculated using vector space retrieval with term frequency-inverse document frequency (TF-IDF) weighting, an information retrieval technique [24].

Information retrieval (IR) is a field that studies the problem of information discovery in textual documents. Each of the phases above will be described in further detail in the following sections. Incremental processing of text is shown in Figure 4. 4a shows the original text of a sample requirement. 4b and 4c show preprocessing steps. Thesaurus tagging and a sample thesaurus are shown in 4d and 4e. Figures 2 and 3 show the overall satisfaction assessment process for the two methods that were implemented for the empirical study

2.1. Preprocessing

Two preprocessing techniques were used in the automated satisfaction methods presented here. First, stop words were removed from the requirements and design elements. Stop word removal filters out words that may be selected by information retrieval techniques as keywords, but are not true keywords. These are common words such as “of” and “the” that occur frequently in text, but do not add significant semantic information. For this study, the Fox stop word list of 421 common English terms was used [8].

Next, remaining terms were stemmed. Stemming removes suffixes from terms to ensure that related terms may be compared. For example, the words

"construction," "constructed," and "constructing" would all stem to "construct." This enables information retrieval techniques to better compare words and determine how often keywords occur in text. Porter's stemming algorithm, the standard English stemming algorithm, was used in this work [19].

2.2. Thesaurus Application

For this work, a domain-specific thesaurus was created based on a subset of the NASA CM-1 data set [5]. It contains entries for words and their corresponding synonyms found in the data set, but not in a standard thesaurus (i.e., 'dpu_hk' which has a thesaurus synonym of 'housekeeping'). The CM-1 data set includes C code, requirements, design, and error data for a NASA scientific instrument. Terms throughout the requirements and design were tagged with related words from the thesaurus [See Figure 4 (d and e)]. For the sake of comparison, a single term from the list of synonyms was substituted for all words in each synonym set.

2.3. Tokenization and Chunking

Tokenization is the process of demarcating, or breaking text down into, individual tokens. Tokens are typically single words. A sample tokenized requirement is shown in Figure 4f. Chunking is a similar process, but occurs at the phrase level, as is shown in Figure 4g [11]. Chunking plays a key role in satisfaction work. Rather than simply performing tokenization or sentence splitting, chunking allows one to parse a sentence into meaningful pieces that can be used for satisfaction assessment. Chunking and tokenization for this study were handled using OpenNLP libraries [18].

2.4. TF-IDF Analysis

Term frequency–inverse document frequency (TF-IDF) is a statistical measurement of the importance of a term within a document [24]. Term frequency is the number of times a term appears within a document, and inverse document frequency is a way to measure term importance over a set of documents (document frequency, or the logarithm of the number of documents that have that term). Formally, TF-IDF weights are assigned to each term in each document:

$$\text{TF-IDF} = \frac{n_i}{\sum_k n_k} * \log \frac{|D|}{|\{d : d \ni t_i\}|}$$

where n_i is the number of occurrences of a term, $\Sigma_k n_k$ is the occurrences of all terms, $|D|$ is the total number of documents, and $|\{d:dət_i\}|$ is the number of documents containing term t_i [24]. Vectors containing these weights are constructed. To calculate TF-IDF similarity scores for satisfaction assessment between each requirement chunk vector, v_r, and design element vector, v_{rd}, is taken:

$$sim = (v_r \times v_d) / (| v_r | | v_d |).$$

This yields a similarity score between 0 and 1, where 0 indicates that the two chunks have no relevancy and 1 indicates that the chunks are identical.

For this work, the set of documents was set to be the entire set of chunks from both requirements and design elements. A similarity measure based on the difference in weights for each requirement chunk and each design element chunk for design elements mapped by the input RTM was calculated. Thresholds of 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.8, and 0.9 were applied to determine relevant related design element and requirement chunk pairs. A threshold value of 0.5, for example, indicates that if there is a similarity score of at least 0.5 between a requirement chunk and a design chunk, these two chunks should be included as a pair in the satisfaction mapping. Requirement chunks with design elements chunks that had similarity measures above the threshold value were marked as being satisfied by the subsequent design element chunks with a confidence level equal to the similarity score.

2.5. Naïve Satisfaction Analysis

Another satisfaction method, the naïve satisfaction method was also developed for this study. Naïve satisfaction looks only at keyword matches within requirement and design chunks. In this method, each requirement and design element chunk pair, (p_{rn}, p_{dm}), is assigned a similarity score. This is calculated by tokenizing each chunk into individual terms. The number of matching terms divided by the total number of terms in both the requirement and design element provide a similarity score. Requirement and design element chunk pairs with a similarity score above a given threshold will be included in the satisfaction mapping. As in TF-IDF, nine threshold values were used. For naïve satisfaction, these were: 0.02, 0.03, 0.04, 0.05, 0.06, 0.07, 0.08, 0.09, and 0.1.

Table 1: Satisfaction Assessment Quality Measures for each TF-IDF Threshold

Threshold	Recall	Precision	Number of Corrections	Selectivity	Candidate Matches	Correct Matches
0.1	56.38%	30.10%	1545	2.02%	1658	499
0.2	53.56%	31.29%	1452	1.84%	1515	474
0.3	48.25%	33.18%	1318	1.57%	1287	427
0.4	40.68%	37.42%	1127	1.17%	962	360
0.5	31.19%	42.27%	986	0.34%	276	653
0.6	22.60%	42.74%	953	0.57%	468	200
0.7	17.40%	44.90%	920	0.42%	343	154
0.8	13.90%	51.25%	879	0.29%	240	123
0.9	12.32%	50.70%	882	0.26%	215	109

Table 2: Satisfaction Assessment Quality Measures for each Naïve Satisfaction Threshold

Threshold	Recall	Precision	Number of Corrections	Selectivity	Candidate Links	Answer Links
0.02	62.94%	11.23%	4733	6.04%	4962	557
0.03	62.49%	11.22%	4707	6.00%	4928	553
0.04	60.23%	11.23%	4564	5.78%	4745	533
0.05	55.59%	12.35%	3884	4.85%	3983	492
0.06	52.77%	12.22%	3773	4.65%	3822	467
0.07	45.42%	14.39%	2874	3.40%	2793	402
0.08	44.29%	14.22%	2857	3.36%	2756	392
0.09	32.77%	18.69%	1857	1.89%	1552	290
0.1	27.80%	18.07%	1754	1.66%	1361	246

3. Related Work

Satisfaction may be thought of as a way to validate whether requirements have been fully addressed by design elements and provides a way to measure the quality of a software project. Previous work on requirements validation has focused on formally specifying requirements [28][22][10], optimizing natural language processing (NLP) approaches to requirements analysis, and discovering potential ambiguities [16][9][7]. Preliminary work on applying NLP to requirements [23] and on understanding requirements [21] has been completed.

Durán et al. used XSTL and requirements in XML to automatically verify requirement qualities such as completeness and lack of ambiguity [6]. Analysts have used requirement defect detection techniques [20] to discover requirements that cannot be satisfied (i.e. inconsistent and omitted) and inconsistencies between requirements and design. Reading methods such as scenario-based [27] and perspective-based reading [2, 26] have also been used to increase the quality of requirement specifications.

Extensive research in requirements tracing has been undertaken. Tracing looks at the creation of a requirements traceability matrix (RTM) that relates requirements to design to code and beyond. Much recent work has been completed on applying information retrieval (IR) methods to tracing. Antoniol et al. [1] and Marcus and Maletic [17] have applied IR methods to the problem of tracing design to code. Cleland-Huang [4] has used IR to trace non-functional requirements. Hayes et al. have investigated the process of tracing and have built a special-purpose requirements tracing tool called RETRO (REquirements TRacing On-target) [12, 13].

Algorithmic techniques that are useful for both assessing requirement satisfaction and tracing include keyword extraction methods [14] and vector space models [24]. Vector space models represent documents as vectors by extracting terms, weighting these by relevance and document location, and ranking the document as a whole based on a given query.

4. Empirical Assessment

Ensuring that requirements are fully satisfied by software design elements is an important part of the software lifecycle and helps lay the foundation for a

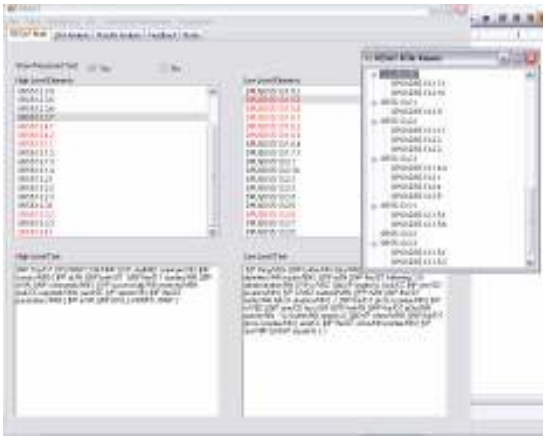

Figure 5. RESAT.

quality software product. Two satisfaction assessment methods have been developed to address this concern. In order to test the TF-IDF and naïve satisfaction methods, an empirical study was undertaken. For this study, TF-IDF Satisfaction Assessments for nine threshold values were compared to a naïve satisfaction assessment approach for nine threshold values. The perspective of this study was from the traceability/satisfaction assessment researcher's point of view.

4.1. Empirical Study Overview

Our empirical study examines two methods of automated satisfaction assessment – naïve satisfaction assessment using textual matching and vector space satisfaction assessment with TF-IDF weighting. The study measures effort required in terms of number of corrections and selectivity and satisfaction assessment quality in terms of recall and precision. These measures are further described in Section 4.2.2.

4.2. Empirical Study Design

For this work, a satisfaction assessment tool, RESAT (REquirements SATisfaction), was developed (See Figure 5). The tool offers various methods for satisfaction assessment, one of which is the TF-IDF satisfaction assessment method described above. The tool loads plaintext requirements and design elements and provides a series of options that enables users to perform preprocessing steps, natural language processing, chunking and tokenization, and information retrieval techniques on the text. The final output is a version of the input with markup to indicate satisfaction assessment results.

4.2.1. Data Set. For this study, a subset extracted from the NASA CM-1 data sets, available through our research lab and the PROMISE repository, was used [5, 25]. CM-1 is a scientific instrument used by NASA. The data set contains plaintext requirements, design elements, and code along with tracing and error data for each stage. The entire CM-1 data set contains 235 high-level elements (requirements) and 220 low-level elements (design elements). From these, a subset of 22 high-level elements and 52 low-level elements was randomly selected. The RTM for CM1 contains 95 true links with an average of 4.3 design elements linked to each requirement. A satisfaction answer set was created independently by two analysts. Then, the analysts discussed differences between their two answer sets until consensus was reached. The satisfaction answer set had 885 correct requirement and design element chunk pairs based on 82,138 possible matches.

4.2.2. Measures. Measurements collected include recall, precision, selectivity, and number of corrections. Recall is defined as the ratio of the number of true satisfaction pairs found to the total number of true satisfaction pairs found in the answer set satisfaction assessment. Precision is the percentage of satisfaction pairs returned in the satisfaction assessment that are true satisfaction pairs. The number of corrections is the number of switches that would have to be made to transform a candidate satisfaction assessment to the answer set satisfaction assessment. For number of corrections measurement, errors of commission (including false pairs) and errors of omission (omitting true pairs) were each given equal weighting (one correction each). Selectivity is defined as the number of matches returned divided by the total number of possible match pairs.

Satisfaction assessments that have high precision and low number of corrections are beneficial to analysts because they provide a good starting point for the tedious task of satisfaction assessment, but will not include a great deal of potentially disastrous false positives (false positives indicate that the method has assumed a requirement is satisfied by design elements, when, in fact, it is not). In satisfaction, false positives may give analysts a false sense of confidence about the quality of requirement and design specifications, so satisfaction mappings should be assigned cautiously. Recall, as in IR, should be maximized and selectivity should be minimized.

The independent variable for this study was the satisfaction assessment method used (TF-IDF

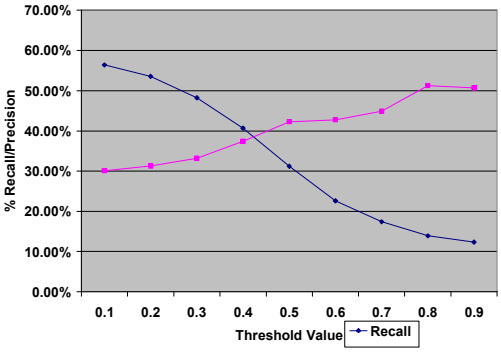

Figure 7. Recall and Precision for TF-IDF Method.

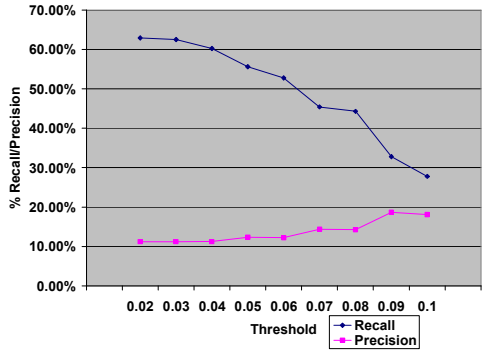

Figure 8. Recall and Precision for Naïve Method.

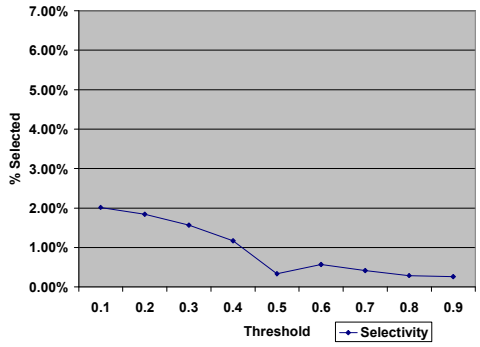

Figure 9. Selectivity for TF-IDF Method.

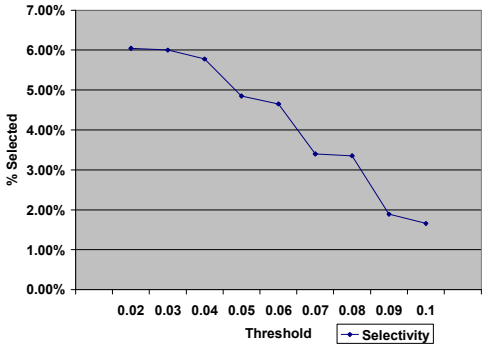

Figure 10. Selectivity for the Naïve Method.

satisfaction assessment method versus naïve satisfaction method, each for nine threshold values). The dependent variables were recall, precision, number of corrections, and selectivity. The purpose of this study was to see if the TF-IDF satisfaction assessment method will yield a satisfaction assessment the CM1 data set that is better than the naïve satisfaction assessment method.

4.2.3. Hypotheses. The purpose of this study was to test the naïve and TF-IDF satisfaction assessment methods. For this work, we will assume that an analyst's workload would be significantly reduced if a satisfaction mapping yields at least 30% of the correct matches, with no greater than 10% of the matches returned being false positives. Thus the null and alternative hypotheses to be evaluated in this study are:

- $\mathbf{H_{01}}$: There will be no difference in precision for the Naïve Satisfaction Method and the TF-IDF Satisfaction Method 2.
- $\mathbf{H_{A1}}$: There will be a difference in precision for the Naïve Satisfaction Method and the TF-IDF Satisfaction Method 2.

- $\mathbf{H_{02}}$: There will be no difference in recall for the Naïve Satisfaction Method and the TF-IDF Satisfaction Method 2.
- $\mathbf{H_{A2}}$: There will be a difference in recall for the Naïve Satisfaction Method and the TF-IDF Satisfaction Method 2.

- $\mathbf{H_{03}}$: There will be no difference in the number of corrections for the Naïve Satisfaction Method and the TF-IDF Satisfaction Method 2.
- $\mathbf{H_{A3}}$: There will be a difference in the number of corrections for the Naïve Satisfaction Method and the TF-IDF Satisfaction Method 2.

- $\mathbf{H_{04}}$: There will be no difference in selectivity for the Naïve Satisfaction Method and the TF-IDF Satisfaction Method 2.
- $\mathbf{H_{A4}}$: There will be a difference in selectivity for the Naïve Satisfaction Method and the TF-IDF Satisfaction Method 2.

4.3. Validity

Our solution to satisfaction assessment is a domain-specific solution and uses a domain-specific thesaurus, so there is a threat to external validity. The satisfaction

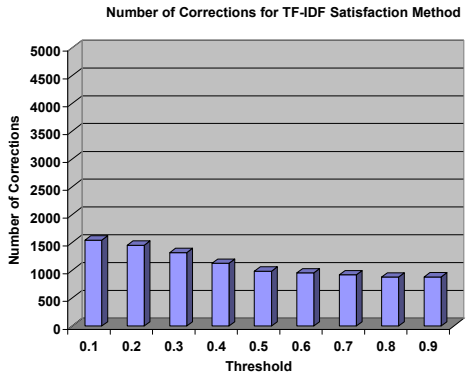

Figure 11. Number of Corrections for TF-IDF Satisfaction Method.

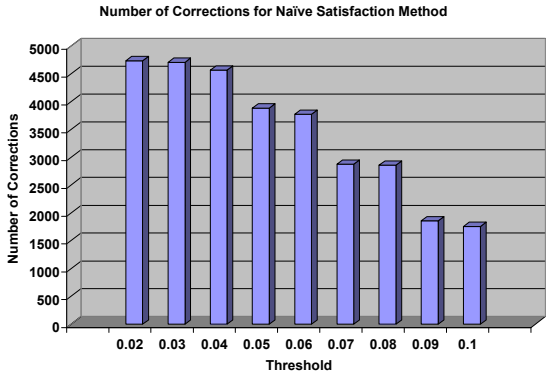

Figure 12. Number of Corrections for Naïve Method.

problem is not generally solvable, so the domain constraint is reasonable in this case. No human subjects were used in this study, so validity threats based on human interaction, training, etc. are avoided.

This study avoids problems with conclusion validity in the sense that a mapping returned will be based solely on the preprocessing and the naïve satisfaction/TF-IDF method results. However, the grammatical complexity, conformity, and language used in the data sets may influence the results. Additionally, due to the complexity and domain specific terms, the answer set used for comparison may not be completely accurate and precise. The answer set was created by human analysts that are familiar with the traceability research domain, so there was the potential for bias. Creating a unified answer set from two independent answer sets helped address this concern. The quality of the domain-specific thesaurus may have also had influence on study results; however, the same thesaurus was used for both methods to reduce the impact of this as much as possible. These threats are intrinsic to the problem studied not to the experimental design.

5. Results and Analysis

In the case of each of the four hypotheses for this study, the null hypothesis was rejected in favor of the alternative hypothesis. Tables 1 and 2 show raw data for each of the two methods. Univariate analysis of variance (ANOVA) showed that for recall, precision, number of corrections, and selectivity, there was a statistically significant difference at the 0.05 level. Tables 3-6 show results from the statistical analysis.

Figures 7 and 8 show the recall and precision for each method. As can be seen, the TF-IDF method provided much higher precision for a given level of recall. Precision is critical to automated satisfaction assessment because if a method determines that a requirement is satisfied, the analyst may accept this assessment. If the requirement was, in fact, not satisfied by the design, the final design will be incomplete. Higher recall values indicate that the analyst will have a more complete picture of the satisfaction assessment of a set of requirements and design elements.

Figures 9 and 10 show selectivity for each method. While both methods were highly selective, selectivity was lower, and therefore better, for TF-IDF. Likewise, the number of corrections necessary for the TF-IDF method was lower than for the naïve method, meaning that an analyst will spend less time correcting the output from the TF-IDF method than the output from the naïve method. The number of corrections for each of the two methods are depicted in Figures 11 and 12. As is shown, there is a large difference in the number of corrections for the TF-IDF method and for the naïve method, with the naïve method requiring many more corrections.

6. Conclusions

Assessing the satisfaction of a set of requirements by the corresponding design elements proves invaluable in maintenance and reuse. Satisfaction assessments provide insight into how a potential change can impact a software system, show how well an existing system fits a new series of requirements for reuse, and provide a valuable way to measure the quality of an existing software system. This study has presented two approaches to satisfaction assessment, a naïve method and a method using information retrieval and TF-IDF. Through an empirical study it was shown

Table 3. ANOVA Results for Recall.

Source	Mean Square	F	Sig.
Corrected Model	1217.218	5.243	.036
Intercept	30469.930	131.237	.000
Method	1217.218	5.243	.036
Error	232.175		
Total			
Corrected Total			

a R Squared = .247 (Adjusted R Squared = .200)

Table 4. ANOVA Results for Precision.

Source	Mean Square	F	Sig.
Corrected Model	3206.136	89.963	.000
Intercept	13201.500	370.428	.000
Method	3206.136	89.963	.000
Error	35.639		
Total			
Corrected Total			

a R Squared = .849 (Adjusted R Squared = .840)

that the TF-IDF method shows statistically significant improvements over the naïve text matching approach and thus provides a basis for future research in this area.

7. Future Work

The empirical study presented in this paper tests the efficiency of two techniques for satisfaction assessment, the TF-IDF satisfaction assessment method and a naïve satisfaction approach based on co-occurrence of terms. This work provides a foundation upon which to address a variety of maintenance concerns: "How does one determine the quality of a software system based on requirements and design?", "If one is reusing a pre-existing software system, how does it address a new set of requirements and what additions will be needed to fully satisfy the requirements for the new project?", and "How does one assess the impact of proposed design changes on existing requirements?" The next step in this work is to see how to apply the results from automated satisfaction assessment to these research questions. To assist an analyst in deciding whether a requirement has been satisfied by its design elements, the satisfaction level for each requirement chunk must be assessed. This work provides a foundation for satisfaction assessment at the chunk level, but does not yet provide a method for making satisfaction decisions at the requirement level. This is left as future work.

Table 5. ANOVA Results for Number of Corrections.

Source	Mean Square	F	Sig.
Corrected Model	24362526.722	34.230	.000
Intercept	93685234.722	131.630	.000
Method	24362526.722	34.230	.000
Error	711731.222		
Total			
Corrected Total			

a R Squared = .681 (Adjusted R Squared = .662)

Table 6. ANOVA Results for Selectivity.

Source	Mean Square	F	Sig.
Corrected Model	47.207	28.021	.000
Intercept	118.118	70.112	.000
Method	47.207	28.021	.000
Error	1.685		
Total			
Corrected Total			

a R Squared = .637 (Adjusted R Squared = .614)

Future research work also includes implementing and testing a variety of other techniques for satisfaction assessment, improving results of the TF-IDF method, and testing the TF-IDF satisfaction assessment method on a variety of other data sets both from the application domain tested here and for other application domains.

8. Acknowledgements

This work is sponsored by NASA under grant NNG05GQ58G. Thanks to Stephanie Ferguson, Marcus Fisher, Ken McGill, Tim Menzies, and everyone at the NASA IV&V facility. Thanks also to the MODIS project and to fellow graduate students Jody Larsen, Senthil Sundaram, Liming Zhao, and Sravanthi Vadlamudi.

9. References

[1] G. Antoniol, et. al. "Recovering Traceability Links between Code and Documentation", *IEEE Trans. on Software Engineering*, Volume 28, No. 10, 2002, 970-983.

[2] V.R. Basili,, et. al. "The Empirical Investigation of Perspective-Based Reading." *Empirical Software Engineering*, 1(2), 1996, 133-164.

[3] B.W. Boehm, Software Engineering, *IEEE Trans. On Computers*, 25(12):1226-1241, 19.

[4] J. Cleland-Huang, et.al. "Goal-Centric Traceability for Managing Non-Functional Requirements", *Int. Conference on Software Engineering*, Saint Louis, May 2003.

[5] "CM1 Data Set," Metrics Data Program Website, CM-1 Project, http://mdp.ivv.nasa.gov/mdp_glossary.html#CM-1.

[6] A. Durán, A Ruiz, M. Toro, "An Automated Approach for Verification of Software Requirements", *Jornadas de Ingeniería de Requisitos Aplicada*, Seville, Spain, 2001.

[7] C. Denger, D.M. Berry, and E. Kamsties, "Higher Quality Requirements Specifications through Natural Language Patterns," *IEEE Software-Science*, Technology & Engineering (SwSTE'03), pp. 80-89, Israel, Nov. 2003.

[8] C. Fox, "A Stop List for General Text." *SIGIR Forum* 24, 1-2 (Sep. 1989), 19-21.

[9] L. Goldin, and Berry, D.M., ``AbstFinder, A Prototype Natural Language Text Abstraction Finder for Use in Requirements Elicitation" *Automated Software Eng.*, 4(4), 375-412, Oct., 1997.

[10] S. Greenspan, J. Mylopoulos, A. Borgida, "On Formal Requirements Modeling Languages: RML Revisited", *Proc. 16th International Conference on Software Engineering*, p.135-147, May 16-21, 1994, Sorrento, Italy.

[11] K. Hacioglu, S. Pradhan, W. Ward, J. H. Martin, and D. Jurafsky. "Semantic Role Labeling by Tagging Syntactic Chunks." *In Proceedings of CoNLL* 2004 Shared Task.

[12] J.H. Hayes, A. Dekhtyar, J. Osbourne, "Improving Requirements Tracing via Information Retrieval", *Int. Conf. on Requirements Engineering*, Monterey, California, Sept. 2003,pp. 138 – 148.

[13] J.H. Hayes, A. Dekhtyar, and S. Sundaram, "Advancing Requirements Tracing: The Study of Methods", *IEEE Trans. on Software Engineering*, 32(1), Jan. 2006, pp. 4 -19.

[14] J.H. Hayes, A. Dekhtyar, S. Sundaram, "Advances in Dynamic Generation of Traceability Links", Tech Report, February 2006, (TR 451-06).

[15] ISO 9000 (2000). *Quality Management Systems – Fundamentals and Vocabulary*. International Organization for Standardization.

[16] R. Lecceuche, "Finding Comparatively Important Concepts between Texts." *Automated Software Engineering* (ASE'00). Washington, DC, 55.

[17] A. Marcus, J. Maletic, "Recovering Documentation-to-Source Code Traceability Links using Latent Semantic Indexing", *Software Engineering*, Portland, May 2003, pp. 125−135.

[18] "OpenNLP," Open Natural Language Processing project. Available for download at: http://opennlp.sourceforge.net/about.html.

[19] M.F. Porter, 1980, "An Algorithm for Suffix Sstripping," *Program*, 14(3) pp 130−137.

[20] A. Porter and L. Votta, "Comparing Detection Methods For Software Requirements Inspections" *Empirical Software Engineering*, 3(4), 1998, 355 – 379.

[21] P. Rayson, R. Garside, and P. Sawyer," Recovering Legacy Requirements." *Requirements Engineering: Foundations of Software Quality,* June 14-15 1999, Heidelberg, Germany, pp. 49-54.

[22] W.N. Robinson, S. Pawlowski, "Managing Requirements Inconsistency with Development Goal Monitors," *IEEE Trans. on Software Eng*., Nov/Dec 1999.

[23] K. Ryan, "The Role of Natural Language in Requirements Engineering." *Requirements Engineering* (RE'93), San Diego, pp. 80-82, 1993.

[24] G. Salton, *Introduction to Modern Information Retrieva*l. McGraw-Hill, 1983.

[25] J. Sayyad Shirabad, and Menzies, T.J. (2005) *The PROMISE Repository of Software Engineering Databases*. School of Information Technology and Engineering, University of Ottawa, Canada. Available: http://promise.site.uottawa.ca/SERepository.

[26] F. Shull, I. Rus, and V.R. Basili, "How Perspective-Based Reading Can Improve Requirements Inspections," *IEEE Computer*, 33(7), pp. 73-79, July 2000.

[27] A. Sutcliffe, "Scenario-Based Requirement Analysis", *Requirements Engineering Journal* 3(1), 1998, 48−65.

[28] J.M. Spivey 28, "*Understanding Z*," Cambridge Press, 88.