Experiences with Text Mining Large Collections of Unstructured Systems Development Artifacts at JPL

Dan Port
University of Hawaii
Shidler College of Business
Honolulu, HI 96822
+1-808-956-7494
dport@hawaii.edu

Allen Nikora, Jairus Hihn
Jet Propulsion Laboratory
California Institute of Technology
Pasadena, CA 91109-8099
+1-818-393-1104
{allen.p.nikora, jairus.hihn}@jpl.nasa.gov

LiGuo Huang*
Southern Methodist University
Computer Science & Engineering
Dallas, TX 75275-0122
+1-214-768-3709
lghuang@smu.edu

ABSTRACT

Often repositories of systems engineering artifacts at NASA's Jet Propulsion Laboratory (JPL) are so large and poorly structured that they have outgrown our capability to effectively manually process their contents to extract useful information. Sophisticated text mining methods and tools seem a quick, low-effort approach to automating our limited manual efforts. Our experiences of exploring such methods mainly in three areas including historical risk analysis, defect identification based on requirements analysis, and over-time analysis of system anomalies at JPL, have shown that obtaining useful results requires substantial unanticipated efforts - from preprocessing the data to transforming the output for practical applications. We have not observed any quick "wins" or realized benefit from short-term effort avoidance through automation in this area. Surprisingly we have realized a number of unexpected long-term benefits from the process of applying text mining to our repositories. This paper elaborates some of these benefits and our important lessons learned from the process of preparing and applying text mining to large unstructured system artifacts at JPL aiming to benefit future TM applications in similar problem domains and also in hope for being extended to broader areas of applications.

Categories and Subject Descriptors

D.2.9 [Management]: Software quality assurance (SQA).

General Terms

Management, Verification.

Keywords

Experience, Text Mining, System Repository Mining, Systems Development Artifact, Assurance, Risk, Risk Assurance, Requirements Assurance.

1. INTRODUCTION

Today's NASA projects produce tens of thousands of important inter-project artifacts such as requirements, systems risks, anomaly reports and lessons learned. However, these artifacts are in general poorly structured and cumbersome to work with at large scale. Despite NASA's policies mandating the archival of project records and lessons learned as well as its explorations into the field of knowledge management, many of the artifact repositories that have been built for well over a decade at NASA's Jet Propulsion Laboratory (JPL) have outgrown our ability to effectively manually process their contents to extract useful information.

These repositories exist to support project development and management decision making, guide assurance activities, and preserve organizational memory. They provide valuable historical data for such tasks as identifying lessons learned, recognizing systemic risks, understanding the causes of cost growth, and determining defect root causes. The artifacts are generally documented in natural language, which are relatively unstructured and vary widely in file and data formatting, naming conventions and use of language (e.g. terms). Given this and the sheer magnitude of data now available in the repositories it is not surprising it exceeds a humans' ability to comprehensively understand and analyze it. Owing to this, we were led to explore if text mining (TM) techniques could help expand our capabilities since they are specifically designed to analyze large sets of unstructured artifacts such as we have. Given the availability of mature and sophisticated TM tools and methods, we imagined it would be relatively straightforward to apply TM to extract the comprehensive and useful information to support objective decision making and to improve the quality of system development. For example, we have applied TM techniques to over 400 mission feasibility studies to help generate an annual top-10 systems risk list, perform risk predictions from study keywords, assure completeness of risk identification, investigate risk changes over time, and conduct other artifact repository processing efforts that humans are not good at. TM techniques have been widely applied in open source domains. With promises in the literature advertising capabilities that swiftly find valuable information in large, amorphous document repositories, TM is deceptively attractive. However, our experiences over the past 5 years exploring TM techniques at JPL have shown us that obtaining useful results in practice requires substantial unanticipated effort in selecting and configuring the TM methods and tools, preprocessing the data for these tools, and transforming tool outputs for practical use by humans.

[*] Corresponding author: LiGuo Huang

Here we have not yet experienced any quick "wins" or short-term effort avoidance by use TM for automating our artifact repository processing. Indeed, after nearly 8 months of effort by four professional developers and researchers we were only able to prepare 180 of the 400 feasibility studies mentioned previously for TM. Nevertheless this effort was not in vein. Through this effort we discovered a number of unexpected long-term benefits from applying TM to our repositories. This paper mainly discusses our TM experiences in historical risk analysis, defect identification/characterization based on requirements analysis, and over-time analysis of system anomalies at JPL. It details 11 lessons we have learned "the hard way" from three of our experiences applying TM to large unstructured system artifact repositories (i.e., historical risk repositories, requirements specifications and system anomaly reports) at JPL. These lessons will be detailed in Section 6 after some background is presented. As a preview, we highlight our **top 3** lessons learned to provide context for the subsequent discussion on background and motivation for the use of TM:

- TM requires input of relevant information and is deficient at ignoring irrelevant information (Lesson 2).
- TM methods do not perform well with inconsistent and dynamically changing terminology. Do not assume static and consistent terminology in your historical artifacts (Lesson 3).
- Supporting information gathering and structuring for manual post-processing and interpretation is more important and practical than trying to completely automate (Lesson 8).

In addition to the lessons, we offer a number of suggestions and cautions in the use of TM on software artifacts (see Section 6). Although we make no claim of their generality, we hope these lessons learned and experiences gained from our TM applications in three areas of studies can be extended and will serve others in making effective use of TM and help them avoid some of the intensely effort wasting pitfalls that we encountered at JPL.

2. MOTIVATION

To date JPL has applied TM primarily in three main areas to: (1) the analysis of historical risks to develop a top-10 risk lists with expected risk scores; (2) the analysis of requirements to characterize and identify defects; and (3) to analyze the large anomaly repositories to determine how the frequencies and types of software anomalies are changing over time. The commonality of these efforts is that each one seeks to process a large number of informally structured artifacts to identify patterns and compile statistics. The output of the processing is useful only if it aids in addressing important development questions. For example, are the risks identified sufficiently comprehensive? Have any important risks been forgotten? How frequently has the same kind of anomaly been reported in the past? If a development organization is to improve, it must learn from its past and apply that knowledge to the effective management of current projects. With more experiences we expect more effective management and better project outcomes. However over time our experience base becomes increasingly fragmented and difficult to digest or apply.

The consequences of this challenge should not be understated. For example, such difficulties may have contributed to the demise of NASA's Mars Climate Orbiter (MCO) launched in December 1998. It was discovered that the developers of the Ground navigation software used English Units while the flight software developers used Metric Units. The discrepancy in units biased trajectory calculations in route and set MCO too close to Mars during its insertion into orbit where MCO went silent and was lost. The specific problem of incompatible units between system components that led to the MCO mishap was a well-known and documented risk on previous projects, yet the development teams still failed to identify it.

From the MCO mishap example we see that a particularly troublesome challenge has been providing confidence in the completeness of a risk analysis (i.e. the degree of certainty that all significant risks have been identified). There are two major problems here – (1) accounting for previously unknown risks, or the so-called unknown-unknown's, and (2) blindness or bias against recognizing known risks or risk patterns. Several techniques have been developed to deal with these problems. Generally these involve risk identification audit checklists and guidelines based on historical risk management experiences such as "top-10" risk lists, risk area taxonomies, and risk analysis processes. However, generating these aids can be cumbersome, costly, and difficult if possible to keep current. Furthermore, when manually generated, these aids too are subject to common risk assurance challenges such as completeness and tangibility.

Such challenges led us to the subject of this paper – to develop effective, automated TM tools to process large unstructured artifact repositories to extract useful project developed aids.

3. RELATED WORK

3.1 Text Mining in Software Engineering

1) LSA/LSI and K-Means clustering: Maletic and Marcus [33] first showed the usefulness of applying LSI in software reverse engineering. Poshyvanyk et al. [37] combined LSI and probabilistic ranking to improve the effectiveness of the feature location in source code. LSI was also applied to recover the traceability links between software documentation and program code [34], and other software engineering areas [35, 36]. Duan [38] compared the K-means with other clustering algorithms, and proposed a technology to automate the traceability process. For instance, LSA and k-means clustering are applied at JPL to identify similar risk patterns in the risk repository and the K-means algorithm to cluster similar risk patterns has different target and application domain with them.

2) Pattern/association rule mining: Michail shows CodeWeb [39] mines association rules from source code as framework reuse patterns. Li and Zhou propose PR-Miner [40], a tool using the closed frequent itemset mining technique to extract frequent program rules and detect the violations in C code. Livshits and Zimmermann show DynaMine [41], a tool that uses association rule mining to extract program rules from version histories for Java code and detect rule violations. Engler et al. [3] proposed a general approach for extracting program rules and detecting derivations by using the predefined rule templates from programmers. Their work focused on patterns mining in a specific programming language. At JPL, Risk Association Rules (RARs) are mined from risk analysis documents in historical projects sharing a specific domain in order to automate the risk reduction process, which addresses a different software engineering issue on software system risk assurance.

3) Data Mining and Machine Learning: Menzies et al. have used data mining/machine learning techniques for (1) predicting the frequency and severity of defect issues that will be observed during a software system's development, and (2) tuning Independent Verification and Validation (IV&V) efforts to software development tasks based on measurable attributes of those tasks. Accurate predictions can reduce the risk of not addressed significant issues during development, as well as the risk of expending effort on ineffective IV&V techniques. In [5], they applied existing data mining techniques to historical logs from NASA IV&V efforts to find selection and ranking rules for IV&V tasks. Due to large-scale ongoing organizational changes within NASA, data available to this study was potentially noisy; i.e., it contained signals not necessarily connected to the target of predicting issue severity and frequency. These studies reveal challenges in analyzing and mining historical project data and artifacts in large industrial organizations and thus have motivated our research addressed in this paper.

3.2 Software Systems Risk Management

Risks are ubiquitous to all system development projects. Boehm argues [2] that risk is a primary driver for project management. Many others agree. Carr [21] states "Risks are inherent in any software development activity. Furthermore, risk taking is essential to progress, and failure is often a key part of learning." We here summarize the risk-related research in two categories.

Software systems risk management has been an important research topic in software engineering for many years. Boehm [2] proposes a general risk management framework and five risk mitigation strategies: risk reduction, buy information, risk avoidance, risk transfer, and risk acceptance. Rashid and Kotonya [22] propose a checklist-based risk assessment technique for COTS-based software development. Carney et al. [4] present a questionnaire-based method for the same problem. Most of these risk analysis techniques and tools require a great deal of manual efforts and take subjective inputs from domain experts and/or system analysts. We have initially applied text mining techniques and tools at JPL aiming to automate software system risk identification, classification and assessment as a complementary to manual risk assurance techniques such as top-10 risk lists [2], 29-risk area taxonomy and checklists, chronic risk detection, and risk patterns, in order to improve the confidence of risk assurance.

3.3 Requirements Traceability Assurance

Although they are not text mining approaches in the strict sense of the term, the requirements tracing methods and tools developed by Hayes et al. for NASA are relevant to this discussion, since they are related to the overall problem of analyzing large volumes of natural-language text to provide assurance that they exhibit a set of specific quality attributes. Their results can reduce the risk that traces between related requirements have not been included in a trace matrix, thereby increasing confidence that the functionality and behavior of the implemented system will be as specified by the requirements. These techniques include keyword matching based SFEP for Software Automated Verification and Validation and Analysis System (SAVVAS) [6] and Information Retrieval (IR) based approach [7, 8]. The IR-based technique has been applied to a small project and has shown improvement over manual tracing. We are investigating its use on a larger effort to help automate the identification of ambiguous and inconsistent natural language requirements (see Section 5.1).

Specifically, as one of the text mining applications at JPL examines the satisfaction or completeness of non-functional requirements by functional requirements (each NFR minimally needs to map to at least one FR in order to be 'satisfied'), Holbrook, Hayes and Dekhtyar examined the use of Requirements Traceability Matrix (RTM) to assist with performing satisfaction assessment – determining if requirements were satisfied by design [6]. When such a technique is used to examine each NFR and see if it is satisfied by one or more FRs, it requires that each FR and NFR be chunked (parsed into phrases, basically) as well as tagged with parts of speech. Our text mining techniques do not require this additional pre-processing in assuring the completeness and correctness of RTM. Another unique aspect to our work is that of 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.

4. TEXT MINING TECHNIQUES AND TOOLS APPLIED AT JPL

4.1 Text Mining Techniques

We applied and experimented with a wide variety of TM techniques including Latent Semantic Analysis (LSA), K-means clustering and classification, (closed) frequent itemset mining, and parts-of-speech identification. In addition, techniques for pre-processing unstructured textual documents (such as removing "stopwords") are always needed to make effective use of TM.

1) Pre-processing artifacts for TM: Initially we need to collect a corpus of documents (e.g., risk analysis documents describing risks, risk reduction actions, and/or risk assessment information at various levels of details) from historical projects. Since documents may be kept in a variety of locations and formats, even within the same project, manual processing is generally required. After collecting and preparing the documents, we pre-processed them in order to construct the risk repository for TM. We performed tokenization (i.e., removing spurious upper case symbols, punctuation, and white space), pruning non-informational "stopwords" frequently used in all documents such as "the" and "a", and stemming (i.e., removing spurious suffixes. For example, after applying Porter's stemming algorithm [9], the terms "connecting, connection, connections" would all become "connect".) In other cases (e.g., anomaly report analysis in Section 5.2), we augmented the text with structural information in the form of parts-of-speech tags (see Section 4.2).

2) LSA/LSI: LSA (LSI) is a natural language processing technique for extracting and representing the contextual usage meaning of words by statistical computations applied to a large corpus of text. It uses a term-document matrix which describes the occurrences of terms in documents to analyze relationships (in particular in vectorial semantics) between a set of documents and the terms they contain by producing a set of concepts related to the documents and terms. We used Latent Semantic Analysis (LSA) to develop document similarity and dissimilarity measures for risks and requirements clustering because estimates produced by LSA depend on a deeper statistical analysis than contiguity and co-occurrence counts, LSA is often a much better predictor of semantic meaning-based judgments and performance.

3) K-Means clustering and classification: K-means clustering in an unsupervised (i.e. does not require a training data set) that aims to partition n observations into k clusters in which each observation belongs to the cluster with the nearest mean. We employed the iterative refinement approach to find natural clusters in the documents under investigation to use as candidate classifications or categories. Similar to this is K-means classification. This is a supervised process that takes a given set of classifications and examples documents with those classifications and applied the same approach as described above to determine the nearest classification for a given document. K-means can make use of a variety of "distance" metrics to determine nearness. We experimented with many different metrics but generally found the "Cosine" distance to have the best performance. This is also what is dictated for the LSA approach.

4) Frequent Itemset mining: Closed Frequent Itemset mining technique accepts the pre-processed mining repository $\boldsymbol{S}$ and a threshold δ, and returns the sub-itemsets that appear at least δ times in $\boldsymbol{S}$. An itemset I, is considered as frequent if its support $Support(I) \geq \delta$. We applied Closed Frequent Itemset mining in automated discovery of Risk Association Rules (RARs) (i.e., a risk with its effective reduction actions, or a risk with its child-risks) from historical risk repository analysis.

4.2 Text Mining Tools

We used widely available tools such as WEKA [10], RapidMiner [11], and R [12] to perform many of the TM tasks. These tools implement a wide variety of supervised and unsupervised data miners, offer numerous pre- and post-processing techniques, present the output in way that can easily be interpreted, and are sufficiently mature to ensure reliable operation when conducting long running experiments involving the application of many data miners on a large set of data. We found that the effort required to learn RapidMiner was less than that to learn other similar tools.

Because no one of these tools implement all of the techniques we needed for our TM efforts, one of the authors implemented their own Risk Association Rule (RAR) mining tool called *RARGen* [13] for mining risk repositories. In particular, we needed a well established approach for reliably automating the identification of associations of similar and dissimilar risk documents. Our experience with using *RARGen*, however, indicated that for most efforts, it is more effective to use multiple well established tools supplemented perhaps with small scripts or programs to perform the appropriate pre- or post-processing rather than writing a custom TM tool that provides the complete capabilities. We did this for our requirements classification and anomaly analysis efforts, in which we wished to augment the information in the text being mined with structural information about the text, in this case the parts of speech associated with each word in the text, to determine the effect on TM performance of the additional information. Before applying text miners, we applied a parts-of-speech (PoS) tagging tool, Trigrams'n Tags (TnT) [14] to the text we intended to mine, and then appended the PoS tag produced by TnT (e.g., NN, indicating a common noun) to each word in the text string. It is these modified strings from which we constructed the repository and to which we applied TM.

5. TM EXPERIENCES AT JPL

5.1 Requirements Analysis

We have been investigating TM at JPL to identify defective requirements more effectively. Previous work indicates that defective or misunderstood requirements are a significant source of anomalous behavior in fielded systems [15]; more detailed and accurate analyses of the requirements can reduce the risk of introducing requirements defects and propagating them into the implemented system. Since a typical space mission of the type developed at JPL may be specified by over 10,000 requirements, unaided manual analysis is not effective for assuring that specifications accurately reflect the system being developed.

Requirements Classification. Earlier work focused on identifying subsets of the requirements that could be analyzed separately, reducing the effort required by assurance engineers to perform the analysis as well as the number of analysis errors [16, 17]. We combined TM (e.g., word frequency analysis) and natural language processing (i.e., TnT parts-of-speech (PoS) taggers) to discriminate between temporal and non-temporal requirements. Using the complete set of over 7500 requirements for a currently operating robotic space mission, we developed learning models that perform well in distinguishing between temporal and non-temporal requirements [16]. We have also applied TM to distinguish between different types of temporal requirements, the goal being to automate the transformation of those requirements into a machine-readable notation that can be checked for consistency automatically. The learning models we developed performed well in many cases [17], but improvements, especially in the false positive rate, are still needed before they can be transitioned to operational use on a development effort. Our experiments indicated that information about the structure of the requirements (e.g., parts of speech for each word in the requirements text) led to better performing classifiers.

Ambiguity and Inconsistency Detection. We are also investigating TM and natural language processing for identifying ambiguous and inconsistent requirements. We have analyzed a subset of requirements for a currently operating mission to show that relatively simple learning and natural language processing techniques (e.g., parts of speech tagging) can differentiate between ambiguous and unambiguous natural-language requirements with a detection rate above 80% and a false positive rate of about 30% [18]. We are investigating more sophisticated learning techniques (e.g., boosting, bagging, voting) to improve the detection and false positive rates.

To automate the detection of temporal conflicts and inconsistencies in requirements, we are adapting tools developed by the University of Kentucky investigators [19, 20] for extracting constraints from natural language requirement documents and producing the results in a high-level constraint-representation language. The representations capture disjunctive temporal constraints ("completion of task A is followed by task B or task C"), often marked up with preferences ("completion of task A by the time t is preferred to the completion of task B in s time units").

Requirements Traceability Assurance. We have also conducted a pilot study using TM to assure software requirements traces [23]. 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 this study, we developed a practical method that applies well established TM and statistical methods (i.e., an integration of LSA and K-means with Empirical Maximum Likelihood estimates to determine the thresholds) to reduce the effort and to increase the confidence in assuring the traceability matrix. This method makes use of both requirements similarity (related to the likelihood that requirements trace to each other) and dissimilarity (or anti-trace, related to the likelihood that requirements do not trace to each other) to generate investigation sets that can significantly reduce the complexity of the traceability assurance task and help personnel focus on likely problem areas. In applying the technique to a set of requirements from the PROMISE data repository [24], we found a 58% reduction in the effort required to verify traces (compared to manual analysis), at the same time finding 120% more missing traces than the manual analysis and verifying nearly the same number of traces as the manual analysis (131 vs. 159). We also generated 154% more spurious traces (99 vs. 39) than manual analysis, which is wasteful of resources, but does not increase the risk of missing a trace critical to mission success. Our case study was small, but representative; we observed no inhibitors to scaling the method up to JPL sized projects. We will further investigate this approach on one or more JPL SQA efforts in the near future.

5.2 Analyzing Anomaly Reports

We have also used TM to analyze large volumes of anomaly reports to discover trends in the number and types of anomalies observed during testing and mission operations, and relationships between different types of anomalies. One of the simplest questions to ask is whether the frequencies and proportions of software anomalies observed during mission operations change from mission to mission. Although JPL has been tracking reported anomalies for many years, these questions have been surprisingly difficult to answer. Recent experience indicates that the quantitative or enumerated fields identifying characteristics such as the anomaly type (e.g., software, hardware, procedural errors) may contain substantial inaccuracies. For example, a detailed reading of the descriptive text (anomaly description, verification of anomaly, corrective actions) for a subset of the anomaly reports indicated that nearly all of the reports tagged as being software related did involve software defects, but many of the reports that were tagged as being something besides software turned out to be software related. In fact, the number of mislabeled anomaly reports was enough to double the count of software related anomalies [27]. By applying TM to the descriptive text in anomaly reports, we have been able to obtain more accurate counts of the number of software related anomaly reports [27]. The data on which we trained the learners was the result of another recent analysis of a subset of the anomaly data focused on identifying the types of anomalies observed during the cruise phase of missions to Mars [28]. The resulting learning models performed well in distinguishing between software and non-software related failures (a detection rate of 82% with a false positive rate of 20%). We are proceeding with the application of the learned models to the larger set of anomaly data. If the models perform well on this larger set of test data, we will then be able to establish a baseline against which we can quantitatively compare anomalous behavior observed during future missions.

5.3 Team X System Risk Assessment

Team X is JPL's concurrent engineering team that performs space mission systems feasibility and trade studies during the conceptual design phase as an input into the proposal decision-making process (see [25, 26] for more details). By Risk assessment we mean the identification and scoring of events that can lead to significant cost growth, schedule slip, or loss of science return including total mission failure. Historically the risk assessment process has been performed by ad-hoc "expert" driven methods. One of the known limitations of expert knowledge is the non-uniform quality of risks identified based upon the individual expert involved in the process. Individual experts have biases from their personal experience that can cause them to overlook or underestimate particular risks of project failure. The result is that there is very limited consistency across studies with respect to the risks identified, how they are described and how they have been scored. This has produced a risk repository where the risk descriptions are unstructured and expressed in natural language which can be redundant, ambiguous, and inconsistent. It also meant that the existing repository was difficult if not impossible to use during active studies.

To improve our confidence in the Team X risk identification process, some form of risk assurance needed to be performed, which required identifying and imposing a structure that could be used to reduce variation in language use and promote consistent risk identification. In particular, it is hoped that a structured risk repository will enable the generation of organization specific risk analysis tools such as a "top-10" project risk list or risk identification checklists.

But where does one obtain the risk source data? Project risks, which could provide 'real' risks, are scattered across documents located at numerous servers many of which have restricted access. They also have even greater variation in language that will be very specific to each project. Risks documented in the literature tend to be too generic. For example, Boehm generates a bi-annual industry top-10 risk list via a survey of industry affiliates of the Center for Systems and Software Engineering [1], which is quite helpful for providing top-level risk guidance. But they are overly general and lack specific association with an organizations particular risk considerations. For example, a risk always on this top-10 list is "Schedules, budgets, process" but how does this apply to JPL in particular? The schedule risk considerations for "missing the Mars launch window" are considerably different than "the product will not ship as announced." Hence, we decided to focus on the Team X repository even though it was a highly subjective predicted risk repository, as all the risks were in one database and could be searched. It was soon determined that manually categorizing risk data for a repository is cumbersome, expensive, and usually requires domain experts to generate useful meta-data and organizational categories. Indeed, at JPL our first attempt to create a semi-structured risk repository from 3361 unique risks from 170 Team X studies where the risk data was stored in a centralized risk management tool took 4 people working 4 to 6 hours per week over 3 months and never completed. The initial estimate was 1 to 2 weeks to ingest 170 projects.

In response to the above risk assurance issues, we are using TM techniques to assist with classifying risks in the Team X risk repository mentioned above. To date we have only engaged in mining the Team X Risk repository because it is more structured, but we intend to mine the distributed project repositories in the future. For our effort, the following key objectives define the value we hope to generate from TM methods:

1. Automate the continuous collection of historical project risks and their mitigations into a repository that enables economical generation and maintenance of risk assurance tools such as top-10 risk lists, 29-risk area taxonomy, chronic risk detection, risk patterns, and risk area taxonomies and checklists.
2. Improve confidence in auditing of risk documents, especially with inexperienced project personnel (i.e. unbiased and comprehensive, reduce variability of results, identify potential gaps and omissions, reduce redundancies, etc.)

3. Increase the cost-effectiveness of risk assurance (reduce cost and time to perform, increase utility of results, etc.)
4. Support targeted risk assurance by automating (i.e. predicting) the identification of historically high-priority risks relevant for a given (new) project.

It is important to note that the above objectives are in support of assuring more accurate risk identification but not to replace human risk identification and assessment activities. We do not believe the complete replacement of human-based risk identification is practical, prudent, or even possible at this time given the quality of available data and the immaturity of TM technology. TM is currently being used to generate clusters and classification of the risk descriptions from the Team X risk repository. The number of risks within a given cluster can provide a sample estimate for frequency of occurrence of risks with a given classification. This and associating classifications with project descriptions enable prediction of relevant historical risks for a given (new) project.

It required several attempts at applying different TM methods to obtain a 'successful' solution. All approaches required significant pre-processing of the data (some details in the lessons learned section below). The specific TM methods and tools used are described in Section 4. Our first attempt used unsupervised clustering methods to observe what risk groupings might be observed in the current repository. The clusters generated using the unsupervised methods were for the most part un-interpretable risk categories. While we did not actually measure the success rate it appeared to be around 5%. To address this, we next performed supervised clustering by supplying groups of key terms based on the 12 risk checklists generated by the Team X subsystem chairs (generated as part of a parallel activity) as the training set for the clustering. This attempt enabled us to observe how well our human generated risk categories are able to describe risks in our repository. Then these results were combined with classification based on the unsupervised clusters to generate a hybrid training set to generate new clusters that became more easily interpreted. We had to experiment with different variations of pre-processing the data based on different combinations of the levels in the hierarchy of the risk categories. Finally we apply classification with the project descriptions and to associate groups of project key terms with the clusters. Over the 12 subsystems the hybrid approach yielded from 35% to 75% correct categorization with an approximate average of 50% and risks were only binned in the wrong subsystem less than 1% of the time. This may appear to be mediocre results. However, it greatly reduced the amount of time required to manually categorize the risks as verifying a correct classification took seconds allowing us to focus on the incorrect classifications. Even with the misclassified risks the TM results enabled a simpler mental evaluation and saved the time for manual learning. The final version took an average of one day per subsystem to successfully complete the classifications. This also yielded numerous specific risk examples for each category that can be used in training new Team X members and providing guidance during sessions.

We have learned a great deal about the nature of our risk data, the effort required to produce useable results from TM these data, and the importance of establishing long-term value-driven objectives. Indeed, the objectives listed above are the a major refinement over our initial "short term" objectives which essentially assumed TM would magically produce useable clusters with little data preparation. How the data preparation effort far exceeded our initial expectations and other lessons learned are discussed in the next section.

6. LESSONS LEARNED

We have accumulated 11 lessons learned from our TM experiences discussed in Section 5. Note that the lessons were significant to us and we make *no claims of their relevance or importance in general*. However, we also note that the TM literature does not adequately address the lessons we discuss below. It appears that it is assumed that the problems described are easy or inconsequential to resolve. This *has not been* our experience. They are not trivial or as straightforward as they may appear and can drastically affect the success of a TM effort. The lessons learned with suggestions and cautions gained from our TM applications in three areas at JPL will benefit practitioners seeking TM aids for similar problems, which we hope could also be extended to other application domains if possible.

A few of these lessons may appear obvious, yet we note that we as experienced developers and researchers did not recognize them immediately. Other lessons are subtler and also perhaps more specialized to our particular environment at JPL. In either case, we share these lessons openly with the hope they will be of benefit for other organizations seeking to effectively apply TM. Collectively they summarize our experiences in applying TM. Our focus on examples from our TM experiences for the Team X risk repository (see Section 5.3) is not meant to imply that the lessons were not drawn for other experiences. Rather, the Team X risk TM is our most recent and ambitious effort. Also the ordering of the lessons below is not meant to imply any ranking.

Lesson 1: ***Preprocessing TM data will take much longer than you expect. The proper choice of the stopword list, stemming mechanism and an indexing mechanism is crucial. And beware of data conversion and formatting overhead.***

When performing TM, there are a number of time consuming pre-processing tasks that can add substantially to your effort. These include removing replicates, tuning stemming, editing stopwords, and removing "generic" terms. For this latter effort, we found in our requirements tracing TM effort that some terms were used frequently specifically for requirements that were not common stopwords. The tf*idf weighting scheme was inadequate in discounting such words because they were not always used in each requirement but they contribute zero information. As an example, we noticed that the words "system" and "product" were always implicit in functional requirements and but not explicitly used in the requirement description. Such terms led the TM to mistakenly assign a higher similarity to requirements that used these terms over those that did not even though the requirements were not related more. There are no TM methods that we are aware of that can automatically detect and remove such generic terms. Our approach was to generate the most frequently occurring terms (the 80th percentile) after removing stopwords and manually review these for informational significance.

In addition, data sources take many different forms and formats. In our experience with Team X, risks were documented in PowerPoint files, PDF files, flat file databases, and Excel spreadsheets. We could not simply point our TM tools to a directory of files and expect results. The files needed to be converted into a useable format (e.g. TXT, ARFF) or conversion tools need to be found or developed (e.g. XLS to ARFF). Even though some automated tools (e.g., Google search) can help with the conversion, there is still a significant amount of tool searching, assessment, configuration, installation overheads and learning curve. And the tool set-up effort is usually not completely reusable for a different project due to the variations of source data formats.

Lesson 2: ***TM requires input of relevant information and is deficient at ignoring irrelevant information.***

Perhaps due to the success of general data mining of unstructured data, the TM literature seems to imply that it too is able to pull meaningful knowledge from a mash-up of information. This seems to assume that the documents do not contribute overly much "noise" or irrelevant information. This is not generally a reasonable assumption. For example, in addition to risk description, the risks in our risk repository contain information such as project description, risk mitigation, risk assessment, risk factors, creation dates, and mitigation dates. By simply loading all this information into the TM and performed LSA for clustering, we did not get useful results. We only obtained meaningful results after eliminating all but the risk descriptions.

Caution 1: *It takes significant manual effort to prepare relevant information from inconsistent and unstructured data to a state useable for reliable and meaningful TM results.*

Often inconsistent specification in unstructured documents makes it time consuming to locate relevant information. This is even the case when documents are gathered from a structured artifact collection tool. For example, many Team X projects made use of the Risk Assessment Process (RAP) tool to describe risks and collect assessment data from multiple stakeholders in a distributed and structured manner. In our dump of the RAP database we often found that the "risk title" contained the "risk description" leaving the field for the latter empty.

Suggestion 1: *Consider migrating data into a structured database.*

In our Team X risk repository TM effort, we found it useful to import the risk repository data into a relational database. Flat files and spreadsheets were too slow and cumbersome to work with. For example, among the 9955 in Team X risks reported in the RAP database, 7182 (72%) of these were essentially duplicates because the RAP tool stores risks by each risk reporter, not by a unique risk ID. If multiple stakeholders provided risk assessments or any risk metadata there would be individual risk entries for these. The K-means clustering we want to apply creates overly narrow similarity classifications in the presence of duplicates. Because duplicate risks differed in only a few ways (e.g. risk assessment values), it was straightforward using the database to remove records that had duplicate risks and project descriptions but may have differed elsewhere. In addition, we were more easily able to locate missing data and remove irrelevant data (usually by removing table attributes). Preprocessing documents (e.g. stopword removal, stemming) in a database is far more efficient than in tools such as WEKA and RapidMiner for large data sets. Furthermore, preprocessing can be specialized to each attribute (e.g. risk description ontology or thesaurus) that is more targeted and easier to maintain over global processing. Perhaps the most valuable is its ability to easily create "flattened" data by combining multiple attributes. For example a risk document to be classified was combined with the name of the subsystem for that risk. As per Lesson 2, determining what information is relevant for a particular learner is essential for obtaining useful results. Although such "flattening" can be done with many of the TM tools we used, but it is not as efficient or flexible as a database for large data sets as in our case. In addition to increased flexibility, by preprocessing the data in a database we estimate we were able to cut 30%-60% of the TM processing time. On our Team X risk database this was a difference between 80 minutes and 4 hours waiting for clustering results.

Another benefit that should not be underappreciated is applying the TM results. For us the TM results are used to generate assurance tools such as top-10 risk lists. Such tools are easily generated by adding a Risk_Cluster table to the database. This further enables useful enhancements such as generating a list of relevant risk mitigation associated with a given cluster.

Lesson 3: ***TM methods do not perform well with inconsistent and dynamically changing terminology. Do not assume static and consistent terminology in your historical artifacts.***

At JPL documents use terminology that varies over different groups, over time, and over different projects. This makes it difficult to generate knowledge based on terms, and we have no terminology "maps" or organization thesauruses (though we do have a comprehensive dictionary). We have found in the Team X projects that different terms are sometimes used to refer to the same thing (e.g. early on a "concern" was the same as what we now call a "risk"). This is annoying, but not impossible to deal with. What is very challenging is that many projects use the same term to refer to completely different things (e.g. heritage sometimes means reuse and sometimes inheritance). This overloading and inconsistent use of terms significantly reduces TM results' usefulness.

Suggestion 2: *Consider using an organization specific thesaurus.*

The thesaurus can be directly integrated into the TM process during pre-processing to eliminate inconstant and duplicate terminology. It is relatively straightforward to create a thesaurus from the term-frequency matrix generated by a TM. It is important to note that maintaining the thesaurus must be an on-going, albeit low level, task if it is to remain relevant to an organization's development efforts.

Lesson 4: ***Metadata is vital but often neglected and hard to get.***

The metadata added to documents can play an essential role in obtaining useful results from classification and clustering. What the TM literature downplays is that this information cannot easily itself be obtained via TM. We have found there is a high manual effort for obtaining and specifying metadata due to a number of common complications. Firstly, metadata information is often implicitly located in a totally different document, location, or not documented at all. For our Team X risk TM, some metadata about a project is maintained in the folder hierarchy in which the project documents are stored. E.g. A→B→C means the file C is an A.B project. Sometimes the metadata is maintained in someone's head, increasing the difficulty of obtaining it, and certainly not automatically. Secondly, there's a lot of metadata to consider. What is actually relevant? What can be accessed? We thought we could get all the metadata manually for the Team X risk TM effort, but after 4 people spent 1 day trying to pull information from the repository directory structure and PowerPoint files and paste it into a spreadsheet, we realized it would take many weeks and was totally impractical. We had to explore alternatives such as methods that can perform association without metadata, incremental addition of metadata as needed, or trying to TM the metadata.

Lesson 5: ***Do not expect "quick win" results from TM. You will be disappointed. But look for long-term benefits.***

Our initial objective for TM Team X risks was simple enough - avoid the huge manual and error prone effort of collecting and processing risk information from our repositories. This objective was short sighted because the effort to establish useful TM is very high too (e.g., preparing the data, pre-processing, interpreting). Contrary to what some TM literature implies about what manual effort it may avoid (which it doesn't), or instantly and automatically obtaining useful knowledge from large information repositories, we discovered that there is more value in what TM enables longitudinally. As an example from our Team X risk TM effort, TM can enable cost-effective continual and automatic updating of "most frequent Team X risks." This gives us a snapshot of the kinds of risks that are most relevant currently which Boehm has shown vary considerably over time [2]. It also enables us to run scenarios and investigate patterns, such as "which risks seem to persist over time" (maybe we do not address these sufficiently), or "how has the risk profile for projects at JPL changed over the years?" (i.e., risk evolution trend)

Suggestion 3: *Have very clear long-term objectives for what you are trying to achieve with TM and set acceptance quality criteria.*

The impression of TM is that it is a short-term, quick, easy to apply tool, but we found otherwise. We had to look for longer term benefit and accept that it requires substantial investment. In our Team X effort we eventually settled on the four objectives indicated in Section 5.3 after considering long-term value not just short term payoff. We found that "save effort over manual through automation" is unlikely to pay off in the short term due to the many effort-consuming development and preprocessing tasks. However, higher assurance, continual low-effort update, insight, "getting house in order", improved process, etc. can have long term high-payoffs. Avoid demanding overly accurate or precise results by TM large collections of unstructured system development artifacts (you won't get this anyway leveraging existing TM techniques and tools) based on the Team X statistics presented in Section 5.3.

Lesson 6: ***Start with modest objectives and pilot your TM techniques and tools on small data sets before jumping in to a full analysis.***

This is especially important if you are inexperienced with TM. The above lessons all indicate that there is substantial data preparation and analysis effort needed to obtain useful TM results. By piloting on a small data set with modest objectives you can more quickly get a handle on what this effort will be, what will be feasible to perform, and what quality of results to expect before investing heavily in a full TM effort.

Lesson 7: ***Avoid developing your own TM methods and tools.***

It is always tempting to develop your own methods and tools. TM is deceptively simple looking. We made the mistake of trying to "roll our own" by developing our own risk association rule mining tool. This was extremely time consuming and error prone. In the end, we were able to obtain more reliable results with small modifications to the established tool packages WEKA, R, and RapidMiner. Develop tools only when existing tools are inadequate (e.g. scale or integration issues).

Suggestion 4: *Try simple, well established methods first.*

Avoid the temptation to use non-established methods. There is a lot of literature about using well-established methods such as LSA and what results can be expected. We have found that using more sophisticated but immature methods did not provide substantially better results and took a lot of effort to understand and get working properly.

Lesson 8: ***Supporting information gathering and structuring for manual post-processing and interpretation is more important and practical than trying to completely automate.***

This lesson cannot be too strongly emphasized. The current state of TM does not provide confidence in fully automating knowledge acquisition or complete replacement of expert judgment and interpretation. What it can do very well is complement human-based methods by performing tasks that are difficult or error prone for humans with large data sets. For example, at Team X the TM risk clusters enable risk assurance personal to reduce the possibility that an important historical risk was omitted. Our risk experts at times can become desensitized to common risks and may "take it as given" that a project has such a risk without identifying it explicitly. The problem is that this risk may be "forgotten" and never be addressed. It is difficult for our risk experts to recall all historical risks, and even more difficult for risk assurance personnel to traverse through the risk repository to look for potentially missing risks. However TM is quite good at being exhaustive for such tasks.

Lesson 9: ***Do not underestimate the TM methods and tools learning curve.***

You need a very thorough understanding of the particular TM methods and tools you use before getting good and reliable results. As discussed above, we have found that results are highly sensitive to the information provided, data pre-processing, and selection from many complex TM methods and configurations (e.g. what clustering method such as K-means or which distance metric such as Cosine). It is imperative to know clearly what is required to get the desired results and how to perform the tasks needed to get them. This takes quite a bit of time to research, study, and experiment via trials and mistakes before converging on a suitable approach and getting it to work. Our team consists of staff with PhD's in the roles of senior researchers, developers and managers. Even with such experienced staff it took several months before we were able to set up a viable K-means classification system for the Team X risk database.

Lesson 10: ***Do not be hasty to delete very infrequent or unique terms.***

Typically there will be a large number of unique terms in the occurrence matrix. It is inadvisable to blindly delete these (or automatically prune). Especially for relatively small data sets you expect to grow, many of these terms will become keywords and provide important classification information later (remember the long term objectives). For example, in our Team X risk TM we found the term "Aero-Capture" occurs in only a few risks (less than 1%). However we know that many projects under consideration and in the near future will involve aerocapture and so this term will be a significant risk identification term.

Lesson 11: ***Supervised or unsupervised learning? Both!***

Our experiences indicate that unsupervised TM rarely can produce useful and interpretable results. Some guidance in the training for clustering or classification can go a long way to addressing this. Obtaining a training set in which you have high confidence can be a challenge. Start with an unsupervised method then look for patterns and indicators. Use these as a basis for generating training sets which are then used for a supervised method. This can be iterated several times until satisfactory results are achieved. The hybrid clustering approach iteratively produced desired risk categorization outcome on 12 Team X subsystems (see Section 5.3).

7. RECOMMENDATIONS AND CONCLUSIONS

We have seen in the preceding sections that TM can be profitably applied to a number of problems in assuring different artifacts of large, complex software systems. As mentioned in Section 6, there are a number of issues to be taken into consideration when setting up a TM program to analyze large volumes of unstructured data. The most important of these, based on our experience are highlighted as follows.

Do not expect that the benefits of TM will be immediately apparent. Before the benefits of TM can be realized, it is important to *conduct exploratory analyses of the data to be mined* to 1) identify the questions that can be answered by applying TM and 2) for those questions that cannot be answered, identify additional data that must be collected as well as data for which the noise is too large to yield useful results. We also emphasize the importance of *setting measurable goals* for the TM program: just as a metrics program should be set up to achieve a set of measurable goals (e.g., the Goal-Question-Metric paradigm [29, 30]) that can be satisfied by taking measurements to answer specific questions, our experience indicates that a TM program must be designed to achieve a set of measurable goals. Because the amount of effort involved in setting up the program and the impact to development staff may be greater than that involved in setting up a metrics program, it is even more important to define appropriate, measurable goals to 1) minimize disruption to the development and assurance staff, and 2) to maximize the likelihood that the results will benefit the organization.

It is also important to note that in many cases, *TM cannot be applied to unstructured information without preprocessing* the data. Large organizations such as JPL that simultaneously develop numerous systems to support multiple missions may organize the development artifacts to be analyzed across several different systems in a number of different formats. For example, high-level requirements specifying the overall mission goals may be contained in a set of natural language documents managed by a system such as DocuShare [31]. Lower-level system and subsystem requirements may be managed by a tool such as DOORS [32], and the lowest level of requirements (e.g., the requirements for the on-board control software) may be managed in a set of spreadsheets that are again managed in a repository. Architectural and detailed design artifacts (e.g., state charts, collaboration diagrams) may also be included in the analysis. Finally, milestone and technical review records may also be analyzed – these are usually formatted as natural language documents or slide presentations mixing text and graphics. For each type of artifact, preprocessing to extract the relevant information from the artifacts being analyzed and transform it into a form that can be input to the text miner(s) is likely to be needed.

To obtain the full benefit of TM, *analysts need to be sufficiently knowledgeable in the operation of the text miners* in order to provide appropriate input to them and interpret the results. At least some analysts will need sufficient training in experimental design and statistical analysis of experimental data to conduct the pilot studies that precede the use of TM in a production environment. Currently, this is not part of the skill sets of many of the assurance engineers with whom we have worked, meaning that some additional, non-trivial training for the assurance staff will be required before these techniques can be deployed.

Finally, our experience indicates that TM should not be instituted with the *goal of fully automating* particular types of analyses. In none of our efforts of the past several years have the text miners performed sufficiently well to supplant a human analyst's insight. What we have observed repeatedly, however, is that text miners can provide useful assistance by 1) reducing the time and effort required by the analyst to accomplish a specific goal (e.g., verifying a trace matrix, identifying ambiguous requirements), or 2) discovering previously unseen patterns in the data that a human analyst can then examine in detail to determine their origins and their consequences for the development effort(s) being supported.

8. ACKNOWLEDGMENTS

Majority of the work described in this paper was carried out at the Jet Propulsion Laboratory, under a contract with the National Aeronautics and Space Administration, and was sponsored by the National Aeronautics and Space Administration's Office of Safety and Mission Assurance Software Assurance Research Program (SARP). SARP tasks are managed locally by JPL's Assurance Technology Program Office. Thanks to Joel Wilf, supervisor of the JPL Software Quality Assurance group for his support and enthusiasm of this work. Special thanks to Prof. Jamie Andrews at the University of Western Ontario, Canada for his valuable suggestions in preparing this paper.

9. REFERENCES

[1] N. Leveson, Safeware System Safety and Computers, Addison-Wesley, 1995.

[2] B. W. Boehm, "Software risk management: Principles and practice". IEEE Software, 8(1): 32-41.

[3] D. Engler, D. Y. Chen, S. Hallem, A. Chou, and B. Chelf, "Bugs as deviant behavior: a general approach to inferring errors in systems code", In Proc. SOSP, pages 57–72, 2001.

[4] D. Carney, E. Morris, and P. Place, "Identifying Commercial Off-the-Shelf (COTS) Product Risks: The COTS Usage Risk Evaluation", CMU/SEI- 2003-TR-023. September 2003.

[5] Menzies, T., Benson, M.,Costello, K., Moats, C., Northey, M., Richardson, J., "Learning Better IV&V Practices," Innovations in Systems and Software Engineering, Springer London, 4(2), June 2008, 169-183.

[6] J. H. Hayes, "Risk Reduction Through Requirements Tracing," Proc. of 1990 Software Quality Week, San Francisco, CA, 1990.

[7] J. H. Hayes, A. Dekhtyar, J. Osbourne, "Improving Requirements Tracing via Information Retrieval," Proc. of 2003 IEEE International Conference on Requirements Engineering, IEEE Press, Sep. 2003, pp. 151-161.

[8] J. H. Hayes, A. Dekhtyar, S. Sundaram, S. Howard, "Helping Analysts Trace Requirements: An Objective Look," in Proc. of IEEE International Conference on Requirements Engineering, Sep. 2004, pp. 249-261.

[9] The Porter Stemming algorithm, Karen Sparck Jones and Peter Willet, Readings in Information Retrieval, San Francisco: Morgan Kaufmann, 1997. ISBN 1-55860-454-4.

[10] WEKA 3 – Data Mining with Open Source Machine Learning Software in Java, http://www.cs.waikato.ac.nz/ml/weka/.

[11] Rapid-I, “RapidMiner”, http://rapid-i.com/content/view/181/190/, accessed Feb. 8, 2011.

[12] The R Project for Statistic Computing, http://www.r-project.org/, accessed Feb. 8, 2011.

[13] L. Huang, D. Port, L. Wang, T. Xie, and T. Menzies, “Text Mining in Supporting Software Systems Risk Assurance”, Proceedings of 25th IEEE/ACM International Conference on Automated Software Engineering (ASE), Sept. 20-24, 2010.

[14] TnT -- Statistical Part-of-Speech Tagging, http://www.coli.uni-saarland.de/~thorsten/tnt/.

[15] Lutz, R., Mikulski, C., “Empirical Analysis of Safety-Critical Anomalies During Operations,” IEEE Trans. on Software Engineering, vol. 30, no. 3, Mar, 2004, 172-180.

[16] A. Nikora, "Classifying Requirements: Towards a More Rigorous Analysis of Natural-Language Specifications." Proc. of the 16th International Symposium on Software Reliability Engineering. Chicago, 2005. 291-300.

[17] A. Nikora, and G. Balcom. "Automated Identification of LTL Patterns in Natural Language Requirements", Proc. 20th International Symposium on Software Reliability Engineering. IEEE Computer Society, 2009. 185-194.

[18] Nikora, A., Hayes, J., and Holbrook, E. “Experiments in Automated Identification of Ambiguous Natural-Language Requirements,” to appear, proceedings 21st IEEE International Symposium on Software Reliability Engineering, San Jose: IEEE Computer Society, 2010.

[19] D. East, and M. Truszczynski. "Predicate-calculus based logics for modeling and solving search problems." ACM Transactions on Computational Logic 7(1) (2006): 38-83.5.

[20] D. East, and M. Truszczynski. "The aspps System." Proceedings of the 8th European Conference on Logics in Artificial Intelligence (JELIA), Lecture Notes in Computer Science. Springer Verlag, 2002. 533-536.

[21] M. Carr, S. Konda, I. Monarch, C. Ulrich, C. Walker, “Taxonomy based risk identification (CMU/SEI-93-TR-6, ADA266992)”, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA, 1993.

[22] A. Rashid, and G. Kotonya: “Risk Management in Component-Based Development: A Separation of Concerns Perspective,” ECOOP Workshop on Advanced Separation of Concerns, Springer-Verlag, 2001.

[23] Port, D., Nikora, A., Hayes, J., and Huang, L., “Text Mining Support for Software Requirements: Traceability Assurance”, Proc. HICSS 2011.

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

[25] Warfield, K. and Hihn J., “Spreadsheets in Team X: Preserving Order in an Inherently Chaotic Environment”, Proc. of the 42nd Hawaiian International Conference on System Sciences (HICSS 42), Waikoloa, HI, Jan. 5-8, 2009

[26] Hihn, J., Chattopadhyay, D., Hanna, R., and Port, D., “Identification And Classification Of Common Risks In Space Science Missions”, Proc. AIAA Space 2010 Conference and Exposition, Anaheim, CA. Sept. 1-3, 2010.

[27] Nikora, A., G. Balcom. "Improving the Accuracy of Space Mission Software Anomaly Frequency Estimates." Proc. of the 3rd IEEE International Conference on Space Mission Challenges for Information Technology. Pasadena, CA, 2009. 402-409.

[28] Green, N., A. Hoffman, T. Schow, and H. Garrett. "Anomaly trends for robotic missions to Mars: implications for mission reliability." 44th AIAA Aerospace Sciences Meeting and Exhibit. Reno, NV, 2006.

[29] Basili, V. R., and Weiss, D. M., “A Methodology for Collecting Valid Software Engineering Data”, IEEE Trans. on Software Engineering, SE-lo, 728-738, November 1984.

[30] Basili, V. R., “Software Modeling and Measurement: The Goal/Question/Metric Paradigm”, University of Mary- land Technical Report. UMIACS-TR-92-96, 1992. http://portal.acm.org/citation.cfm?id=137076.

[31] DocuShare Document Management Solutions, http://www.office.xerox.com/software-solutions/xerox-docushare/enus.html, accessed Mar 15, 2010.

[32] IBM Software – Rational DOORS, http://www-01.ibm.com/software/awdtools/doors/productline/.

[33] J. I. Maletic, A. Marcus, “Using latent semantic analysis to identify similarities in source code to support program understanding”, In Proc. 12th IEEE International Conference on Tools with Artificial Intelligence (ICTAI’00), p 46, 2000.

[34] A. DeLucia, F. Fasano, R. Oliveto, G. Tortora, “Can Information Retrieval Techniques Effectively support Traceability Link Recovery?”, Proc. 14th IEEE Int. Conf. on Program Comprehension, Athens, Greece, 2006, pp.307-316.

[35] J. H. Hayes, A. Dekhtyar, and S. K. Sundaram, “Advancing candidate link generation for requirements tracing: the study of methods”, IEEE Trans. on Software Engineering, vol.32,no.1, pp 4-19, January 2006.

[36] M. Lormans and A. Van Deursen, Can LSI help Reconstructing Requirements Traceability in Design and Test? Proc. 10th IEEE European Conf. on Software Maintenance and Reengineering, 2006, pp. 47-56.

[37] D. Poshyvanyk, Y.-G. Gueheneuc, A. Marcus, G. Antoniol, and V. Rajlich, “Combining Probabilistic Ranking and Latenet Semantic Indexing for Feature Identification”, In Proc. 14th IEEE Int. Conf. on Program Comprehension, pp. 137-148, Athens, Greece, 2006.

[38] C. Duan, and J. Cleland-Huang, “Clustering support for automated tracing”, In Proceedings of the 22 IEEE/ACM international Conference on Automated Software Engineering, Page 244-253, Atlanta, Georgia, USA, 2007.

[39] A. Michail, “Data mining library reuse patterns using generalized association rules”, ICSE, pp.167–176, 2000.

[40] Z. Li and Y. Zhou, “PR-Miner: Automatically extracting implicit programming rules and detecting violations in large software codes”, In Proc. ESEC/FSE, pages 306–315, 2005.

[41] V. B. Livshits and T. Zimmermann, “DynaMine: Finding common error patterns by mining software revision histories”, In Proc. ESEC/FSE, pages 296–305, 2005.