Text Mining in Supporting Software Systems Risk Assurance

LiGuo Huang[1], Daniel Port[2], Liang Wang[1], Tao Xie[3], Tim Menzies[4]
[1] Southern Methodist University, Dallas, TX; [2] Jet Propulsion Laboratory, California Institute of Technology, CA;
[3] North Carolina State University, Raleigh, NC; [4] West Virginia University, Morgantown, WV; USA
{lghuang, liangw}@smu.edu, Dan.Port@jpl.nasa.gov, xie@csc.ncsu.edu, tim@menzies.us

ABSTRACT
Insufficient risk analysis often leads to software system design defects and system failures. Assurance of software risk documents aims to increase the confidence that identified risks are complete, specific, and correct. Yet assurance methods rely heavily on manual analysis that requires significant knowledge of historical projects and subjective, perhaps biased judgment from domain experts. To address the issue, we have developed RARGen, a text mining-based approach based on well-established methods aiming to automatically create and maintain risk repositories to identify usable risk association rules (RARs) from a corpus of risk analysis documents. RARs are risks that have frequently occurred in historical projects. We evaluate RARGen on 20 publicly available e-service projects. Our evaluation results show that RARGen can effectively reason about RARs, increase confidence and cost-effectiveness of risk assurance, and support difficult-to-perform activities such as assuring complete-risk identification.

Categories and Subject Descriptors
D.2.9 [**Management**]: Software quality assurance (SQA).

Keywords
Risk assurance; risk reduction; text mining; mining software repositories; association rule; latent semantic analysis.

1. INTRODUCTION
Motivation. Leveson [2] remarks that, in modern complex systems, unsafe operations often result from insufficient risk analysis. A risk is defined as a combination of the likelihood of an accident and the severity of the potential consequences. More generally, Boehm [1] defines software risk as a potential for the development or product to have an unsatisfactory outcome to project stakeholders. Unsatisfactory software project and system outcomes (e.g., problems and failures) due to insufficient risk analysis have been extensively reported and documented in the literature [1, 2, 8, 9]. As a result, there is an increasing interest in independent review of risk assessment (also called *risk assurance*) for systems with a high criticality and risk.

Contributions. In this paper, we present the Risk Association Rule Generation (RARGen) approach and associated tool to tackle the challenges of supporting software systems risk assurance. The following *4 key objectives* define the value that we hope to generate from RARGen, and also set as our evaluation criteria in Section 5.

Objective 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, chronic risk detection, risk patterns, and risk-area taxonomies and identification checklists.

Objective 2. **Improve confidence in review of risk documents**, especially with inexperienced project personnel (i.e., offer unbiased and comprehensive identification, reduce variability of results, identify potential gaps and omissions, reduce redundancies, etc.).

Objective 3. **Increase the cost-effectiveness of risk assurance** (i.e., reduce cost and effort, increase utility of results, etc.)

Objective 4. **Support the refinement of risk mitigations** by offering comprehensive historically significant mitigation options.

Our research in this paper is guided by the following questions:
RQ1 Is it possible to reason about **R**isk **A**ssociation **R**ules (RARs) from unstructured Software/System Engineering (SE) artifacts?

RQ2 Can we increase human confidence and improve the cost-effectiveness in review risk documents via text mining?

RQ3 Can the mined RARs support risk assurance activities such as identifying incomplete risk identification or reduction actions?

We investigate these questions by applying RARGen within various organizations such as the Jet Propulsion Laboratory (JPL) and the University of Southern California's Center for Systems and Software Engineering (USC-CSSE). Section 4 presents a feasibility evaluation using 20 publicly accessible USC-CSSE real-client e-service projects. In this evaluation, we found ample support that RARGen is able to meet Objectives 1-4.

2. BACKGROUND: Risk Assurance
Insufficient system risk analysis includes errors such as failure to identify a significant risk (omission), incorrect risk specification, redundant risks, vague or poorly specified risks, and risks without mitigation options. Such errors may have led 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 using the required 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 presumed 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, but the development teams still failed to identify it.

MCO is one of many examples of insufficient risk analysis. Given the rapidly increasing costs and consequences of software errors, understandably there is increasing interest in ensuring that

ASE'10, September 20-24, 2010, Antwerp, Belgium.

sufficient risk analysis is performed. Organizations such as JPL and the USC-CSSE have started employing risk assurance practices. **Risk assurance** is the use of quality assessment techniques such as verification and validation (V&V) to ensure that sufficient risk analysis has been performed, i.e., the risk analysis has been correct, complete, clear, and actionable as much as possible.

A primary activity of risk assurance is the review of risk documentation. This activity involves reading through documents by assurance personnel who "manually" scan for risk analysis errors.

Risk Assurance Challenges. The authors have a great deal of experience in risk assurance research and practice, ranging from educating undergraduate/graduate computer science and business students in software risk management at various institutions, to working with assurance professionals from NASA's IV&V facility and JPL's Software Quality Assurance and Software Quality Improvement groups. From this experience, we have observed a number of challenges in performing risk assurance:

- Risk assurance is costly relative to the results that it generates; it is often considered a "non-critical path" activity and frequently the first target for budget cuts or schedule slips.
- Auditors not involved in the development need significant time to become familiar enough with the details of the project (requirements, design, etc.) to creditably understand and assess risk documents.
- There is low confidence in the completeness of auditor's results due to the unknown-unknown's and blindness/bias problems. It is difficult for auditors to recognize or suggest appropriate risk mitigation details for a given project.
- Risk descriptions are generally written in natural language, which can be redundant, ambiguous, and inconsistent across documents, document versions, and different documenters.
- Risks are often stated with such generality or vagueness as to render detecting their explicit connections to design choices and options impossible.

Many organizations, including USC-CSSE and JPL have been collecting historical project risk data into risk repositories in part as a response to the preceding risk assurance issues, and also in part to provide a resource for improving risk analysis in general (e.g., a lessons-learned database). The use of a risk repository also has its challenges. Generally, we have observed that project personnel rarely make direct use of repositories. It is difficult and time consuming to dig out information that is relevant to a particular project. Even given relevant data, it is easy to get overwhelmed with the mass of details.

3. RARGEN APPROACH

RARGen uses text mining to automate the collection of risks and their associated mitigations in risk analysis documents from historical projects into a repository. This repository is useful for supporting risk assurance activities such as generation of risk lists and system-risk completeness analysis (meeting our Objectives 1 and 4 and in part, 2). One of the major goals of RARGen is to require little human effort or input from domain experts while still providing useful results. If achieved, this goal is one way of meeting our Objective 3 on cost-effectiveness, and again to some extent of Objective 2.

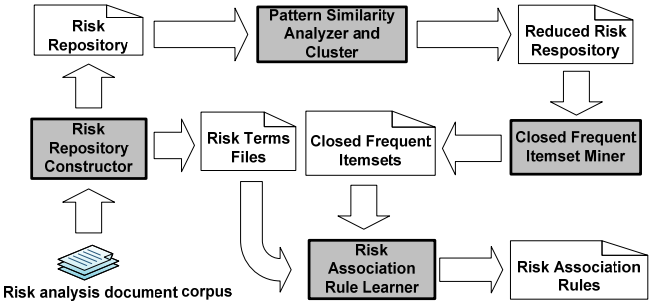

Figure 1. Overview of Risk Association Rules Generation (RARGen) Approach

Figure 1 shows an overview of our RARGen approach. There are four major components (shaded boxes in Figure 1): **(1) risk repository constructor, (2) pattern similarity analyzer and cluster, (3) closed frequent itemset miner, and (4) risk association rule learner**. Note that RARGen mandates the use of collected risk analysis documents as inputs; general project documentation, even documents intermixed with risk descriptions, cannot be used. RARGen currently neither attempts to extract risks from natural language, nor does it try to abstract and associate general project or design context or information. It is assumed that contents in a risk analysis document include risks, risk reduction actions, or risk assessment information.

Risk Repository Constructor. After collecting and preparing these risk analysis documents, we removed punctuation, stopwords and performed stemming [3] to preprocess the risk-analysis documents. RARGen accepts risk analysis documents in two templates: (1) a top-level risk reduction plan in a table or Excel file, and (2) a detailed risk reduction plan usually documented in HTML or MS Word files. RARGen detects and disambiguates Risk Association Patterns (RAPs) from the preprocessed corpus of risk analysis in preparation to construct (or add to) the risk repository.

Pattern Similarity Analyzer and Cluster. RARGen leverages the Latent Semantic Analysis (LSA) [5, 6] and the K-means clustering algorithm [4] to construct a reduced risk repository by merging redundant RAPs with the similar semantic meaning among RAPs from the risk repository constructed earlier.

Closed Frequent Itemset Miner. RARGen applies the frequent itemset mining algorithm [7] to mine closed frequent itemsets from the reduced risk repository. From the closed frequent itemsets, the Risk Association Rule Learner of RARGen (described next) mines Risk Association Rules (RARs) as desired rule sets. An RAR is an RAP that occurs over a given frequency.

Risk Association Rule Learner. RARGen takes the Risk Terms Files (RTF) containing all distinct words depicting risks and closed frequent itemsets, and then returns RARs. Based on the RTF, RARGen categorizes the mined closed frequent itemsets into three types: (1) those containing only risk terms, (2) those containing both risk terms and other words, and (3) those containing no risk terms. RARGen mines RARs from only Types 1 and 2 but not from Type 3 because itemsets from Type 3 contain no information about specific risks and have no value for risk reduction.

4. EVALUATION

We next discuss the RARGen evaluation in meeting the value-generating ***Objectives 1-4*** as stated in Section 1. The evaluation is

conducted on 20 e-services projects from USC-CSSE, which are publicly accessible at http://greenbay.usc.edu/csci577/fall2007/site/projects/index.html. While these one-year-long projects were undertaken by graduate software engineering course students, all are "real" in the sense that there were real clients, who wanted real systems, developed with real processes, encountering and managing real risks. Many of the students were full-time working software professionals with substantial development experience.

Objective 1: Automated collection of RAPs into a useful repository

RARGen recovered more than 150 human-readable *RARs* with approximately 3000 occurrences from over 180K words mined from 20 projects without prior knowledge or processing from domain experts. It took less than 6 hours from a single novice student to prepare the risk analysis documents, execute the RARGen tool, and cross-validate the results with manually generated RARs. In contrast, it took over a week for an expert risk researcher (the 1st author) to manually browse the 20 risk analysis documents, dig out 135 RARs (not even prioritized), and cross-validate them.

Hence it is practical to use RARGen for automating generation of a risk repository. The quality of the repository is discussed subsequently with Objective 2. Here we address another critical part of our Objective 1 – is the repository "useful?" The repository could be used to generate risk assurance aids such as easily maintained domain/organization-specific top-10 risk lists that are directly applicable to a given project rather than being high-level inspirational. Additionally, examination of the 20 e-Service projects' risk analysis documents revealed that they tend to be a mix of high-level and detailed risks.

Objective 2: Improve confidence in risk review

There are two aspects to be evaluated for Objective 2. First, how high confidence can we have in the mined RARs? Second, how high confidence can we have in applying the mined RARs and the RARGen approach for risk assurance activities?

We investigate the first aspect with two questions: "what is the quality of the mined RARs?" and "do the mined RARs represent real rules in practice?" Because the selection of the threshold in closed frequent itemset mining directly influences the quality of mined RARs, we have evaluated the quality of mined RARs based on seven different thresholds. We use *Recall* and *Precision* as two metrics, which are typically used in information retrieval:

$$(\mathrm{I})\,RECALL = \frac{|R_{\mathrm{min}ed} \cap R_{man}|}{|R_{man}|} \qquad (\mathrm{II})\,PRECISION = \frac{|R_{\mathrm{min}ed} \cap R_{man}|}{|R_{\mathrm{min}ed}|}$$

where R_{man} is the set of manually identified RARs from risk analysis documents of historical projects (for this evaluation). As indicated earlier, 135 RARs were manually identified from the 20 e-Service projects by an expert risk researcher. These RARs are used as our golden benchmark and we assume that our expert out-performs typical risk analysts in general. $R_{\mathrm{min}ed}$ is the set of RARs mined by RARGen. $R_{\mathrm{min}ed} \cap R_{man}$ (True Positive) indicates how many "real" rules were mined from the projects. Our approach achieves a Precision of 67%-97% and Recall of 71%-85% under seven different thresholds.

We also note that risk assurance aims to provide increased confidence in the completeness and correctness of risk analysis for a given system through independent review. Because Recall and Precision are defined relative to manual effort, higher values in these measures do not translate into higher confidence, especially with respect to errors of omission. Note that our objective is not to replace human risk analysis, but to complement it by providing an unbiased and comprehensive check against historical documents. Because RARGen was able to perform automatically at a high level relative to a human, the resulting RARs serve as a second "independent" review that increases confidence when cross-checked with manual results. That is, there is objective information to help guide the manual assurance by partitioning the effort into investigation sets. RARs found both by RARGen and manual efforts are "true positives" whereas those found by only RARGen would be considered as "false positives" (or possible manual omissions), and those found by only manual efforts would be considered as "false negatives". If an RAR is thought not to exist in manual risk analysis and it does not appear in the RARGen set, then this RAR is some indication of a "true negative" and risk assessment tends not to focus on non-risks.

Objective 3: Increase cost-effectiveness of risk assurance

What is less obvious is RARGen's effectiveness when applied to new projects. Earlier we evaluated RARGen's recall and precision relative to a repository manually generated by our expert. However, it is possible that RARGen would perform poorly when supporting risk assurance activities such as identifying incomplete risk identification or reduction actions. Here again we assess effectiveness relative to our expert as this design represents the best known practice when dealing with unknowable values (e.g., we cannot determine the "true" number of RARs present in the projects used in the evaluation).

To evaluate the effectiveness in using a generated risk repository, we conducted a study that compares mined RARs with RAPs on a "new" project (i.e., a project whose risks are not in the repository). For the new project, we use RARGen to identify RAPs, which are then compared to RARs in the repository to detect risk analysis errors (e.g., omitted risks, incomplete specifications). The detected errors are finally compared with the risk analysis errors identified by our expert to determine which errors were actually relevant. Because the majority of the 20 projects are COTS-based, to reduce complexity, we focus our study on only COTS Interoperability risks.

We performed 20 separate evaluations via **L**eave **O**ne **O**ut **C**ross **V**alidation (LOOCV) [10]. We left 1 out of 20 projects as the "new" project and used the risk analysis documents from the remaining 19 projects as the input for RARGen. Table 1 shows LOOCV comparisons for 6 projects with true design defects resulting from the missing/incorrect RARs detected by RARGen. The column "# Detected from RARGen" shows the total number of risk specifications (risk description and corresponding mitigations from the left-out projects' risk analysis documents) that do not conform to, or are omitted with respect to RARs mined from the remaining 19 projects' risk analysis documents. Risks in the left-out project that cannot be matched with an RAR in the repository are ignored. Because we have not specified any rules for determining "related" risks, RARs in the repository that cannot be matched with an RAP in the left-out project are considered an "omission" although strictly speaking some RARs may be omitted because they are not relevant to the left-out project. The column labeled "# of FP" is the number of false positives as determined by our expert. These are detected errors

from RARGen that are incorrect or, more frequently, omitted RARs irrelevant to the left-out project.

Table 1. Effectiveness in detecting risk analysis errors

Project	# Detected by RARGen	# of FP	# Design Defects
3	5	4	1
8	17	16	5
9	7	5	2
14	11	8	3
19	10	8	2
20	8	6	2

The column labeled "# of Design Defects" shows the number of true COTS interoperability design defects in the system traced to risk analysis errors by our expert. This metric is used mainly to indicate the utility of risk assurance. If a "true" risk error is identified, then it is conjectured that the number of design defects is correlated with the number of errors. In most of our cases, this number of true design defects is equal to (*#detected by RARGen - #FP*) and indeed our small sample seems consistent with the conjecture except that in Project 8, one risk error leads to five design defects.

Objective 4: Support refinement of risk mitigations

To evaluate our final objective, we investigate whether RARGen is able to provide meaningful support for a common risk insufficiency problem – risks with inadequately or incompletely specified reduction actions. For this objective, we focus on a single project, the "Conference Room Reservation System" Project 14. It is a database and web-based application. Table 2 shows the two manually specified RAPs defining reduction actions related to the interoperability risk between SQL Server and Web Application Server**.**

RARGen mined RARs 5.1-5.4 in Table 3 from the other 19 e-service projects. RARs 5.1 and 5.2 address the interoperability risk between the MS SQL and Coldfusion. If the support-based ranking had been used, the interoperability analysis between the IIS Server and Coldfusion specified by RAR 5.3 (whose support was less than the predefined threshold 18), would have not been mined as a rule. However, its subset {Coldfusion, IIS Server} (whose support is greater than 18) is a frequent closed itemset. Thanks to the closed frequent itemset mining, RARGen mined {Coldfusion, IIS Server} as a closed itemset.

Table 2. Manually specified RAPs for *SQL Server* and *Web Application Server* interoperability risk in Project 14

4.1	Backend Interface (SQL Server) ←→ Web Interface (Web Application Server);
4.2	SQL Interface (SQL Server) ←→ Backend Interface (Web Application Server);

Table 3. Sample RARs mined from 19 e-service projects

5.1	{Interoperability(SQL Server, Coldfusion)}➔ { Backend Interface (SQL Server) ←→ Web Interface (Coldfusion)};
5.2	{Interoperability(SQL Server, Coldfusion)}➔ { SQL Interface (SQL Server) ←→ Backend Interface (Coldfusion)};
5.3	{Interoperability(Coldfusion, IIS Server)}➔ {Backend Interface (Coldfusion) ←→ Web Interface (IIS Server)};
5.4	{Interoperability(Web application Server)}➔{Coldfusion, IIS Server};
5.5	{Interoperability(MS SQL, Apache, Windows Server 2003)}➔ { Safari, JVM};

Through this example, we see that mined RARs can provide a useful reference on how similar risks get reduced in previous projects and can be used to verify the completeness and correctness of a current risk analysis for a new project within the same domain.

5. CONCLUSIONS

With our evaluation results, we address the questions that we proposed to guide this research as below. ***Object 1* evaluation** shows that it is practical to use RARGen for automating generation of a useful risk repository with low maintenance costs from pre-processed unstructured SE artifacts taking into account the manual pre-processing efforts. In ***Objectives 2&3* evaluation**, we have evaluated the quality and utility of RARGen on 20 publicly available e-service projects. Our results indicate that RARGen can mine RARs with high Recall and Precision. We have also shown that RARGen is cost-effective in supporting risk assurance activities, helping address the significant problems of incompleteness and error-proneness in manual risk analysis. The evaluation results also demonstrate low effort relative to the effectiveness of results compared to non-automated risk assurance. Even with as-well Recall and Precision relative to human-based risk analysis, by applying RARGen in tandem with traditional human-based risk assurance, we can largely reduce effort and improve confidence in assurance. ***Objective 3* evaluation** also shows the total number of risk specifications (identified from the left-out projects' risk analysis documents) that do not conform to, or are omitted with respect to RARs mined from the remaining 19 projects based on LOOCV comparisons. A study in ***Objective 4* evaluation** presents a specific example of RARGen identification of missing risk and reduction actions. It also shows how specific risk reduction actions are suggested using mined RARs.

6. REFERENCES

[1] B. W. Boehm, "Software Risk Management: Principles and Practice". IEEE Software, 8(1): 32-41.

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

[3] Stemming: http://www.tartarus.org/martin/PorterStemmer/

[4] J. B. MacQueen: "Some Methods for classification and Analysis of Multivariate Observations," Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability, UC Berkley Press, 1:281-297, 1967.

[5] S. C. Deerwester, S. T. Dumais, T. K. Landauer, G. W. Furnas, and R. A. Harshman. "Indexing by latent semantic analysis" Journal of the American Society of Information Science, 41(6):391-407, 1990.

[6] T. Landauer, P. Foltz, and D. Laham, Introduction to Latent Semantic Analysis, Discourse Processes, 25:259-284, 1998.

[7] G. Grahne, J. Zhu, "Efficiently using prefix-trees in mining frequent itemsets", In Proc. 1st IEEE ICDM Workshop on Frequent Itemset Mining Implementations, 2003.

[8] R. Lutz, C. Mikulski. "Operational anomalies as a cause of safety-critical requirements evolution", Journal of Systems and Software, 65(2):155-161, 2003.

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

[10] Ron. Kohavi, "A study of cross-validation and bootstrap for accuracy estimation and model selection", Proc. of the 14th International Joint Conference on Artificial Intelligence 2 (12): 1137–1143, 1995.