TBRIM: Decision Support for Validation/Verification of Requirements

Dr. Joseph J. Romano
Software and Systems Integration Group
SAIC, Inc.
McLean, VA 22102 USA

Dr. James D. Palmer
Systems Engineering Department
George Mason University
Fairfax, VA 22030 USA

ABSTRACT

A decision support system has been developed that provides a structured approach to aid the validation/verification of requirement sets and enhance the quality of the resulting design by reducing risk. Additionally, an automated implementation of this approach has been completed utilizing the Advanced Integrated Requirements Engineering System (AIRES) software. Corroboration for the application of this decision support system has been attained from automated and manual testing performed on the system specification of a large-scale software development and integration project. The decision support approach, called the Test-Based Risk Identification Methodology (TBRIM), has been used to detect four major types of requirements risk potential: ambiguity, conflict, complexity, and technical factors. The TBRIM is based on principles of evidence testing and eliminative logic developed by Bacon and Cohen for making decisions under uncertainty. New techniques for detection of complexity and technical risks have been added to existing methods for identification of risk from ambiguous and conflicting requirements. Comparisons of the automated and manual test results on risk category-by-category basis showed good correlation, and category-independent comparisons showed improvements that were consistent with expectations. Benefits from use of this decision support system include higher accuracy, consistency, and efficiency in the validation/verification of requirements. Knowledge of senior personnel can be captured to provide an expert system for less-experienced personnel.

1. INTRODUCTION

Developing software-based systems of any meaningful size is often very difficult, time consuming, and expensive; taking years and costing millions of dollars. Unfortunately, experience over a long period of time has shown that software development projects have a high likelihood for large budget overruns, extensive schedule slippage, and legions of dissatisfied customers and users. Many software development projects experience severe problems or are outright failures despite efforts to improve this situation through the use of such techniques as structured design, CASE tools, fourth-generation programming languages, and object-oriented methods.

Difficulties with requirements specifications have been extensively documented as a major contributor to the problems experienced on software projects [2], [7], [12], [17]. The substantial opportunities to mitigate these problems by reducing risk in the requirements phase of the project life cycle are also well documented [1], [3], [6], [9], [16], [17], [18]. These circumstances argue the importance of enhancing the ability of software development teams to identify, assess, manage, and reduce requirements risks.

Software system requirements may contain minimal risk, enormous risk, or some level in between these extremes. At the start of a software development project, the actual level of risk from requirements is uncertain. For risk reduction to occur, the requirements with risk potential first must be identified in some manner (preferably using a logical and structured approach such as presented in this paper). Project management, technical, and functional personnel then can work together to assess the likelihood and potential impact of the risk associated with these requirements. The information resulting from this assessment subsequently can be used to prepare a strategy to eliminate, manage, or mitigate the most significant risks.

Because errors in risk identification during requirements verification can be carried through to each subsequent step in the lifecycle, and the associated negative impact increases the later the error is corrected, minimizing risk in the requirement phase is very important to the entire design and development process. However, the identification of risks in software requirements specifications is often fatiguing, intellectually complex, and very consuming of scarce and expensive personnel resources. Consequently, the results of these efforts are frequently incomplete, inconsistent, and inaccurate.

This paper addresses the problem of how to improve the identification of requirements with risk potential, the results of which can be used to help mitigate software development risk. The focus of the work discussed in this paper was the development of a methodology for creating a decision aid to help reduce errors and uncertainty in the products of the requirements review process and to improve the efficiency of the personnel performing this task.

2. DESCRIPTION OF THE TBRIM

The TBRIM provides a consistent, repeatable, and structured process for reviewing requirements for risk potential. The TBRIM evidence-test method uses an approach derived from the works of Bacon and Cohen [16]. These works advocate the application of multiple tests (in order of decreasing importance) and eliminative logic to prove or disprove hypotheses. These evidence-test concepts have been used extensively in legal applications and knowledge-based computer systems. The extension of these concepts to requirements risk identification provides a highly feasible and powerful means to help control the costs and schedules associated with software development projects.

In the application of the evidence-test method for requirements review, failure of a test produces evidence that a requirement possesses risk potential of the associated test type. These tests are performed in order of decreasing importance so that the most significant risks are identified first. Using a strict Baconian approach, testing of a requirement should cease upon initial failure. However, in the TBRIM all tests are executed against all requirements for a more comprehensive evaluation of risk potential.

Evaluation of requirements using the TBRIM can be performed manually or with the aid of an automated requirements assessment tool. The results of applying the TBRIM are a set of metrics, which indicate how well individual requirements and the aggregated requirements set are able to pass tests for detection of various types of risk potential. These metrics can be formatted and displayed in tabular or graphical form to create a decision aid to facilitate requirements risk identification. This information can be used to improve software risk management by providing a warning of potential software requirements problems early in the development life cycle; before budget and schedule problems become unmanageable.

Use of the TBRIM with an automated tool can reduce the level of effort required for identifying specific types of risk by large percentages. If all requirements are to be reviewed manually, the combined use of the TBRIM and an automated tool can facilitate and improve the accuracy of this effort by identifying in advance the types of risk that may exist in each requirements.

This paper presents and compares the results of manual and automated applications of the TBRIM on the requirements specification for a large-scale software system development project. Both types of tests were performed on the 748 requirements that comprised the specification to detect risk potential from ambiguity, complexity, conflict, and/or technical factors.

The manual TBRIM application was performed using hard copies of the specification by project personnel responsible for developing the new system. These personnel also had functional experience with and knowledge of the system to be replaced. As the first step in testing, each of the participants was provided brief written instructions on how to detect each of the four types of risk potential. They were then asked to review of the specification for these types of risk potential and given forms to record the results of the reviews. As a final step, a group review was conducted on the individual results and a consolidated set produced.

The automated testing using the TBRIM presented in this paper was conducted making extensive use of AIRES applied to an ASCII format version of the specification. AIRES is a tool that supports the acquisition and analysis of software requirements. This tool was developed at George Mason University in Fairfax, Virginia [11]. AIRES features include automated formatting, characterization, sizing, syntax analysis, and key word searching of requirements.

3. AUTOMATED TESTS AND RESULTS

The tests described in this paper expanded upon the tests types and criteria used for detecting ambiguity and conflict that were investigated in previous work performed at George Mason University [5], [8], [13], [14]. In addition, new tests were created to investigate requirements risk potential from complexity and technical factors.

In the TBRIM approach to identifying requirements with risk potential described by this work, several key word tests were composed to examine requirements for four major risk categories (i.e. ambiguity, complexity, conflict, and technology). These tests were then applied to the subject specification to determine the requirements with potential risks. In addition, results of AIRES analyses of the composition of the requirements were used in complexity assessments.

Key word tests are frequently used in information retrieval systems. These tests consist of multiple test items that are key words or phrases, which represent or strongly suggest the attribute that the test is being conducted to detect. In the execution of AIRES key word test module, each item in the designated test file is compared to each sentence in the requirements file being tested. Matches between test items and the requirements file that are consistent with the criteria set for the test items, are recorded in output files.

Technical Risk
Three major sources of technical risk in system requirements are technical standards, technology, and specification referential integrity. The ability of a system to meet requirements that incorporate technical standards or technology-based criteria can typically be established through analysis, inspection, or testing of the developed system. Therefore, failures of the system to meet these requirements can be determined without dispute by or cost/schedule relief to the developer.

Referential integrity in a specification requires that requirements incorporating references to other parts of the specification, or other documents are correct, so that the requirements are consistent, complete, and accurate. Lack of referential integrity can produce technical problems and/or schedule delays during design or development of a system, or can result in deficiencies in the functionality or performance of the delivered system.

Technical Standards: Technical standards may specify methods for performing work; components or composition of the work; or minimum and/or maximum capacities and features. These standards may be created by local, state, or federal laws; by government agencies such as the Department of Defense (DOD) or the Environmental Protection Agency (EPA); by technical organizations such as ANSI or IEEE; or by some part of a corporate organization. To meet requirements that specify technical standards, the system must comply with all of the relevant provisions of the referenced standard, some of which may pose a high level of technical difficulty.

Evaluating this type of requirement is often complicated by the fact that referenced standards usually are not included in or attached to the specification. In this case, the requirement may impose technical obligations that are unknown to the reviewer, inappropriate for the application, in draft form, or even nonexistent. Consequently, requirements that invoke technical standards can embody a high risk potential that must be detected and resolved.

The contents of the technical standards test for risk potential consisted of 87 test items. These items included of the names of government agencies and standards organizations as well as the abbreviations for each. Generic terms that are part of the names and descriptions of many types of technical standards documents were also included. In addition, since technical standards often have dates, each of the twelve months and their abbreviations were added. The numeric representation for years (e.g. 1988, 1989) could be added to this test. However, the AIRES key word search would require a separate entry for each year of interest. This was judged to be unnecessary for this application because these dates would be detected by numeric tests that are part of technology testing.

The results of the technical standards tests showed a total of 262 test item occurrences, but only 40 of the 87 items in the test set were detected. Although there were 262 test item occurrences detected, some of the requirements contained multiple types of test items. After accounting for these multiple occurrences, there were only 171 requirements that actually tested positive for technical standards risk potential.

Technology: Technology requirements impose quality factors such as performance or reliability; or specify limitations on characteristic such as electrical, environmental, or physical features. Technology requirements often include a unit of measurement (e.g. seconds) with an accompanying numeric value. To meet these requirements, the system or designated component is typically obligated to meet, be greater or less than, or be within a range of a specified value and unit of measurement. These requirements also can be identified by a generic title, such as availability or response time. The risks associated with this type of requirement are due to possible difficulties or the impossibility of meeting them with current technology or within budgetary and schedule constraints.

The technology risk test consisted of the 320 items. To assist in the assessment, three categories of risk items were created: quality factors and characteristics (120 items), units of measure (147 items), and numbers and symbols (53 items). The items in these categories were selected taking into consideration the requisite functionality of the system as well as the types of components and associated characteristics of the system. This type of information could be deduced from the overview and/or introduction sections of most specifications. Organizations or individuals responsible for developing the system are also likely to have an experience base, which could be useful for creating additional test items for these tests.

A total of 918 occurrences of technology test items were found in the specification. However, only 127 of the 320 test items were present. Although a total of 918 test item occurrences were detected, there were only 419 requirements that actually tested positive for technology risk potential.

Referential Integrity: Requirements often include references to other parts of the specification (e.g. other requirements, tables, figures) or other documents. This situation introduces referential integrity risk potential due to the possibility that the information specified is inaccurate, does not exist, or is incorrect. The negative consequences of this condition may occur early in the design, when needed information is found missing, or it remain hidden until installation at the customer site when an interface does not work.

If a reference is not included in a specification, the risk can be categorized as incompleteness. If this reference is to another technical document, the risk is the same as the technical standards risk discussed previously. Thus, referential risk tests frequently produce positive results for requirements that also exhibit technical standards risk potential, providing a higher confidence level of technical standards risk detection.

The contents of the test for referential integrity risk potential that was used in this work consisted of 84 test items. These test items consisted of two types. The first type was composed of words and phrases that are used frequently as a means of referencing other information by name or location within the specification (e.g. refer to, shown in). The second type consisted of the names or identification of the repositories of the information in the specification (e.g. Appendix, Figure, Table).

The test showed 270 occurrences of referential integrity test items in the specification, but only 21 of the 84 test items were present. Although there were 270 test items detected, there were only 138 requirements that actually tested positive for referential integrity risk potential.

Ambiguity Risk
Ambiguity is a widely-recognized source of requirements risk [8], [13], [14]. A requirement can be considered ambiguous if it contains a word or phrase (typically an adjective or adverb) that can be interpreted in more than one way by different people. System compliance with ambiguous requirements usually can not be verified in a manner acceptable to all parties involved. Thus, if a specification contains requirements with ambiguity, there exists a risk that the developed system will not meet the users expectations or needs.

Ambiguous requirements often result from the use of qualitative descriptions (e.g. fastest, greatest, highest) rather than quantitative terms containing numeric objectives and associated units of measure (e.g. 10 seconds). Ambiguous requirements also may be open-ended and defy closure over a reasonable time period. For example, a requirement may specify that a system must be operational "all the time." In this case, disagreement could arise on how long test data must be accumulated to confirm this requirement really has been met. In other cases, deliberate attempts to make flexible requirements (e.g. at least, a minimum of) can create *de facto* ambiguity because a system that exactly meets the requirement may fall short of hidden expectations.

The TBRIM approach for identifying requirements with ambiguity risk potential incorporates the same procedures used for detecting technical risks. A set of key word test items were created and subsequently applied to the specification to determine evidence for or against requirements having ambiguity risk potential.

The contents of the test used for identifying requirements with ambiguity risk potential were developed incrementally, using ambiguity test items obtained from prior research as the nucleus for this process [10], [14]. As the first step, each of the 85 test items from the prior research was expanded into a test set through the use of a thesaurus. These test sets were then consolidated into a single set of 921 items by removing duplicate terms.

A total of 690 occurrences of ambiguity test items were detected in the specification. However, only 117 of the 921 test items were present. Although 690 test items were found in the specification, only 414 requirements actually tested positive for ambiguity risk potential.

Conflict Risk
Conflict between requirements exists when one requirement prescribes conditions, circumstances, or results that are inconsistent with those prescribed by another requirement or makes another requirement very difficult or impossible. Requirements can conflict on the basis of quality goals that they set, or by the functional and nonfunctional characteristics that they dictate.

Conflict risk is different from other types of requirements risk and is more difficult to detect. This increased difficulty arises because a requirement examined by itself, without reference to other requirements in the specification, may appear to be without risk. This is because conflict occurs between pairs of requirements, and a requirement can conflict with more than one other requirement. In addition, conflicting requirements may be widely separated or dispersed throughout the system specification.

Quality goals: Quality goals are derived from a category of nonfunctional requirements characterizing features and capabilities that can be used to quantitatively measure and evaluate a system [8]. These requirements are intended to set (possibly conflicting) expectations for capabilities and/or constrain the cost of system design, development, operation, or maintenance.

The identification of requirements with quality goal risk potential is a two-step process. First, requirements that dictate quality goals must be detected, characterized, and grouped by type. Second, all quality goal types discovered in the specification must be compared on a pair-wise basis. The requirements contained in each of the conflicting groups are then considered to be conflicting (e.g. performance and maintainability).

Although conflict is bidirectional, there are often some quality goals that are more important or have greater utility to the system owners or users [8]. In this case, the requirements associated with the lesser-important quality goal become candidates for de-emphasis, revision, or removal from the specification. If any of the conflicting quality goal pairs are found, then all of the requirements in one of the groups conflict, to some degree, with all of the requirements in the other group. Thus, the number of conflicting requirements is equal to the total number of requirement in the two requirements groups.

The list of 11 quality goals and associated definitions used in this work were synthesized from the prior works [8], [14]. As the first step in this research, a set of test items was created for each of the quality goal types resulting in a total of 672 test items. Then, key word searches, using each test set, were performed to identify and categorize each of the quality goal requirements.
A total of 993 test items occurrences were found in the specification using the test sets. However, only 222 of the 672 test items were found. Although there were a total of 993 occurrences of test items in the specification,

only 526 requirements actually tested positive for risk potential associated with quality goal conflicts.

Functional and Nonfunctional: Functional and nonfunctional conflicts are more specific than quality goal conflicts. These conflicts arise from the presence of requirements that dictate opposing inputs, operations, or outcomes. In order to detect this type of conflict, it is first necessary to adequately categorize types of functional and nonfunctional requirements and to define those pairs of categories that have conflict potential. It is then necessary to determine a mechanism for allocating requirements to the categories and to apply that mechanism to the requirement set.

Samson proposed a classification scheme for functional and nonfunctional requirements based solely on the presence of key nouns and verbs [14]. Thus, test sets containing 1070 noun and 916 verb test items were constructed and applied to the specification. These tests resulted in the classification of 93 percent of the requirements, with multiple test items per requirement.

The results from the classification tests were then used in conjunction with associated conflict-pair definitions created by Palmer [10]. This exercise revealed that the use of key words for detecting functional and nonfunctional conflict: 1) produced a large number of requirements in many of the conflict pairs, and 2) many of the requirements were part of more than one conflict pair. Thus, there were too many possible conflicts and use of this method furnished unacceptably weak evidence of conflict risk potential.

Complexity Risk
Requirements that are lengthy, have complicated sentence structure, contain faulty word selection, or include domain or technology specific terms can be very difficult to understand. Misinterpretation of these complex requirements can lead designers, developers, and testers to deliver a system that fails to meet the intended purpose or user needs.

The appropriateness of word selection and terminology depends greatly on the reader's background and education as well as the subject matter. Even though requirements are meant for well-educated technical personnel, there are numerous words and phrases that are used frequently in technical and non-technical writing that reduce, rather than improve, a reader's understanding. Furthermore, although domain or technology specific terms may be essential to convey the intended need, these terms can cause difficulties if not properly understood by the responsible software development personnel.

Sentence Length and Structure: Relatively short simple, sentences are most easily understood [4]. Thus, requirements that are long (greater than 26 words) and/or have more than one verb (e.g. compound or complex sentences) possess complexity risk potential. For this work, the output of the AIRES style module was used to assist in the manual review of the sentence length and the number of verbs in each requirement to detect complexity risk. This review revealed that 169 of the 748 requirements in the specification had more than 26 words and 91 requirements had more than one verb. A total of 217 requirements showed positive test results from either or both of these complexity test criteria. Thus, 43 requirements tested positive for both sentence length and structure complexity risk potential.

Word Selection: The effectiveness of a document can be severely decreased as a consequence of faulty or inappropriate word selection. Thus, to reduce complexity, jargon, clichés, and commonly misused words and phrases should be avoided.

As part of this work, key word searches were performed to detect words or phrases recognized as problems and the source of complexity risk. These test sets included big words, wordy phrases, inflated and meaningless phrases, redundancies, and clichés. These test sets consisted of a total of 165 test items. The key word tests revealed a total of 179 test items occurrences that were produced by only 19 of the 165 items. These 179 occurrences were contained in 158 requirements.

Domain and Technology Specific Terms: In those cases where uncommon domain or technology terms are unavoidable, it is extremely important that the specification includes appropriate clarification, definitions, or references to where definitions can be found. Otherwise, the selection and use of otherwise appropriate words in a requirement can introduce complexity risk into the specification.

The AIRES similarity module was used to determine the number of unique terms in the specification and to select terminology that might be difficult to understand by other than domain experts. These terms were then used in key word searches to locate the requirements containing those terms. The list of unique terms consisted of 1632 items, which were reviewed manually to identify those that might be particular to the domain or technology. This review resulted in the selection of 83 items (primarily acronyms) that were used in key word searches. These searches revealed a total of 444 test items occurrences. However, only 258 requirements actually tested positive for this type of risk potential.

Consolidated Results: The consolidated results for all of the individual complexity tests totaled 633 requirements. However, the consolidated number of requirements with complexity risk potential from any factor was only 464.

4. COMPARISION OF RESULTS

Comparisons between manual and automated test results were performed to obtain an indication of the capability of the automated approach to match and improve on the

TBRIM manual process through more comprehensive, consistent, and repeatable results. The comparisons were performed on both category-by-category and category-independent bases.

The category-by-category comparisons show how well the manual and automated test results correlate for each risk type. Table 1 compares manual and automated test results for technical, ambiguity, conflict, and complexity (Type 1 though 4). This table shows the number of requirements identified with risk potential by the manual review, automated tests, manual review only, and automated tests only.

Case	Manual	Auto.	Manual Only	Auto. Only
1	146	501	31	386
2	132	414	41	323
3	13	526	6	519
4	17	464	0	447

Table 1 – Comparison by Category

The category-independent comparisons relate the risk identification capability of the two methods, exclusive of differences in risk type classification. The manual review identified 255 requirements as possessing some type of risk. This compares to automated tests identifying 737 requirements as having risk potential. Of the 255 requirements identified by the manual review, 253 were also identified by automated tests (just two manual-only results). These results show a high correlation between the manual review and automated tests, although the automated tests found a much higher number of requirements with risk potential.

These results are consistent with expectations for the automated TBRIM approach, which is intended to provide a decision aid for manual efforts. This is to be accomplished by first using automated methods to create a relatively large set of requirements with risk potential. This set is then reduced during the manual effort by removing requirements with little or no actual risk. In the manual review results, project personnel had the opportunity to concurrently perform the latter part of this process. Thus, identification of a smaller number of requirements as having risk potential would be expected.

5. CONCLUSION

The implication of these statistics is that the TBRIM shows great promise as a decision aid for requirements assessment, particularly for the identifying requirements risk potential from ambiguity, complexity, conflict, and technical factors. It is also likely that this decision aid has the capability to enhance the speed and accuracy of the review process and contribute in a meaningful way to reducing risk in software systems development.

REFERENCES

[1] B. W. Boehm, *Software Engineering Economics*, Prentice-Hall, 1981.

[2] B. W. Boehm, *Software Risk Management*, IEEE Computer Society Press, 1989.

[3] R. N. Charette, Software Engineering Environments, Concepts and Technology, McGraw Hill, 1986.

[4] W A. Damerst, *Clear Technical Reports*, Harcourt Brace Jovanovich, 1972.

[5] R P Evans, *Methodological Framework for Requirements Assessment*, George Mason University, Fairfax, Virginia, 1994.

[6] W. W Gibbs, "Software's Chronic Crisis," Scientific American, September 1994, pp. 86-95.

[7] A. M Jenkins, J. D. Naumann, and J. C. Wetherbee, "Empirical Investigation of Systems Development Practices and Results," *Information Management*, Volume 7, 1984, pp. 73-82.

[8] M. E. Myers, *A Knowledge-Based System for Managing Software Requirements Volatility*, George Mason University, Fairfax, Virginia, 1988.

[9] T. Nakajo, and H. Kume, "A Case History Analysis of Software Cause-Effect Relationships," *IEEE Transactions on Software Engineering*, Volume 17, Number 8, August 1991, pp. 830–837.

[10] J. D. Palmer, *REQSPERT: A System for Assisting Requirements Analysis and Generating Test Plans*, George Mason University, October 1990.

[11] J. D. Palmer and R. P. Evans, "Advanced Requirements Engineering System (AIRES)," *Proceedings form the Complex Systems Engineering Synthesis an Assessment Workshop (CESAW' 93)*, Washington, DC, July 20-22, 1993.

[12] D. Phan, D. Vogel, and J. Nunamaker, "The Search for Perfect Project Management," *Computerworld*, September 26, 1988, pp. 95-100.

[13] J. M. Robinson, *Risk Assessment in Software Requirement Engineering: An Event Driven Framework*, George Mason University, Fairfax, Virginia, 1995.

[14] D. Samson, *Automated Assistance for Software Requirements Definition*, George Mason University, Fairfax, Virginia, 1988.

[15] D. A. Schum, *Evidential Foundations of Probabilistic Reasoning*, John Wiley & Sons, 1994.

[16] R. Stevens and G. Putlock, "Improving Requirements Management," *INCOSE Insight*, Issue 16, 1997.

[17] M. Van Genuchten, "Why is Software Late? An Empirical Study of Reasons for Delay in Software Development" *IEEE Transactions on Software Engineering*, Volume 17, Number 6, June 1991, pp. 582-590.

[18] R. M. Weintraub and J. Burgess, "Drastic Revisions Are Likely for Air Traffic Control Contract," *Washington Post*, April 11, 1994, p. A-6.