Software Risk Management: Requirements-Based Risk Metrics

James D. Palmer, Fellow   Richard P. Evans, Member

Center for Software Systems Engineering, George Mason University, Fairfax, VA, 22030-4444

Abstract

There are several problems associated with current approaches to requirements-based risk metrics for software risk management. For example, the approaches are qualitative and limit risk considerations to probability of occurrence and severity of impact. They also do not take into account the relative size of the requirements category of which they are a part; i.e., the development effort needed to implement that requirements category. Finally, quantitative factors are not used. The Advanced Integrated Requirements Engineering System (AIRES) provides automated support for the quantitative identification and extraction of requirements-based software risk metrics throughout the requirements engineering life cycle. The AIRES risk assessment framework and techniques for the integrated application of both semantic and syntactic rules provide for effective, efficient, and comprehensive identification of requirements at risk in large complex multiple segment systems. A review of a portion of the requirements for a Navy Gun Mount System is included to provide an example of the use of AIRES to support the development of requirements-based risk metrics.

Keywords:

Software, requirements, quantitative, risk, metrics, size.

Background and Introduction

Software risk management has been demonstrated to be problematic [Boeh91, Fair94]. This has been an especially difficult problem at the requirements phase of a software development project due to the lack of metrics of relative risk within and across software requirements. One goal for software risk management is to identify which components of the system present the highest risk; i.e., identification of those components that are critical to the success of the software, require the greatest time and effort to produce, and have problems or issues that possess the potential to become "show stoppers" if not resolved.

Current methods for identification of components at risk have been qualitative; based on the subjective judgment of the software developer. Using these approaches, only limited success has been noted in identification and resolution of risk at the requirements phase. A major need is for a *quantitative* approach for the detection, identification, and extraction of software components that are potential causes of risk to the successful development of a software project.

Two models that represent the state-of-development in risk model application that use *qualitative* subjective assessments to identify potential risk elements are the Boehm "Top Ten Problems" model [Boeh91] and the SEI Risk Assessment Model [Chit93]. Fairley [Fair94] in a recent paper has provided another subjective approach to apply the findings of these qualitative models. The Top Ten Problem model requires the software development team to subjectively detect and identify the top ten problem areas in order of importance that confront the software development project at a given point in time. These top problems are addressed in order of importance until the problems are resolved (or held for later resolution). Using the Boehm Spiral model, another sweep of the project is made and another top ten problem list is prepared, and these are addressed and resolved. This process continues throughout the software development cycle or until the software development team is satisfied that risks have been satisfactorily detected and addressed. The SEI Risk Assessment Model is equally qualitative. The SEI model proposes an extensive list of questions; the answers to which are used to infer the degree of risk for the software development project. The questions address the typical development phases of a software development project requirements, design, code, test, and implementation, without regard to the particular software development process utilized. None of these models provides a quantitative approach to the identification of risk elements in software requirements. An Advanced Integrated Requirements Engineering System (AIRES) that is under development supports a *quantitative* approach and provides a complement to *qualitative* risk assessment processes by adding *quantitative* requirements-based risk metrics derived from the actual requirements.

The probability of risk in requirements is related to the existence of problems and issues within individual requirements or groups of requirements statements. If these are not detected and satisfactorily resolved they will lead to an unsuitable software product. Typical of the types of problems and issues found in requirements are use of ambiguous terminology, redundant terms, inconsistent construction, statements in conflict, non-traceable statements, and incomplete requirements. Quantitative determination of problems and issues such as these form the basis for the visualization of risk metrics. The approach to visualization of these problems and issues is through use of a classification structure to place risk metrics in categories defined by risk potential.

Risk assessment typically considers two metrics: (1) the probability of occurrence based on the existence of issues; and (2) the value of the loss incurred or the expected value due to the severity of impact. Two of the four possible high-low value combinations of these two parameters are then treated as high risk: (1) high loss with high probability; and (2) high loss, even with low probability. The AIRES-supported risk metric approach that is described adds a third factor to these two: the relative size of separate categories of the overall requirements. The hypothesis is that two risks, with equal combinations of unsatisfactory outcome and loss probabilities, are unequal if the categories of which they are a part differ in size; i.e., will require different levels of effort to implement.

We describe, with illustrations, the application of the AIRES assessment framework, with its categorization process, for automated support to the establishment of three classes of quantitative requirements-based risk metrics. They are counts, risk ratios, and relative size.

Counts include: (1) *counts* of requirements in designated risk categories; (2) *counts* of requirements in a given category; (3) *counts* of requirements, in a given category, that are in separate risk attribute categories. A *risk ratio* is the ratio, in a given category, of the total requirements to those in individual risk attribute categories. *Trend risk ratios* may also be established by capturing the changes in the risk ratios as the basic requirements change. *Relative size* is a text-based measure for the requirements in a given category; i.e., the relative development effort for their implementation. The application of relative size, as a multiplier on both individual risks and risk ratios, enables the definition of *size risk ratios.*

Through the categorization processes, AIRES provides information on requirements with risk attributes such as ambiguity, conflict, imprecision, complexity, volatility, inconsistency, and incompleteness. In this way, software users and developers are provided a quantitative method to identify software components at risk; metrics which give information as to relative degree of risk, risk trend information, the relative size of the components at risk; and mechanisms to manage risk through resolution of problems and issues in requirements.

This paper describes the use of the Advanced Integrated Requirements Engineering System (AIRES), to detect, identify, and extract risk elements during the requirements phase of software development projects. AIRES is under development by the Center for Software Systems Engineering (CSSE), George Mason University [Palm93]. The paper also includes an example application of the AIRES quantitative software risk management process for the development of requirements-based risk metrics. The example included is a review of a portion of the requirements for a Navy Gun Mount System. An overview of AIRES is also provided.

Categorization

As illustrated in Figure 1, we have defined two categories in developing requirements-based risk metrics: *requirement type* and *risk attribute. Requirement type* categories support definition by requirement type; e.g., functional, non-functional, data object and as shown in Figure 1 they are represented by rounded corner boxes. *Risk Attribute* categories form the end-nodes of a classification structure; they are represented by square corner boxes. Both categories may be employed in multiple hierarchies.

Requirement type categories may be established around a wide variety of perspectives that include functionality, data objects, conditional statements (for real-time systems), entity-relationships, or other parameters. For example, there may be requirement type categories around unique data objects. Additional categories below these in a hierarchy may be formed around functionality.

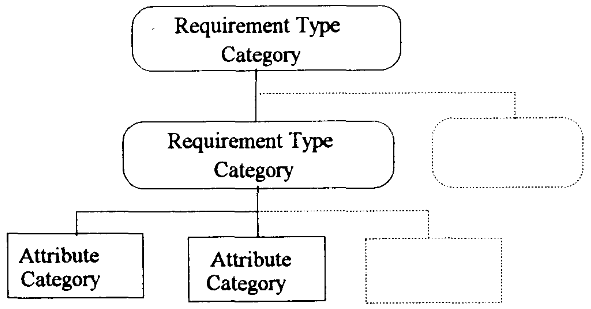

Figure 1. Example Classification Structure and Categories

Attribute categories may be generic or system-specific. Generic risk attributes include ambiguity, complementary completeness, comparative completeness, inexactness, complexity, and volatility. System specific attributes include inconsistency, conflict, and complementary and comparative completeness. *Complementary completeness* is a form of completeness that is identified in prose text through use of domain knowledge-based rules. These rules establish the combinations of categories of requirements the individual terms or phrases and synonyms that must be present to satisfy completeness; i.e., the necessary complements exist. *Comparative completeness* is a function of system-specific knowledge and requires domain knowledge for determination.

AIRES Assessment Framework: CEBRA

In the AIRES assessment framework both requirement type and attribute categories are established in terms of: (1) the characteristics of the text (a designated list of the characteristics for a given category is defined as the *tables*); and (2) the *rules* for the application of the characteristics to the input requirements statements. We use the term classification ectype (CE), where *ectype* is a synonym for a representation, as the name for the combination of the rules and tables to establish a category. With that as the basis for categorization, the AIRES assessment framework is called CEBRA for CE-Based Requirements Assessment, as is shown subsequently in Figure 8.

Relative Sizing

Relative sizing by the assessment framework provides a quantitative measure of the relative size of the software development project either as a whole or in selected requirement type categories. Sizing is based on a measure of parameters such as the number of unique data objects and the associated unique functionality parameters in these categories. Relative size is used as a multiplier on the potentially at-risk requirements in a given category. It is our hypothesis that, with all other factors being equal, if the relative sizes (estimated development efforts) of requirement type categories of requirements differ, then those size ratios are multipliers on the risk measures for individual requirements within each category.

Risk Metrics

The metrics from which risk is to be inferred are determined from the number of problems and issues found in a given requirement type category. The classification structure example of Figure 2 has one tier of requirement type categories: Functions "A", "B", and "C"; and one tier of attribute categories: volatility, ambiguity, and conflict.

The numbers shown, other than those in parentheses, represent the total number of requirements statements in each category. The size estimates of the three requirement type categories are the shown in parenthesis. With these counts, we develop *risk ratios. Risk ratios* are the ratio of counts in the attribute categories divided by the count in their parent requirement type category.

When the size estimates for the requirement type categories (as shown in parentheses) are applied as a multiplier, the resulting metrics are called a *size risk ratios*. As a separate metric, when two values of a risk ratio are compared as a ratio over time, the resulting metric is called a *trend risk ratio*.

To illustrate, the risk ratios using the counts shown in Figure 2 are tabulated in Table 1. The risk ratio for conflict in Function "A" is the ratio of their counts : 45/335 = 0.13. The same *risk ratios* for Functions "B" and "C" are 45/610 = 0.07, and 70/455 = 0.15 respectively. Application of the relative size estimate results in a conflict-based size risk ratio for Function "A" of (0.13)(105) = 13.65, Function "B" of 14.7, and Function "C" of 8.2 from which we propose Function "B" to have the greatest risk potential.

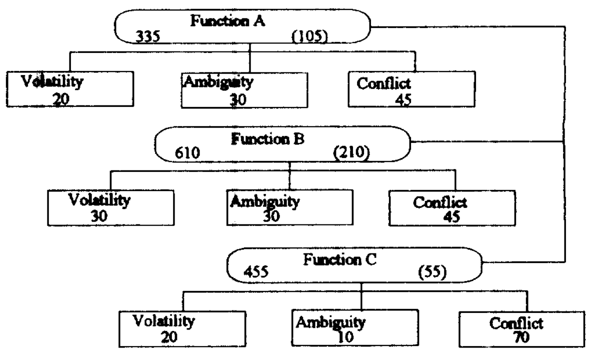

Figure 2. Example AIRES Categorization, with ID counts and *size* estimates.

Function	Attribute Category	Count	Count	Risk Ratio	Risk Ratio	Size Risk Ratio
A			335		105	
	Volatility	20		0.06		6.3
	Ambiguity	30		0.09		9.4
	Conflict	45		0.13		13.6
B			610		210	
	Volatility	30		0.05		10.5
	Ambiguity	30		0.05		10.5
	Conflict	45		0.07		14.7
C			455		55	
	Volatility	20		0.04		2.2
	Ambiguity	10		0.02		1.1
	Conflict	70		0.15		8.2

Table 1. ID Counts, Risk Ratios, and Size Risk Ratios

Various graphical displays of this table of total IDs, risk ratios, and size risk ratios can be employed to support rapid visual estimates of both the individual and relative risk metrics values. Figure 3 depicts the risk attribute total IDs for the three Functions "A", "B", and "C"; their associated *risk ratios* are in Figure 4, and Figure 5 presents the *size risk ratios*.

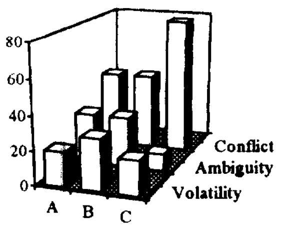

Figure 3. Risk Attribute Counts

Function "C" has the highest total ID attribute counts (70) and the highest *risk ratio* (0.15), while Function "B" has the highest *size risk ratio* (14.70). Of particular interest is the potential significance of the relative size. Without those multipliers, the requirements for Function "B" appeared to have the least risk ratios, in all three contexts. With application of the relative size, Function "B" has the highest relative risk ratios in all three.

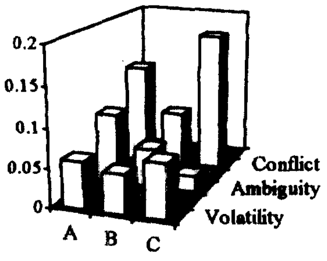

Figure 4. Risk Ratios

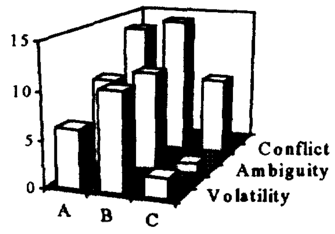

Figure 5. Size Risk Ratios

Navy Gun Mount Example

A review of a portion of the requirements for a Navy Gun Mount System provides a sample of the scope of risks that can be evaluated by the risk ratios developed by categories. Fifteen requirements in a *firing circuit control* category, as listed in the appendix, were considered.

For a *complementary completeness* risk, or in this instance non-traceable risk attribute category, three of seven states (Casualty, Test, and Training) were not specified. That resulted in a complementary completeness risk ratio of 3/7 = 0.43.

One requirement (ID # 7) was identified in an imprecision risk attribute category. That was due to the use of the clause*: any selection other than*: *"If **any selection other than** fire SALVO is made at this point, the firing Circuit Enabled signal shall be cleared to disable the effect of the fire SALVO selection, and the Salvo Warning signal shall be cleared."* This generated an imprecision-based risk ratio of 1/15 = 0.07. For purposes of illustration, the state incompleteness is depicted in the manually-generated state transition diagram of Figure 6. Both the risk ratios are depicted graphically in Figure 7. Thus, those requirements IDs at risk are clearly shown.

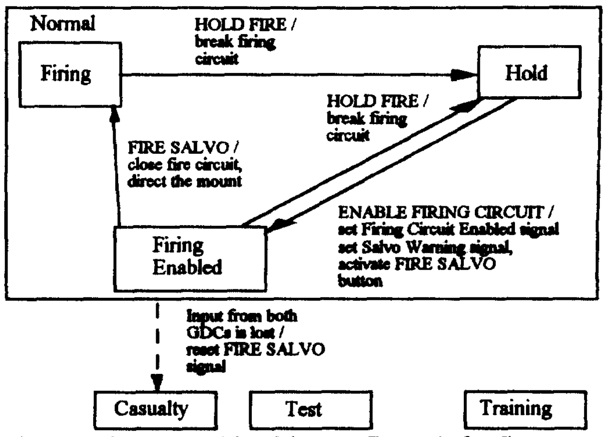
Normal
Firing
HOLD FIRE / break firing circuit
Hold
HOLD FIRE / break firing circuit
FIRE SALVO / close fire circuit, direct the mount
ENABLE FIRING CIRCUIT / set Firing Circuit Enabled signal set Salvo Warning signal, activate FIRE SALVO button
Firing Enabled
Input from both GDCs is lost / reset FIRE SALVO signal
Casualty
Test
Training

Figure 6. State Transition Diagram Example for Gun Mount Firing Circuit Controls

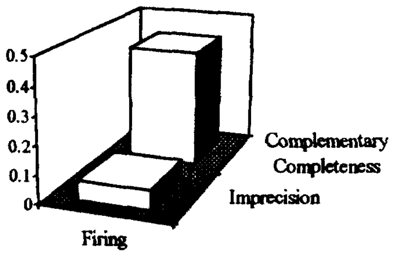

Figure 7. Two Risk Ratios for Gun Mount System *Firing Circuit Control* Category

AIRES Overview

AIRES supports the development, assessment, and transformation of natural language requirements to a semi-formal or formal language. In its present form, AIRES is a laboratory model capable of providing automated support to users and/or designers. As depicted in Figure 8, AIRES provides an integrated systems approach to requirements activities that includes a Process, a Methodology, and a Database. These operations are integrated yet enable independent application of current or in-development methodologies (separately or in combination) to provide automated support to user and/or designer.

The need to address requirements at the earliest possible stage of systems development has been amply demonstrated. Research has shown that up to 80% of the deficiencies in delivered software may be attributed to problems in requirements engineering [Boeh87]. Errors in requirements that are undetected until software test and operation cost 200 times more to correct than if those same errors were detected and corrected during the requirements engineering phase [Jaff91]. The most frequently cited areas for the highest potential payoff in systems development, especially for large complex systems, have centered on ways to assure

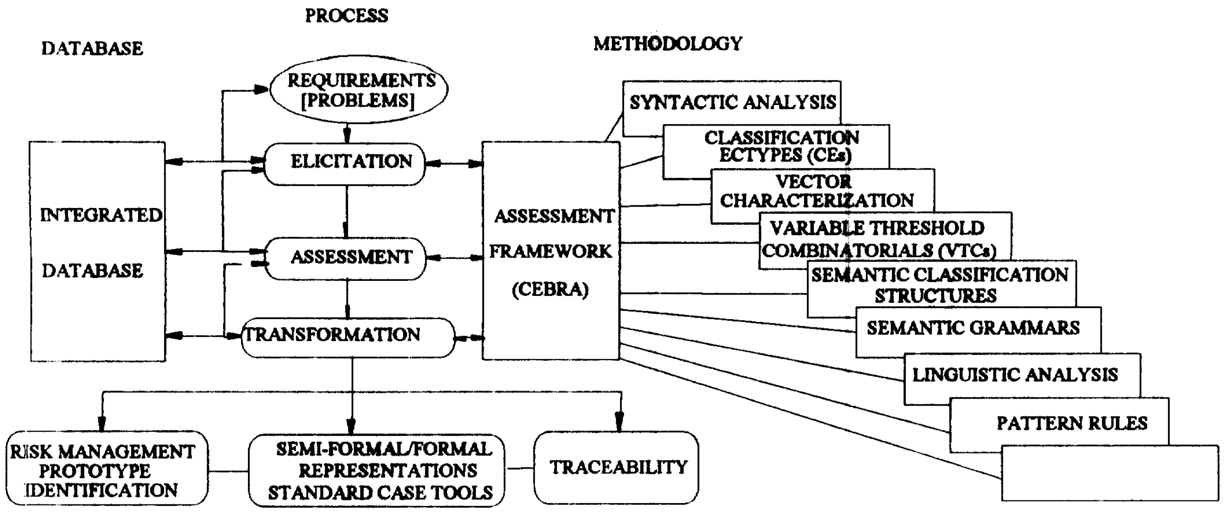
PROCESS
DATABASE
METHODOLOGY
REQUIREMENTS [PROBLEMS]
ELICITATION
ASSESSMENT
TRANSFORMATION
INTEGRATED DATABASE
ASSESSMENT FRAMEWORK (CEBRA)
SYNTACTIC ANALYSIS
CLASSIFICATION ECTYPES (CEs)
VECTOR CHARACTERIZATION
VARIABLE THRESHOLD COMBINATORIALS (VTCs)
SEMANTIC CLASSIFICATION STRUCTURES
SEMANTIC GRAMMARS
LINGUISTIC ANALYSIS
PATTERN RULES
RISK MANAGEMENT PROTOTYPE IDENTIFICATION
SEMI-FORMAL/FORMAL REPRESENTATIONS STANDARD CASE TOOLS
TRACEABILITY

Figure 8. AIRES Architecture: Process and Methodology with Integrated Database

the quality of natural language requirements prior to transformation to more precise semi-formal or formal language representations, and also on approaches to achieve these needed transformations that combine traceability from the detailed formal specifications to the original natural language requirements.

AIRES provides three processes to handle requirements information. These are supported by three frameworks, elicitation, assessment, and transformation. Taken together, these are both necessary and sufficient for the development of requirements for entirely new systems, requirements dependent upon legacy systems, and requirements to modify existing systems. The frameworks provide capabilities such as the capacity to address new concepts, detect and identify issues, determine impact on existing systems, and provide full traceability from originating natural language statements to transformed semi-formal or formal language specifications.

The AIRES research efforts on elicitation have been devoted to development of processes and procedures that enable users to maximize the efforts associated with elicitation. Specifically, AIRES includes approaches to facilitate information extraction from individuals working in a group and groups of individuals who have responsibility for various aspects of a large-scale complex system. The processes have been adapted from the conflict resolution field of study and tailored to meet the specific needs of requirements engineering. Automated assistance has been developed to provide nearly instantaneous feedback to groups through use of the methods contained in the assessment framework. The methods applied in AIRES include innovative features such as semantic grammars, vector calculation of syntactic similarity coefficients, syntactic parsing, and linguistic analysis to assist in organization,, and presentation of concepts.

The AIRES research efforts on assessment have centered on means provide for discovery and to ascertain the presence of and provide for interactive resolution of issues in and across requirements statements and groups of statements. Facilities for interactive use of AIRES are provided for purposes of discovery of vital characteristics and aspects of requirements statements within and across requirements attributes. Issues such as ambiguity, imprecision, conflict, risk detection and identification, and redundancy may be determined through application of this framework. The AIRES assessment framework integrates several unique and innovative capabilities for discovery and issue identification, including: (1) *text-based relative system sizing*; (2) *semantic categorization* of requirements by the use *of semantic classification structures, domain grammars, and linguistic analysis*; (3) *discrimination of multiple subjects and objects* (parts of speech such as nouns, verbs, adjectives and adverbs; and syntactic elements in combinations, such as noun-verb phrases, subject nouns and subject noun phrases); (4) *syntactic categorization*; and (5) techniques for establishing *pair-wise similarity coefficients* (SCs) across requirements statements.

Transformation research has focused on *transforming natural language requirements statements to semi-formal or formal specification languages*. AIRES provides the capability to identify and extract the elements for transformation to semi-formal or formal languages that include *E-R Diagrams, Finite State Machines, Parnas Four-Variable representation, and Data Structure Diagrams*. Full *traceability* is provided from the original natural language statement representation of the requirement to the representation in semi-formal or formal languages.

AIRES incorporates several innovative approaches to support the elicitation and assessment frameworks. These innovations provide automated support to necessary requirements engineering activities.

Conclusions

We have demonstrated an approach to providing automated support to the development of quantitative requirements-based risk metrics for software risk management. We have described the application of the AIRES assessment framework for automated support to the establishment of three classes of quantitative requirements-based risk metrics: counts, risk ratios, and relative size.

Through the use of the AIRES assessment framework category structure that we have presented, requirements statements may be classified, and through this, risk metrics can be determined. The approach uses the semantic and syntactic properties of natural language requirements statements, as well as domain specific properties of the requirements attribute to provide quantitative measures of the presence of potential risk situations.

We have developed AIRES as a CASE Tool to assist user and developer in risk management for software requirements for the detection, identification, and extraction of requirements statements with problems and issues. When used in conjunction with other qualitative methods for the identification of risk in software development, this provides a more complete perspective of the total risk situation. [1]

References

[Boeh87] Boehm, Barry W., "Improving Software Productivity," *IEEE Computer* Vol. 20, No. 9, September 1987

[Boeh91] Boehm, B. W., "Software Risk Management: Principles and Practices," *IEEE Software*, January 1991

[Chit93] Chittister, Clyde, and Haimes, Y. Yacov, "Risk Associated with Software Development: A Holistic Framework for Assessment and Management," *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 23, No. 3, May/June 1993

[Fair94] Fairley, Richard, "Risk Management for Software Projects", *IEEE Software*, May, 1994

[Jaff91] Jaffe, Matthew S., Leveson, Nancy G., Heimdahl, Mats P. E., and Melhart, Bonnie E., "Software Requirements Analysis for Real-Time Process Control Systems," *IEEE Transactions on Software Engineering*, March, 1991

[Palm93] Palmer. J. D. and Evans, Richard P. "Advanced Integrated Requirements Engineering System (AIRES)": *Proceedings, 1993 Complex Systems Engineering Synthesis and Assessment Technology Workshop* (CSESAW '93), 20-22 July, 1993, Washington D. C

Appendix

Firing Category Requirements from Navy Gun Mount Processor Specifications

1. 3.4.3.2.2.5 Firing Circuit Controls.

2. MCF shall provide three single-action buttons to control firing of the gun mount: a HOLD FIRE button, an ENABLE FIRING CIRCUIT button, and a FIRE SALVO button.

3. When the HOLD FIRE button is pressed, this function shall break the firing circuit so shall be disabled.

4. The other two firing selections shall work in conjunction with each other

5. The fire SALVO button shall have no effect unless the ENABLE FIRING CIRCUIT was the last selection made before the FIRE SALVO selection.

6. When the ENABLE FIRING CIRCUIT selection is made, the Firing Circuit Enabled signal shall be attribute, the Salvo Warning signal shall be attribute, and the FIRE SALVO button shall be activated.

7. If any selection other than fire SALVO is made at this point, the firing Circuit Enabled signal shall be cleared to disable the effect of the fire SALVO selection, and the Salvo Warning signal shall be cleared.

8. If the next selection made after ENABLE FIRING CIRCUIT is fire SALVO, the firing circuit shall be closed.

9. And the mount shall be directed to fire via the Interface function (see paragraph 3.4.6).

10. Fire capability from this panel shall not be allowed unless authority has been delegated by GCC, via the digital interface, during normal conditions.

11. During Casualty conditions, the operator shall be given the capability of firing from the DCP without requiring prior delegation from GCC.

12. If input from both GDCs is lost during gun firing, the FIRE SALVO signal shall be reset so as to stop firing.

13. Firing shall be resumed if the operator reselects the fire SALVO button, even though both GDCs may still be down.

14. The firing circuit shall be interrupted during Test and Training

15. The ENABLE FIRING CIRCUIT and FIRE SALVO buttons shall generate a Simulated Firing Order when the doctrine is either Test or Training.

[1]This work was supported in part by the Virginia Center for Innovative Technology.