NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

AN INVESTIGATION OF THE QUANTIFICATION OF THE PROBABILITY OF OCCURRENCE OF SOFTWARE ENGINEERING PROJECT RISKS WITH BAYESIAN PROBABILITY

by

Matthew L. Klabon

December 2007

Thesis Advisor:	John Osmundson
Second Reader:	Paul Kimmerly

Approved for public release; distribution is unlimited

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.			
1. AGENCY USE ONLY ***(Leave blank)***	**2. REPORT DATE** December 2007	**3. REPORT TYPE AND DATES COVERED** Master's Thesis	
4. TITLE AND SUBTITLE An Investigation of the Quantification of the Probability of Occurrence of Software Engineering Project Risks with Bayesian Probability			**5. FUNDING NUMBERS**
6. AUTHOR(S) Matthew L. Klabon			
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000			**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A			**10. SPONSORING/MONITORING AGENCY REPORT NUMBER**
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.			
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited			**12b. DISTRIBUTION CODE**

13. ABSTRACT (maximum 200 words)

This work undertakes an analysis of the methods used in the field of software engineering to measure the probability of occurrence of identified software engineering project risks. The purpose of this work is to investigate the viability of the exclusive use of quantitative values to measure the probability of occurrence of identified project risks within the field of software engineering rather than the qualitative values used in some probability of occurrence scales. More specifically, the goal of this analysis is to investigate the viability of achieving more precise and less subjective quantitative measurements of the probability of occurrence of identified software engineering project risks with Bayesian probability.

14. SUBJECT TERMS Risk, Risk Management, Software EngineeringSoftware Engineering Risk Management, Bayesian Probability, Baye's Theorem, Baye's Formula, Bayesian Networks, Bayesian Belief Networks, BBNs, Probability of Occurrence			**15. NUMBER OF PAGES** 57
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UU

NSN 7540-01-280-5500 Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release; distribution is unlimited

AN INVESTIGATION OF THE QUANTIFICATION OF THE PROBABILITY OF OCCURRENCE OF SOFTWARE ENGINEERING PROJECT RISKS WITH BAYESIAN PROBABILITY

Matthew L. Klabon
Civilian, United States Marine Corps Technology Services Organization
B.S., Purdue University, 2002

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN SOFTWARE ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
December 2007

Author: Matthew L. Klabon

Approved by: Professor John Osmundson
Thesis Advisor

Paul Kimmerly
Second Reader

Peter Denning
Chairman, Department of Computer Science

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

This work undertakes an analysis of the methods used in the field of software engineering to measure the probability of occurrence of identified software engineering project risks. The purpose of this work is to investigate the viability of the exclusive use of quantitative values to measure the probability of occurrence of identified project risks within the field of software engineering rather than the qualitative values used in some probability of occurrence scales. More specifically, the goal of this analysis is to investigate the viability of achieving more precise and less subjective quantitative measurements of the probability of occurrence of identified software engineering project risks with Bayesian probability.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF FIGURES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

The author thanks Professor John Osmundson and Paul Kimmerly for the sound guidance and support they provided over the course of this work. Professor Richard Riehle is also recognized as his risk management instruction was an important part of this work's inception.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION AND STATEMENT OF PROBLEM

Measurement of quantitative entities is best achieved with quantitative numeric values. Identified software engineering project risks are quantifiable entities and measuring their probability of occurrence with quantitative numeric values is superior to the use of qualitative descriptive values.

Quantitative numeric values are superior to qualitative adjectives such as “high,” “medium,” and “low” for measurement as they are not subjective. When quantitative values are derived scientifically through mathematical probabilistic and/or statistical methods and used as intended they are generally not subject to a number of varied, subjective interpretations or misinterpretations by those using them. Quantitative values do not require their users to make judgments regarding the intended meaning of the measurements that are being applied.

The use of quantitative numeric values for measurement rather than qualitative adjectives will result in the communication of these measurements in a way that is likely to be more precise, objective, and clear. This explains why quantitative numeric values are typically used for measurement in the fields of engineering and science. Quantitative numeric values are the standard means of communication for measurements of all sorts.

The use of quantitative values that are derived through a rigorous scientific approach that utilizes sound mathematical techniques instead of qualitative values that are based upon subjective and/or intuitive judgment for the measurement of the probability of occurrence of identified project risks could afford the field of software engineering many benefits. From a software engineering project management perspective these include the ability to better prioritize risks, more efficiently allocate project risk mitigation resources to identified risks, and, as was previously mentioned, communicate the probability of occurrence of identified risks with a medium that is universally understood and is not given to subjective interpretation.

The question of why methods that employ qualitative adjectives to measure the probability of occurrence of identified risks are sometimes used in the field of software engineering is raised. The matter of whether the software engineering discipline itself is suited to the use of qualitative, subjective and/or perhaps less accurate measurement techniques should also be considered.

Both qualitative and quantitative measurement techniques may be found in use in the field of software engineering project risk management to measure and communicate the probability of occurrence of identified software project risks. This work will investigate the reasons qualitative types are used in the field of software engineering and whether quantitative types can and should be used exclusively.

This primary focus of this work will be to resolve the matter of qualitative adjectives being applied to identified software project risks to describe their probability of occurrence. The solution to this matter that will be investigated is the viability of replacing the adjective descriptors used in qualitative probability of occurrence scales in the field of software engineering with quantitative numeric values. This is to achieve more precise and easily understood probability of occurrence measurements.

As this work is addressing the measurement of the probability of occurrence of identified project risks, a more detailed explanation of each of the definitions of these terms should be provided. Relative to the field of software engineering, definitions of "identified software project risks" will be discussed as will the definition of "probability of occurrence" measurements. The value of accurately assessing the probability of occurrence of identified software project risks will be discussed in more detail before the discussion and presentation of the proposed solution to the problem this work addresses.

In general terms, a risk is commonly viewed as an event whose occurrence will result in some detrimental effect upon individuals and/or entities. More specifically, risk is defined as the "possibility of loss or injury" and also, in a way perhaps more directly related to this work, as "the chance of loss or the perils to the subject matter of an insurance contract" and also as "the degree of probability of such loss" (Webster's, 1998).

A software engineering project risk may also be defined and referred to as a hazard if it exists in the context of a safety-critical software system. Safety-critical software is defined as "any software that can directly or indirectly contribute to the occurrence of a hazardous systems state." A hazard is defined as a "state or set of conditions of a system (or an object) that, together with other conditions in the environment of the system (or object), will lead inevitably to an accident (loss event)" (Leveson, 1995). This work applies to hazard type software engineering risks as well as all others.

Additionally, and perhaps less typically considered is the view that a risk should also be viewed as an entity that possesses attributes. An example of an attribute that a risk possesses pertinent to the focus of this work is the probability that it will occur. Another attribute a risk possesses is the effect its occurrence will have on individuals, entities, and the software engineering project itself.

Viewing risks as distinct entities with specific and identifiable attributes justifies viewing them as tangible entities. Tangible entities and their particular attributes may be measured. As a result, identified software engineering project risks can and should be viewed by software engineers as entities to be analyzed, measured and managed (Gluch, 1994). The purpose of the strategies implemented by software engineers to address and mitigate software project risks is primarily to help prevent these risks by anticipating their occurrence rather than reacting to them (Higuera, et al., 1994).

As software engineers can and should view identified software engineering project risks as measurable and manageable entities with identifiable and measurable attributes these attributes should be discussed. Opinions vary somewhat within the field of software engineering on this subject but a number of risk attributes are generally considered to be typical. These attributes are considered worthy of measurement, analysis, and management in the context of a software engineering project. Three risk attributes will be discussed as they are pertinent to this work and its goal of determining the viability of using quantitative values instead of qualitative values for measuring risk probability of occurrence. These attributes are the source of the software engineering project risk, the context in which it exists, and as previously mentioned, its probability of occurrence.

The source of an identified project risk is the aspect of the software engineering project whose existence brings the risk with it. An example of an identified software engineering project risk's source could be an employee whose retention is important to the success of the project. While the employee is an asset to the project, the loss of the employee is a risk.

An identified software engineering project risk's context is the environment in which the risk exists. The attributes of this environment may be dynamically changing throughout the course of the software engineering project. An example of an identified project risk's context is the organization in which a software engineering project is being undertaken.

Probability of occurrence is defined as the likelihood of a risk actually occurring during the course of a software engineering project. Qualitative values such as "high" are sometimes assigned to assess the probability of occurrence of an identified project risk in the field of software engineering. Quantitative values such as ".7" are also used.

There are several types of risks that software engineers must manage during the course of a software engineering project. Edmund Conroe, the founder and owner of Acquisition and Technology Associates cites several "key risk issues" in his IEEE Software journal article "Implementing Risk Management on Software Intensive Projects." Included among these risks are "excessive, immature, unrealistic, or unstable requirements," "unrealistic cost or schedule estimates and/or allocated amounts," and the use of "immature" technologies.

Software engineers may be charged with measuring risks whose occurrence could result in the loss of human life or other types of casualties. Safety-critical software engineering requires that a rigorous approach to risk management being undertaken, including the accurate measurement of identified risks' respective probabilities of occurrence. An example of safety-critical software is an embedded software application that is used to control and regulate the hardware for a medical device that sustains human life. A respirator or other life support type system must function predictably and to its specifications and also handle exceptional conditions in an acceptable manner as human lives may depend upon it. Another example of safety-critical software is that used for the

navigation of commercial, private or government (such as military) aircraft. Unpredictable behavior from this type of software could result in individuals being injured or losing their lives.

As has been previously mentioned in brief, software engineers have two types of scales at their disposal for the measurement of probability of occurrence of identified software engineering project risks. The quantitative probabilistic scale used in the field of software engineering for identifying the probability of an identified project risk actually occurring employs a numeric value. This is a value between 0 and 1 with the value 0 indicating that there is no probability of the risk occurring and the value 1 indicating that the risk will occur. A probability of occurrence value such as .5 that is applied to an identified project risk indicates that the likelihood of the risk actually occurring is 50%. It should be noted that quantitative probabilistic scales may be referred to as "ordinal" scales in the software engineering discipline.

Alternatively, probability of occurrence scales employing qualitative adjective values rather than quantitative numeric value measurements are sometimes used in the field of software engineering. These scales may consist of a varied number of degrees and utilize qualitative, descriptive terms such as "high," "medium," "low," etc., to indicate the probability of an identified software engineering project risk actually occurring. Each of these terms may or may not have a stricter and more meaningful definition to the software engineering organization and software engineers utilizing them.

An example of a software engineering organization that has implemented the use of qualitative values that have more clearly defined meanings to describe the likelihood of identified project risk occurrences is NASA's Software Assurance Technology Center (SATC). SATC utilizes a probability of occurrence scale containing the values "Low," "Medium," and "High" with each value having a specific qualitative meaning (Hyatt, et al., 1996). Table 1 illustrates the specific meanings of these values to the software engineers at the NASA SATC.

Table 1. NASA Software Assurance Technology Center Identified Project Risk Probability of Occurrence Scale

Probability Of Occurrence of Identified Project Risk	Definition of Probability of Occurrence Value
Low	Very likely to meet objectives if current trend continues. Does not need contingency plans.
Medium	Based on current trend, likely to meet objectives. Should have contingency plans.
High	Not likely to meet objectives based on current trend. Implement contingency plan immediately.

The authors of the NASA report that discusses the SATC's qualitative probability of occurrence scale state that "It would be nice to be numeric and quantify the risk and probability of occurrence, the state of the art does not currently permit this."

The Carnegie-Mellon Software Engineering Institute (SEI) proposes a similar method for qualitatively measuring the probability of occurrence of identified software development project risks. Their Software Risk Evaluation (SRE) v. 2.0 method also contains three qualitative values for measurement but uses a quantitative numeric value to represent each (Williams, et al., 1999) and is illustrated below.

Table 2. Carnegie-Mellon Software Engineering Institute Software Risk Evaluation Method v. 2.0 Identified Project Risk Probability of Occurrence Scale

Probability of Occurrence of Identified Project Risk	Definition of Probability of Occurrence Value
3	Very likely.
2	Probable.
1	Improbable.

This qualitative probability of occurrence measurement scale is less detailed and more subjective than the measurements used by NASA's SATC. More specific and practical meanings of the qualitative values "Very likely," "Probable," and "Improbable"

relative to the context of a software engineering project are not provided as they are in the NASA SATC's qualitative probability of occurrence scale. No attempt at quantifying or providing a more detailed qualitative meaning for each of the probability of occurrence values is made with the Software Risk Evaluation Method v.2.0. Software engineers could interpret these qualitative values in any number of ways. Vague qualitative values such as these inaccurately measure the probability of occurrence of identified software engineering risks as they represent, at best, educated guesses. These qualitative values could also be applied to identified software engineering project risks in an arbitrary manner such that their inaccuracy is a detriment to the project.

An example of a "quantitative" probability of occurrence measurement is presented by Barry Boehm in the IEEE Software journal article "Software Risk Management: Principles and Practices." The article refers to the probability of occurrence measurement scale discussed as quantitative. This is qualified by Boehm's statement that, "As with most other decision-analysis quantities, is the problem of making accurate input estimates of the probability." The "quantitative" measurements for identified software engineering project risks proposed by Boehm are excerpted from his article and are illustrated below.

Table 3. Excerpt From Barry Boehm IEEE Software Article "Software Risk Management: Principles and Practices"

Quantification of Probability of Occurrence of Identified Project Risks

Probability of Occurrence of Identified Project Risk	Definition of Probability of Occurrence Value
Frequent	0.7-1.0
Probable	0.4-0.6
Improbable	0.0-0.3

Boehm's measurement of the probability of occurrence of identified software engineering project risks uses descriptive qualitative values to measure their probability of occurrence. A range of quantitative values defines each qualitative value that more clearly defines the meaning of each of the qualitative values in the scale.

While this probability of occurrence system of measure is more clearly and easily understood than the two previous examples it is not quantitative. This system of measurement is a combination of qualitative and quantitative measurements. This is because the qualitative terms "Frequent," "Probable," and "Improbable" are used. These and other qualitative terms used to measure the probability of occurrence of identified software engineering project risks are open to the individual and subjective interpretations of the software engineers and software engineering project managers using them.

Additionally, the quantitative numeric values employed in Boehm's probability of occurrence measurement system are not derived through scientific methods. Like the two previous examples, the measurements are obtained through methods that represent, at best, educated guesses. Boehm's estimates are however perhaps of greater value than the previously presented probability of occurrence scales as they are more clearly defined. The fact that Boehm qualifies the accuracy of the probability of occurrence measurements with the caveat that the input estimates are problematic is of benefit. This qualification makes it clear that the reader should not understand them to be derived through purely scientific methods and view the measurements as absolutely sound.

Elaine M. Hall's comprehensive book on risk management, "Managing Risk, Methods for Software Systems Development" proposes a method for quantifying the probability of occurrence of identified software engineering project risks that is very similar to Boehm's. While there are some differences in the content of Boehm's and Hall's methods, including the fact that Hall's contains a greater number of degrees of measure and is somewhat more detailed, the manner in which qualitative and quantitative values are combined to describe probabilities of occurrence are identical. The problems this approach presents are identical to those outlined in the discussion of Boehm's method for quantifying the probability of occurrence of identified software engineering

project risks and all apply to Hall's method for quantifying the probability of occurrence for the same reasons they apply to Boehm's. Hall's method is illustrated below.

Table 4. Excerpt From Elaine M. Hall's "Managing Risk, Methods for Software Systems Development"

Probability Evaluation Criteria

Probability	Uncertainty Statement	Evaluation
> 80%	Almost certainly, highly likely	5
61-80%	Probable, likely, probably, we believe	4
41-60%	We doubt, improbably, better than even	3
21-40%	Unlikely, probably not	2
1-20%	Highly unlikely, chances are slight	1

Some important points are raised by the discussion of the Carnegie-Mellon Software Engineering Institute's Software Risk Evaluation Method v. 2.0, Boehm's quantification/qualification of probability of occurrence measurement techniques and Hall's probability evaluation criteria method. A quantitative system for measuring the probability of occurrence of identified risks need use quantitative values only to measure the probability of occurrence of project risks. A value such as .2 assigned to an identified software development project risks to measure its probability of occurrence indicates a 20% chance that the risk will actually occur. A value of .7 indicates a 70% chance that the identified risk whose probability of occurrence is being measured will occur. These quantitative numeric values need not be combined with some qualitative adjective or adjectives describing the probability of occurrence of the identified risk.

If the numeric values used to measure the probability of occurrence of identified risks are derived through sound scientific methods such as statistical and/or probabilistic methods the addition of qualitative adjectives to their value degrades their accuracy and meaning. Qualitative adjectives should not be combined with scientifically derived numeric values to measure the probability of occurrence of identified software

engineering project risks. As has been previously discussed, qualitative terms are open to the individual and subjective interpretations of the software engineers using them. Qualitative adjectives only serve to confuse the otherwise clear meaning of quantitative numeric measurement values when used in combination with them.

By the same token, qualitative descriptive values do not lend themselves to being described with quantitative values. The quantitative measurement "high" that is assigned to an identified software engineering project risk's probability of occurrence can be interpreted as a quantitative value by software engineers in a number of ways. Some software engineers may interpret the value as a probability of occurrence higher than 50% and lower than 100%. If the occurrence of the risk will have a large negative impact on the software engineering project some software engineers may deliberately interpret the definition of "high" as a probability of occurrence that is higher than 90%. In so doing they will be assured that risk mitigation resources are heavily focused upon the identified project risk.

The quantitative and qualitative types of techniques for measuring the probability of occurrence of identified software engineering project risks should not be used in combination. Qualitative measurement systems seem to benefit somewhat from their measurement values being more clearly defined with quantitative numeric values but more detailed qualitative descriptions of the practical meaning of each value is more appropriate.

Quantitative numeric probability of occurrence measurement systems are degraded when combined with qualitative, descriptive attempts at furthering their definitions. There is no need to qualitatively describe the meaning of a numeric probability of occurrence value that is derived through scientific methods.

Accurately assessing the probability of occurrence of identified project risks is important to implementing sound software engineering practices. Roger L. Van Scoy states in the 1992 Software Engineering Institute report "Software Development Risk: Opportunity, Not Problem" that "Until we use a disciplined and systematic way to identify and confront technical risk, we will never be able to control the quality, cost, or

schedule of our software products." The creators of the Software Risk Evaluation v.2.0 method apparently did not consider Van Scoy's advice when developing their probability of occurrence measurement scale.

There are numerous uses for the probability of occurrence value of identified project risks that are quite valuable and of great consequence relative not only to the respective projects on which they exist but also to the field of software engineering as a whole. A common application of the probability of occurrence value measurement is as a variable in the risk severity formula (Software Technology Support Center, 2005). This formula may be referred to in slightly different terms depending upon the software engineering project or published work in which it is being used and/or discussed as may the variables in the equation but the meanings of these are identical. This formula is often used within the field of software engineering and referred to in numerous risk management related and other published software engineering works and is described by the equation:

$z = ab$
where
z = risk severity
a = probability of occurrence
b = potential negative impact

The variable z, risk severity, is defined as the "seriousness" of the risk by the Software Technology Support Center. The variable a is the probability of the risk actually occurring and is defined in the same manner as has been discussed in this work. The variable b, the potential negative impact, is a measure of monetary, schedule, or project performance impact that is expected to be incurred should the identified risk actually occur.

As has been previously demonstrated and discussed, the value of the probability of occurrence variable in this equation is not likely derived through rigorous mathematical techniques and/or historical project post-mortem data but through methods that are often heavily or entirely dependent upon intuitive or purely arbitrary means that are counter to sound scientific method. The "Understanding Risk Management" article cited above from which the risk severity formula was taken provides no guidance on how

to derive the probability of occurrence variable. One might fairly assume that as with the previously discussed examples of probability of occurrence scales the value of the probability of occurrence variable is expected to be derived with a single or combination of approaches that entail the use of intuition, guesswork, and/or a method invented by the software engineering organization who wishes to apply the formula to calculate the severity of the risks it has identified on its projects.

These approaches to deriving the value of the probability of occurrence of an identified software engineering project risk are employed and advocated (through insinuation when no method for derivation is recommended) despite the fact that, as was illustrated in the previous example of how the value may be applied on an actual software engineering project, it is frequently quite important to effective project risk management. The probability of occurrence of a risk also frequently plays a key role in the processes used by the members of a project's leadership to decide how project resources for and within risk management programs are allocated. This may also mean that the success of the project is at least somewhat dependent upon the efficacy of the method used to derive the probability of occurrence value for each identified project risk. In the context of a safety-critical project the importance of the probability of occurrence values of identified risks is of even greater concern as effective risk management is an activity that is vital to the prevention of the software system being produced causing loss of human life and/or casualty once it is in use.

II. JUSTIFICATION OF WORK

There are a number of benefits to quantitatively and accurately assessing the probability of occurrence of identified software engineering project risks. There is also a substantial amount of evidence and number of experts in the field of software engineering supporting the notion that accurately assessing the probability of occurrence of identified software engineering project risks is a necessity. Assessing the probability of occurrence of identified risks is not a discipline that may be taken lightly or viewed as optional by software engineers during the course of a software engineering project.

The viability and ultimate success of the strategies employed by software engineers to mitigate identified risks depends on the measurements that describe the attributes of risks. Accuracy is an important key to successful risk management and also important to the overall success of the software engineering project.

Inaccurate measurements may result in wasted project resources such as project funding and/or unexpected risk occurrences. Overestimating the likelihood that a risk will occur may result in software engineering project management resources being over-expended and wasted to mitigate identified risks. Underestimating the probability of occurrence of identified software engineering project risks may result in unexpected occurrences of risks that may be expensive both in monetary terms and to the overall success of the software development project (Charette, 1991).

The importance of applying accurate measurements to identified software engineering project risks cannot be understated. Boehm and Demarco state that developing "metrics and tools for reasoning about risk management's return-on-investment issues" is important to the discipline of "fully effective software risk management" (Boehm, et al., 1997). The "return-on-investment" issues they are speaking of are the same types of cost effectiveness/project resource management issues related to risk management that Charette cites. In addition, software engineering project risk management techniques have been shown in scientific case study to produce cost benefit returns (Freimut, et al., 2001).

There are additional issues software engineers should consider with regard to the importance of accurately assessing the probability of occurrence of identified software engineering project risks. In 1995, an estimated $59 billion dollars was spent by American companies on cost overruns for software engineering projects. That same year, another $81 billion dollars was expended by private sector companies in the US on canceled software engineering projects (Keil, et al., 1998).

It should be clear that this work's goal of investigating the viability of accurately quantifying the probability of occurrence may be of some importance to the field of software engineering and has the potential to advance the state of the discipline.

III. PRESENTATION OF WORK

The viability of the field of software engineering using quantitative values exclusively to measure the probability of occurrence of identified software engineering project risks instead of qualitative values should be investigated. Qualitative values such as "High" should not be used to describe the probability of an identified software engineering project risk occurring when a more accurate and clearly defined value such as ".7" could be derived through scientific methods.

It would appear that quantitative values are always better suited for the task of measuring the probability of occurrence of identified software engineering project risks. There is however a considerable amount of support for the use of qualitative scales within the field of software engineering that is based upon sound reason and logic. It should be noted that support for the use of qualitative scales for measuring the probability of occurrence of identified software engineering project risks does not necessarily mean that the software engineer voicing his or her support is entirely in favor of the exclusive use of one type of measurement over the other.

Robert Charette advances a rather straightforward view in support of the use of qualitative scales to measure the probability of occurrence of identified software engineering project risks. Charette states that a "major risk with risk analysis is that it often over-relies upon producing numbers, and does not rely enough upon human analysis and common sense to interpret the results. Similarly, the existence of competing analyses usually results in arguments ensuing over the analysis techniques, rather over what the results may be indicating." (Charette, 1991).

There are other opinions critical of the use of quantitative numerical values being applied to identified software engineering project risks to measure their probability of occurrence. These include the view that numeric values may be just as subjective and inaccurate as qualitative values when used incorrectly and/or arbitrarily. While quantitative values appear to be precise as they utilize numerical values rather than qualitative adjectives "Ordinal risk assessment scales are often incorrectly applied. Mathematical operations cannot be applied to scores obtained from uncalibrated ordinal

risk assessment scales. Risk values generated by mathematical operations on uncalibrated ordinal scales will almost always be meaningless and may hide true risk issues" (Conrow, et al., 1997).

There is research-based evidence indicating that qualitative probability of occurrence measurements may be effective. One study indicates that qualitative values may be effective in addressing the matter of measuring identified software engineering project risks when experienced project managers and technical staff members are brought together to decide upon and reach consensus on the probability of occurrence of such risks. Participants could choose from one of three qualitative values to indicate the probability of occurrence of identified software engineering project risks. These values were "High," "Medium," or "Low" (Kontio, et al., 1996). This approach seems to be in line with Charette's "human analysis and common sense" as it utilizes the experience of software engineers.

Quantitative values are superior to qualitative adjectives for indicating the probability of occurrence of identified software engineering project risks. As identified software engineering project risks are tangible entities with quantifiable attributes the use of quantitative values instead of qualitative for their measurement is viable. It is however relatively difficult to scientifically derive quantitative values to indicate the probability of occurrence of identified software engineering project risks for a number of reasons.

The intangible physicality of the products that software engineers are charged with creating results in engineering processes that contain a number of dynamic variables not found in other engineering disciplines. More mature engineering disciplines that produce tangible products such as bridges, electrical circuits, or mechanical devices may often have more stable, "tried and true" engineering processes that better lend themselves to quantified measurement. The tangibility of the products produced by these disciplines requires and encourages stable product specification. The tangible nature of the products also requires the engineers of these products to more strictly adhere to specifications and other project requirements.

Factors endemic and particular to the discipline of software engineering include the design and development of a product whose requirements and functionality may be

relatively easily redefined. The processes currently employed to engineer software products are relatively immature in comparison to other engineering disciplines and there is not a large set of industry standard methods or software engineering project post-mortem data available to software engineers. These sorts of standards and data would be useful in aiding software engineers in the establishment of a method for measuring the probability of occurrence of identified software engineering project risks quantitatively.

Qualitative measurements of the probability of occurrence of identified software engineering project risks are effective when the software engineers using them share a common understanding of the practical meaning of the qualitative adjectives being employed. These definitions vary from organization to organization however and are largely subjective. They are also derived from and dependent upon the particular processes and culture of each individual software engineering organization. Additionally, the definitions and practical meaning of the probability of occurrence of identified software engineering project risk measurements used are not usually derived from quantified methods in these organizations but on the personal experience of their respective software engineers. As a result, these methods of qualitatively measuring the probability of occurrence of identified software engineering project risks are not adaptable for use by all, if any, other software engineering organizations.

The context, source, and other attributes of identified software engineering project risks directly influence their probability of occurrence singly or in combination. This influence is complex and often dynamic and irregular enough that the quantified measurement of these risk attributes through scientific methods is difficult. The quantification of probability of occurrence of identified project risks clearly requires an advanced method that analyzes and considers all the factors that combine to influence their probability of occurrence.

Charette's emphasis on the importance of "human analysis and common sense" for risk management rather than scientific methodology indicates that a method for quantifying the probability of occurrence of identified software engineering project risks is needed. The statement that the software engineering "state of the art does not currently permit" quantification of the probability of occurrence of risks made by the members of NASA's SATC also supports this claim.

Charette's "human analysis and common sense" can be effective for managing software engineering project risk and should be used by software engineers when methods for quantifying probability of occurrence are insufficient or nonexistent. However, quantified numeric values that are derived through scientific methods for indicating the probability of occurrence of identified project risks should be viewed as the ideal.

As has been previously discussed, the probability of occurrence of an identified software engineering project risk is determined by the manner in which it is influenced by its attributes. An identified software engineering project risk's attributes are quantifiable and combine to determine its probability of occurrence. As the engineering processes and other attributes of individual software engineering organizations vary, so too will the manner in which the attributes of identified software engineering project risks combine to determine their probability of occurrence.

The engineering processes and other attributes of a software engineering organization influence the specific attributes of the identified project risks that determine their respective probabilities of occurrence. Additionally, the attributes of the software engineering organizations themselves may be attributes of identified project risks. Examples of software engineering organization attributes could be its software engineering processes, risk management procedures, the technology or technologies it works with, and its size.

Any method for quantifying the probability of occurrence of risks must account for and quantify the degree and manner in which each of the individual attributes of a given risk combines to determine its probability of occurrence. The attributes of a risk may combine in a way that influences its probability of occurrence in a manner that is greater or less than the sum of its individual attributes. Additionally, the method must also consider the dynamic nature of these attributes and facilitate appropriate revisions to the probability of occurrence of the identified software engineering project risk as needed.

Quantifying the probability of occurrence of identified software engineering project risks requires consideration of the fact that certain attributes of identified risks may influence probability of occurrence more than others. As an example, the probability of occurrence of a key employee leaving a software engineering project is likely to be heavily influenced by the attribute of the employee's salary and whether it is competitive for the job market. The employee's health is another attribute of the risk that influences its probability of occurrence. If the employee is in good health this attribute will not likely increase the risk's probability of occurrence to as great or any degree than if the employee's health were poor.

Statistical software engineering project post-mortem data coupled with conditional probabilistic (including Bayesian) mathematical techniques could also be useful in the creation of a method for quantifying the probability of occurrence of software engineering project risks. Post-mortem data collected from a number of software engineering organizations could be used to help establish a universal method for accurately quantifying the probability of occurrence of identified software engineering project risks. Individual software engineering organizations utilizing the method could also utilize their own project postmortem data to tailor the risk probability of occurrence quantification method for their organization.

More specifically, the field of conditional probability, including a consideration of the Bayesian effect upon conditional probability, provides methods for quantifying the probability of occurrence of identified software engineering project risks. These methods can accommodate Charette's "human analysis and common sense" as well as "hard" statistical data (including software engineering project post-mortem data) and these two types of methods may be used in conjunction with one another.

The following discussion of conditional probability and the Bayesian effect is preceded by an introduction to some basic concepts of probability. Though this discussion draws from R. Lyman Ott's "An Introduction to Statistical Methods and Data Analysis, Fourth Edition" the formulas and concepts discussed are common knowledge within the field of mathematics and areas in which they are often applied such as actuarial science. The examples are original to this work to illustrate how the techniques

may be adopted and applied to quantify the probability of occurrence of identified project risks within the field of software engineering.

A. CLASSICAL AND BAYESIAN PROBABILITY THEORY

An introductory type discussion of probability and terms related to the field that does not assume the reader is familiar with the topic should be undertaken to afford a basis for the discussion of how it may be applied to quantify the probability of occurrence of identified software engineering project risks. Three definitions of probability will be discussed in this introduction to probability.

The first definition of probability that will be discussed is the classical interpretation of probability which was derived from games of chance. Classic examples are those that involve observations related to determining the probability of a coin toss such as "there is a 50% chance the toss will produce a result of 'heads'." Another is related to the probability of drawing a card such as the ace of spades from a deck. As there is only one ace of spades in a deck of cards, which contains 52 cards, the probability of drawing the ace of spades is 1/52 or approximately 2%. Relative to the classical interpretation of probability definition, an outcome is the term used to describe each possible unique result and an event is the term used to describe a collection of outcomes.

The classical interpretation of probability states that the probability of occurrence for an event E is determined by the ratio of the number of outcomes that are in favor of event E, N_e, over the total number of outcomes that are possible, N. This may also be described as:

$$P(\text{event E}) = N_e / N$$

The classical interpretation of probability assumes that and depends on all outcomes being equally likely. If this assumption does not hold to the instance to which it is being applied the resulting probabilities will be erroneous.

The second of the three definitions of probability that are being discussed is an empirical, or experience based, approach that is defined and referred to with what is called the relative frequency concept. This definition of probability states that the

probability of occurrence of an event E is determined by the ratio of n_e, which is the number of times E occurs in an experiment, over n, which is the number of times that the experiment is conducted. This may also be described as:

$P(\text{event E}) \approx n_e / n$

This definition of probability produces an approximate result as it is derived through the observation of several repetitions of the entity being observed. In this case this is the event E.

The third and final definition of probability that is discussed in this work is referred to as the subjective or personal probability. As its name implies, this definition of probability involves individuals making subjective/personal judgments about the probability of occurrence of an event which can vary from person to person and cannot be verified. This is identical or quite similar to the previously presented examples of methods that are typically used to quantify the probability of occurrence of identified project risks within the field of software engineering such as Hall's Probability Evaluation Criteria.

There are some properties of probabilistic calculations that should be discussed. One of these that has been previously discussed in brief is that the probability of event A or P(A) will always be a value greater than or equal to 0 and less than or equal to 1. This may be restated as P(A) will always lie in the interval from 0 (this means that there is no probability of the event occurring) to 1 (this means that the event will definitely occur) or as:

$0 \leq P(A) \leq 1$

Another property states that two events A and B are mutually exclusive if the occurrence of the events eliminates the probability of occurrence of the other event. In other words, A and B are mutually exclusive if the occurrence of A makes it impossible for B to occur.

It follows that if A and B are mutually exclusive the probability of either event A or B occurring is:

P(either A or B) = P(A) + P(B)

The complement of an event is the event not occurring. In other words, the complement of event A is event A not occurring. The symbol for event A not occurring is:

$\bar{A}$

It follows that the sum of the probability of event A, P(A) and the probability that event A will not occur $P(\bar{A})$ is 1. This may also be represented as:

$P(A) + P(\bar{A}) = 1$

It may seem that this sum should be zero but an example should make this property clearer. Consider a coin toss experiment that uses two coins. If numerous repetitions of tossing the two coins indicates that the probability of occurrence of the results being "two heads," which may be though of as event A, is .75 then the probability of another outcome whose results are not "two heads," or event $\bar{A}$ is .25. The sum of these two values is 1.

The union of two events A and B is defined as the set of all outcomes that are included in either A or B or both A and B. This is represented symbolically as:

$A \cup B$

The intersection of two events A and B is defined as the set of all outcomes that are included in both A and B. This is represented symbolically as:

$A \cap B$

The probability of the union of two events A and B is:

$P(A \cup B) = P(A) + P(B) - P(A \cap B)$

As an introductory discussion to probability has been undertaken the definition of conditional probability may now be discussed. The conditional probability of nonzero probability of occurrence event A given nonzero probability of occurrence event B is defined as:

$$P(A|B) = P(A \cap B) / P(B)$$

Event A could be used to represent the event of occurrence of an identified software engineering project risk whose probability is influenced by the known probability of occurrence of the event B. The previously discussed influence of various factors on the probability of occurrence of identified software engineering project risks is described in a simplistic way by the formula above with the probability of occurrence of one event being influenced by the known probability of occurrence of another. A more sophisticated application of the conditional probability of occurrence formula above could be useful in quantifying the probability of occurrence of identified software engineering project risks.

Additionally, the conditional probability of event B given event A is:

$$P(B|A) = P(A \cap B) / P(A)$$

The definition of conditional probabilities above allows the multiplication law to be defined, this is:

The probability of the intersection of two events A and B is:

$$P(A \cap B) = P(A)P(B|A)$$
$$= P(B)P(A|B)$$

The difference between these two definitions lies in what needs to be calculated and what is known. If $P(A \cap B)$ and $P(A)$ are known, $P(B|A)$ can be calculated. $P(A \cap B)$ can be calculated if $P(A)$ and $P(B|A)$ are known.

Baye's Theorem allows conditional probabilities to be updated as is needed and/or possible. This means that data regarding the probability of occurrence of a risk, such as a software engineer of a given level of experience leaving a project, could be

collected and/or derived for each project while it is underway and used in conjunction with the probability of occurrence data that was derived from several software engineering projects over a period of time. A formal definition of Baye's Theorem is below.

Baye's Theorem

If $A_1, \ldots, A_k$ are mutually exclusive states of nature and if $B_1, \ldots, B_m$ are m possible mutually exclusive observable events, then

$$P(A_i|B_j) = P(B_j|A_i)P(A_i) \;/\; P(B_j|A_1)P(A_1) + P(B_j|A_2)P(A_2) + \ldots + P(B_j|A_k)P(A_k)$$
$$= P(B_j|A_i)P(A_i) \;/\; \sum_i P(B_j|A_i)P(A_{i,})$$

Baye's Theorem considers some number k of underlying events $A_1, \ldots, A_k$ which may be referred to as states of nature. The unconditional probabilities $P(A_1), \ldots, P(A_k)$, which may be referred to as prior probabilities, are defined. The mutually exclusive possible events, also sometimes called observable events are $B_1, \ldots, B_m$. Also specified are $P(B_i|A_i)$, which are the conditional probabilities of each observable event given each state of nature. $P(B_i|A_i)$ are also referred to as likelihoods.

The potential for expounding upon the definition of conditional probability and applying it in a more sophisticated manner to address the numerous factors influencing the probability of occurrence of identified software engineering risks is realized with a consideration of the Bayesian effect upon such calculations. Baye's Theorem can be used with statistical sample data and/or experience-based, intuitive probabilistic values (e.g., those derived with Charette's "human analysis and common sense") to update conditional probabilities. This could be effective in applying the previously discussed software engineering project postmortem data to help achieve a more accurate quantification of the probability of occurrence of identified software engineering project risks and is investigated below.

B. APPLYING BAYE'S THEOREM

A consideration of the hypothetical yet plausible and perhaps commonplace scenario of a high profile software engineering project being planned whose success is critical to the future of a software engineering organization will be used to illustrate how

the probability of occurrence of a project risk may be achieved with Baye's Theorem. The software engineering organization's leadership may wish to mitigate software engineer attrition type problems that have occurred on past projects as their impact on the project may have a "ripple effect" on other aspects of the project. This could include adverse effects upon schedule, budget, productivity, etc. More specifically, past software engineering projects may have been put severely behind schedule and also perhaps failed due to software engineering personnel with senior level experience leaving the organization prior to their completion.

The software engineering project's leadership decides to take a more proactive and scientific approach to risk management on this project and create a risk management plan. One of the items outlined in the personnel related risk portion of this plan may recommend that to prevent the same types of schedule slippages that have been caused by the loss of software engineering personnel with senior level experience in the past the organization should immediately and aggressively recruit a replacement for each of these individuals if and when they decide to leave the project.

In order to ascertain how much funding should be allocated for any possible future recruitment effort for the project (monies will be required for advertising, time spent interviewing candidates, etc.) the project's leadership may like to know the probability of occurrence for the risk of a software engineer with senior level experience leaving the project due to a conflict with their supervisor.

The project's leadership decides to use the project post-mortem data related to software engineer attrition that has been collected by the organization over the course of several years through automated and/or other relatively low resource-intensive means (see Section D Collecting and Applying Statistical Data for some discussion of some methods that may be employed to accomplish this). This data is related to several typical aspects of software engineering projects that have been undertaken in the past including the likelihood of a software engineer of a given experience level (junior, intermediate, or senior) leaving the organization after the start of a software engineering project to which they have been assigned. A hypothetical compilation of this information is presented in Table 5 below. The probabilities of occurrence for each software engineer expertise level are referred to as conditional probabilities or random variables.

Table 5. Software Engineer Attrition Example Data

	Software Engineer Level of Experience			
Reason for Employee Leaving Project	Senior	Intermediate	Junior	Total
Conflict with Supervisor	7%	25%	55%	80%
Dissatisfied with Software Engineering Standards and/or Process	13%	20%	25%	60%
Illness or Family-Related Issue	20%	50%	10%	85%
Dissatisfied with Work Load	60%	5%	10%	75%

Suppose that 15% of the staff is classified as having senior level software engineering expertise, 40% are intermediate level, and 45% are junior level. What is the probability that a software engineer with senior level expertise will quit the project due to a conflict with their supervisor?

An application of Baye's Theorem provides:

P(Senior|Conflict with Supervisor) =
P(Conflict with Supervisor|Senior|)P(Senior) / P(Conflict with Supervisor|
Senior)P(Senior) + P(Conflict with Supervisor|Intermediate)P(Intermediate)
+ P(Conflict with Supervisor|Junior)P(Junior)
= (.07)(.15) / ((.07)(.15) + (.25)(.4) + (.55)(.45))
= .03

There is a 3% chance that a software engineer with senior level expertise will leave the project due to a conflict with their supervisor.

This example illustrates how conditional probability may be applied to quantify the probability of occurrence for the identified software engineering risk of a software engineer with senior level experience leaving a project due to a conflict with their supervisor. As has been previously noted, the data used to calculate the quantified probability of occurrence for this hypothetical project risk could have been collected from several software engineering projects over a period of time. It is however also possible to derive the values of conditional probabilities (the Bayesian network nodes that influence

the probability of occurrence of a risk) via Charette's "human analysis and common sense"/experience-based estimates. The use of conditional probabilities that are derived through both methods and combined in a single model is another option.

C. BAYESIAN NETWORKS

Bayesian networks (Jensen, 2001), which may also be referred to as Bayesian Belief Nets (BBNs), or Belief Nets, make applying Baye's Theorem more convenient. Bayesian networks provide a graphical representation of probabilistic models as directed asynchronous graphs (DAGs) whose nodes represent the problem domain's conditional probabilities. This graphical representation aids in creating, maintaining, and understanding large and complex probabilistic models and has been applied successfully in other disciplines such as data mining, medicine, reliability analysis, and real-time weapon scheduling (Neopolitan, 2004).

Another benefit of employing Bayesian networks for solving probability of occurrence type problems is that there are a number of good quality, "industrial strength" tools of commercial/proprietary, free, and open source types. The free GeNIe/SMILE toolset (GeNIe & Smile, 2007) has been evaluated and compared with some other Bayesian network tools and selected to demonstrate the application of a Bayesian network. In addition to being freely available via the Internet and containing features that will often be of use to "real world" software engineers and/or software engineering project managers, the GeNIe/SMILE toolset is compatible with a number of other popular Bayesian network tools (including supporting their file formats), contains good quality tutorial and "help" documentation, and allows statistical data to be imported into Bayesian network models via databases and spreadsheets.

To illustrate the application of Bayesian networks the software engineering attrition example that was discussed previously has been modeled with the GeNIe/SMILE toolset. Figure 1 illustrates the GeNIe/SMILE environment and the completed software engineer attrition model. The images and discussion that follow it provide some details related to creating and using this toolset and the model.

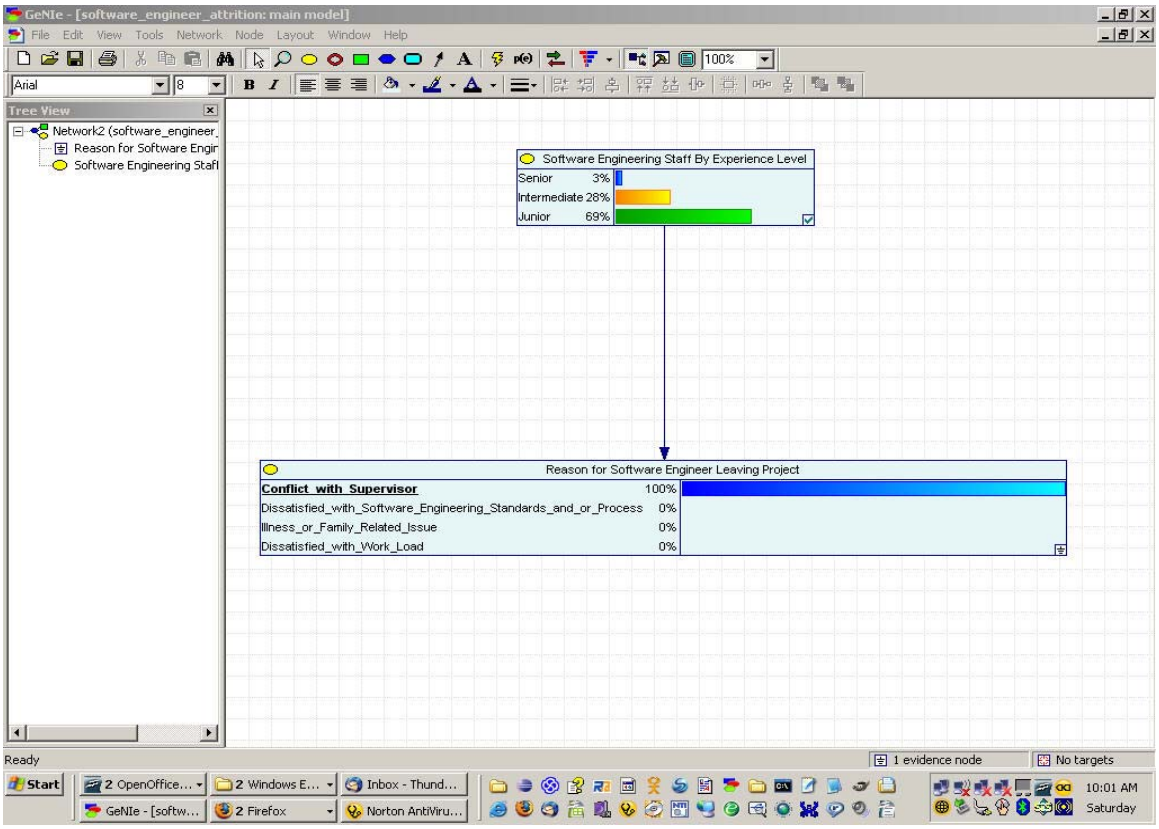

Figure 1. GeNIe/Smile Toolset Environment with the Software Engineer Attrition Bayesian Network Model

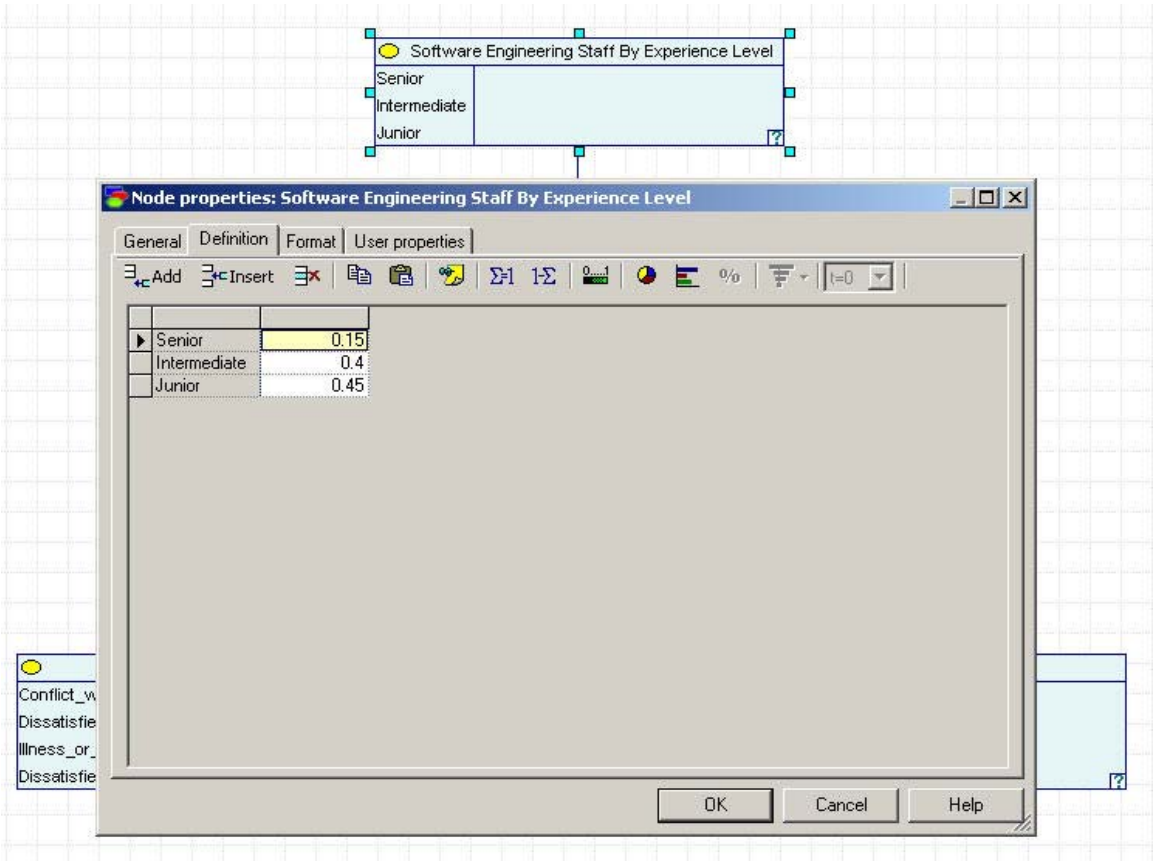

Figure 2. Properties of the "Software Engineering Staff By Experience Level" Node

The three properties of the "Software Engineering Staff By Experience Level" node are shown in Figure 2 above along with their values. Recall that these values are taken from the previous example. The details of how to create a node with the GeNIe/SMILE platform may be found in its "help" documentation.

The four properties of the "Reason for Software Engineer Leaving Project" node are shown in Figure 3 below along with their values. Recall that these values are taken from the previous example and based on the combinations in its table. Other details related how to create a Bayesian network with the GeNIe/SMILE platform, including its nodes, may be found in its "help" documentation. The properties of the "Software Engineering Staff By Experience Level" node are also present as an arc has been created between the two nodes in the model (arcs are described in more detail below).

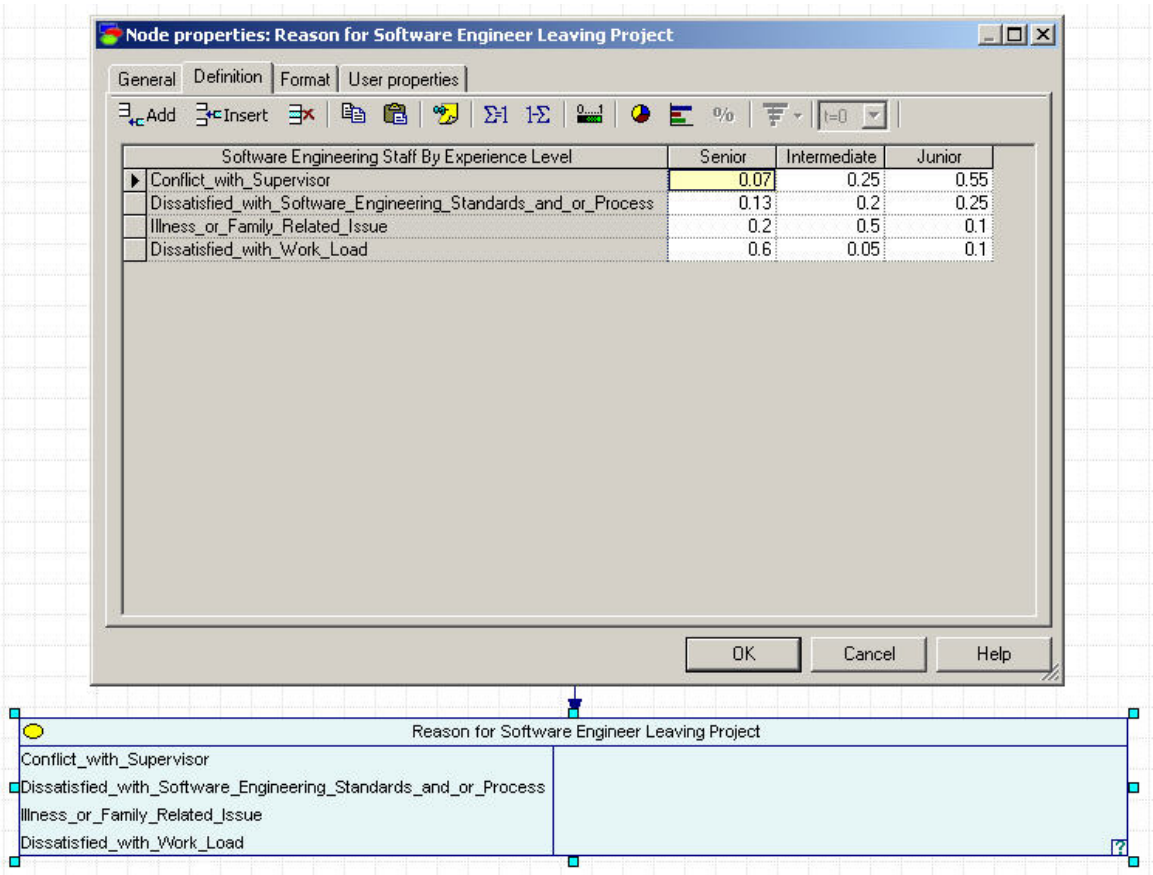

Figure 3. Properties of the "Reason for Software Engineer Leaving Project" Node

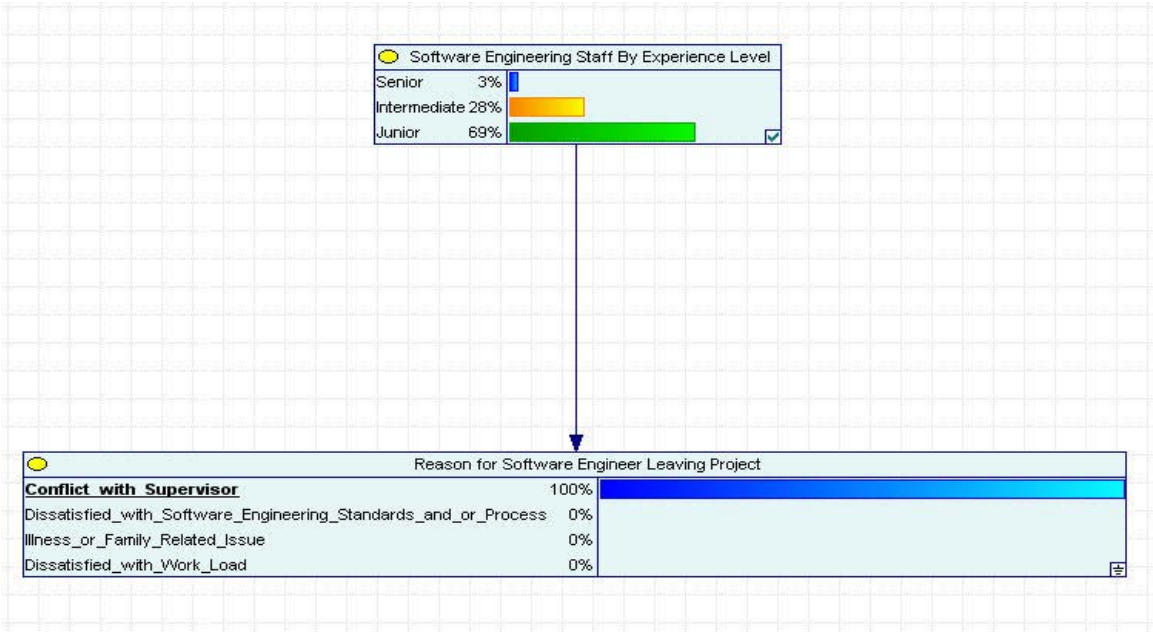

Figure 4. The Completed Software Attrition Bayesian Network Model

The completed model in Figure 4 depicts the completed Bayesian network including its two nodes, “Software Engineering Staff By Experience Level” and “Reason for Software Engineer Leaving Project.” As with the other illustrations, these nodes are displayed as “bar charts” to show their properties. The arc, or arrow (which are referred to as “arcs” in directed asynchronous graphs), from the “Software Engineering Staff By Experience Level” node to the “Reason for Software Engineer Leaving Project” node indicates that a software engineer's level of expertise influences the particular reasons a software engineer leaves a project.

The “Conflict with Supervisor” attribute has a value of 100% as it has been selected as the “evidence” value of the node, or the value that is of particular concern for the probabilistic problem at hand. After setting the node's evidence value, the probabilities for each level of software engineering level expertise leaving a project due to a conflict with their supervisor is displayed by selecting the “Update” command icon. Details related how to create a Bayesian network with the GeNIe/SMILE platform, including the details of how to perform the activities described in this section, may be found in the toolset's “help” documentation.

D. COLLECTING AND APPLYING STATISTICAL DATA

While experience and/or intuition may be used to derive conditional probabilities, “hard” statistical data may often be important for creating the nodes in Bayesian network probability of occurrence models. This is important as it can help ensure that the probabilities of occurrence that are derived with Bayesian networks are based upon objective values and an organization's prior experience with the probabilities of risks which may be more likely to be accurate than those whose nodes are created with intuitive “guesstimates” (e.g., Charette's “human analysis and common sense”). This section discusses collecting and applying statistical data to create Baysian networks.

Spreadsheets and/or databases are commonplace tools that may often be found within software engineering organizations and may be used to collect statistical data. Free, open source varieties such as the Open Office Calc spreadsheet (Calc, 2007), Open Office Base database (Base, 2007), or the MySQL database (My SQL AB :: Developer

Zone, 2007) may be used to help minimize the cost of collecting statistical data. Figure 5 illustrates an example of an Open Office Calc spreadsheet that could be used to collect data for the software engineer attrition example.

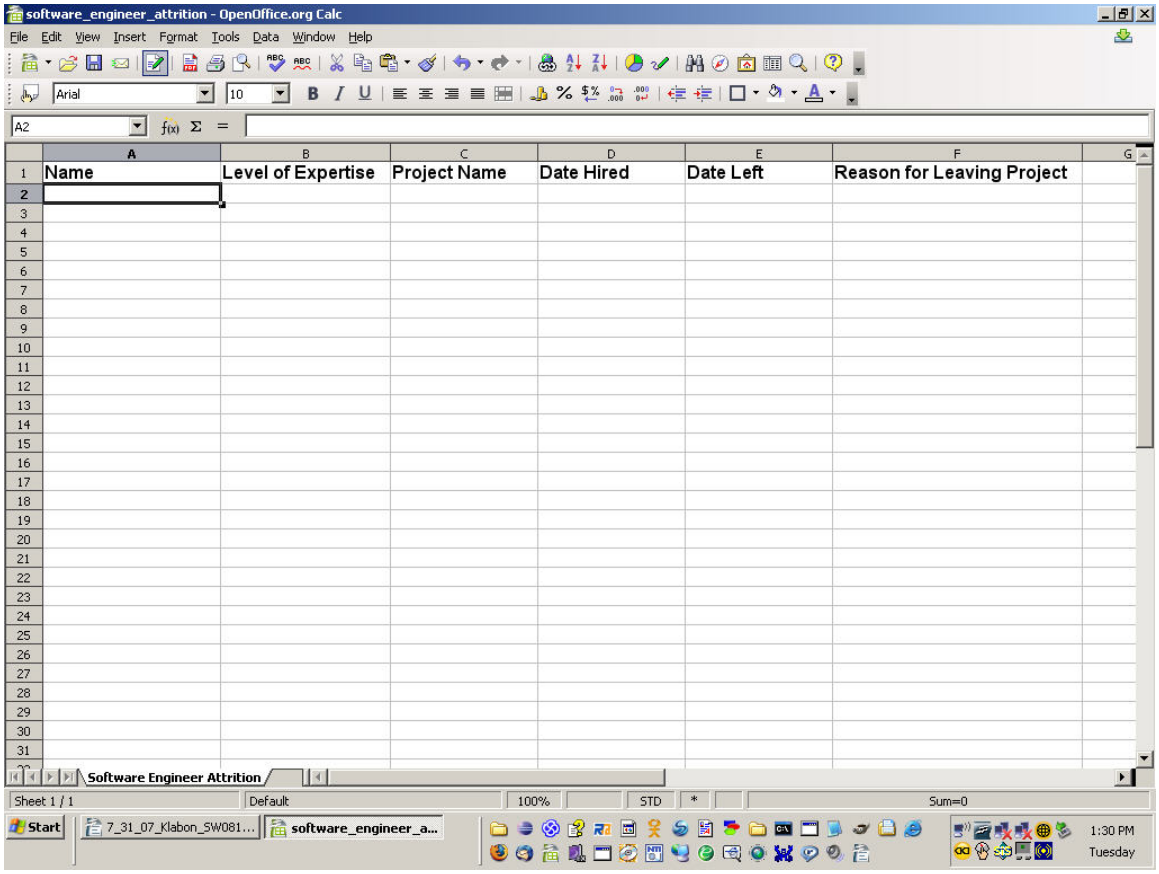

Figure 5. Open Office Calc Spreadsheet That Could Be Used to Collect Data for the Software Engineer Attrition Example

As with software engineering projects in general, it is important to identify, understand, and document the goals and requirements of a risk management program before implementing it. This is especially true of risk management programs' statistical data collection components. It is equally important to document a detailed and precise plan for how these goals and requirements will be satisfied. Failure to thoroughly analyze, understand, and document a risk management program's requirements, including its statistical data requirements, at its inception and approaching these in an ad hoc manner can, and likely will, result in a large amount of wasted time and effort.

Additionally, this could cause the risk management program to (perhaps justifiably) be viewed as wasteful and ineffective and/or fail before it has been afforded sufficient time to mature and realize its potential efficacy.

As a brief example of how a software engineering organization could undertake the collection of statistical data it could first list the software engineering project risks whose probability of occurrence it wishes to quantitatively measure. The next step would be to identify the factors that influence these risks' respective probabilities of occurrence (i.e. the problem domain's conditional probabilities). When undertaking this step it is important to recognize that a conditional probability can influence the probability of occurrence of more than one identified risk. As an example, a software engineer's level of expertise may not only influence their reason for leaving a project. It can also influence other software engineering project risks such as the probability that a project will not meet its deadline or will exceed its budget. By the same token, the probability of occurrence of these risks could also be influenced by the factors that influence software engineers leaving a project. The practical means of collecting data, including the tool(s) that will be used, should also be discussed along with what project stakeholders are responsible for this.

The GeNIe/SMILE toolset aids the application of statistical data by providing support for the creation of Baysian networks with the data contained in several different types of databases and text files (e.g., spreadsheets in .csv format) . Bayesian networks that are derived via this method are referred to as a “Learning” Bayesian networks and the GeNIe/SMILE toolsets' functionality could be used exclusively, or in conjunction with, “manual” methods for building probability of occurrence models (e.g., the method used for creating the previous software engineer attrition Bayesian network).

As an example, a Bayesian network's nodes may be derived with the columns in a spreadsheet. The rows of the spreadsheet represent the combinations of the states that exist between the nodes. The toolsets' help documentation contains some more detailed information on this and includes some tutorials on how to undertake the creation of Learning Bayesian networks.

Finally, while “hard” statistical data can be quite valuable it is important to recognize that conditional probabilities that are derived through this means can, and sometimes should, be revised with Charette's “human analysis and common sense” during the course of a software engineering project if needed. As an example, if there is a local influenza outbreak during the course of a software engineering project it may be wise to increase the probability of software engineers leaving due to an “illness or family related issue” by an amount that “seems about right.” In other words, the impact of the influenza outbreak can be intuitively judged to estimate how much its influence will increase the probability of software engineers leaving the project even though no statistical data that can provide insight into this has been collected.

IV. CONCLUSIONS

Bayesian networks may be used to quantitatively measure the probability of occurrence of any type of identified software engineering project risk and could be an important part of the "way forward" in the improvement of this measurement within the field of software engineering. Bayesian networks allow all of the factors that are deemed to influence the probability of occurrence of a risk to be accounted for along with their respective degrees of influence. The probabilities of occurrence of conditional probabilities may be derived with the use of "hard" statistical data and/or Charette's "human analysis and common sense." When experience and/or intuition is used to derive the probabilities of occurrence for Bayesian network node properties, a type of sharing of this experience and/or intuition in a quantitative way that should be relatively easy for others to understand and is not necessarily entirely dependent upon the individual culture of a particular software engineering project may occur.

The practical matter of having a convenient means of applying Bayesian networks to quantify the probability of occurrence of risks within the typically tight time and/or cost constraints that exist on many software engineering projects is largely, if not entirely, solved by the fact that there is at least one good quality, free toolset that will likely meet the "real world" risk management needs of most software engineering projects (GeNIe/SMILE). Other Bayesian network tools should, however, be evaluated and compared to ensure that they meet users' needs.

Bayesian networks seem to be an improvement upon the way that the measurement of the probability of occurrence of software engineering project risks is typically undertaken. While worthy of serious consideration, it is important to recognize that they are not a Brooksian "silver bullet" (Brooks, 1995) or "magical solution" to this problem. As with all statistical and probabilistic methods, the principle of "garbage in, garbage out" (or "GIGO") applies to the use Bayesian networks. As an example, if the conditional probabilistic values that are used in a Bayesian network are inaccurate it is not likely that a valuable probability of occurrence will be derived with them. It is also

important to recognize that some practice and review of tutorials may often be necessary before modeling “real world” problem domains with Bayesian networks as their application is not necessarily intuitive or trivial.

There are methods of applying Bayesian networks that are more sophisticated than what has been presented within this work's somewhat limited, investigative scope and some in-depth discussion and citation of these may be found in (Neopolitan, 2004). Finding ways to “borrow” established techniques from other applications of Bayesian networks is worthy of additional investigation as it could advance the measurement of the probability of occurrence of software engineering project risks with a relatively minimal investment of time and effort.

LIST OF REFERENCES

"risk." Merriam-Webster's Collegiate Dictionary, Tenth Ed. 1998.

Base. (n.d.). Retrieved July 31, 2007 from http://www.openoffice.org/product/base.html.

Boehm, B. W. (1991). Software Risk Management: Principles and Practices. IEEE Software, 8(1):32-41.

Boehm, B. W., DeMarco, T. (1997). Software Risk Management. IEEE Software, 14(3):17-19.

Brooks, F.P. Jr. The Mythical Man-Month, Essays on Software Engineering. USA: Addison-Wesley, 1995.

Calc. (n.d.). Retrieved July 31, 2007 from http://www.openoffice.org/product/calc.html.

Carr, M. J., Konda, S. L., Monarch, I., Ulrich, F. C., and Walker, C. F. (1993). Taxonomy-Based Risk Identification. Technical Report CMU/SEI-93-TR-6, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania 15213, USA.

Charette, R. N. (1991). The Risks with Risk Analysis. Communications of the ACM, 34(6):106.

Chittister, C., Kirkpatrick, R., and Van Scoy, R. (1993). Risk Management in Practice. Technical Review SEI-93-TechReview-004, Software engineering Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania 15213, USA.

Conrow, E. H., Shishido, P. S. (1997). Implementing Risk Management on Software Intensive Projects. IEEE Software, 14(3):83-89.

Fairley, R. (1994). Risk Management for Software Projects. IEEE Software, 11(3):57-67.

Freimut, B., Hartkopf, S., Kaiser, P., Kontio, J., Kobitzsch, W. (2001). An Industrial Case Study of Implementing Software Risk Management. ACM SIGSOFT Software engineering Notes, 26(5):277-287.

GeNIe & SMILE. (n.d.). Retrieved July 31, 2007 from http://genie.sis.pitt.edu.

Gluch, D. P. (1994). A Construct for Describing Software Development Risks. Technical Report CMU/SEI-94-TR-14, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania 15213, USA.

Hall, E.M. Managing Risk, Methods for Software Systems Development. USA: Addison-Wesley, 1998.

Higuera, R. P., Gluch, D. P., Dorofee, A. J., Murphy, R. L., Walker, J. A., and Williams, R. C. (1994). An Introduction to Team Risk Management (version 1.0). Special Report CMU/SEI-94-SR-1, Software engineering Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania 15213, USA.

Higuera, R. P., Haimes, Y. Y. (1996). Software Risk Management. CMU/SEI-96-TR-12, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania 15213, USA.

Hyatt, L., Rosenberg, L. (1996). A Software Quality Model and Metrics for Identifying Project Risks and Assessing Software Quality. NASA Software Assurance Technology Center, Presented at the 8th Annual Software Technology Conference, Utah, USA, Retrieved July 31, 2007 from http://satc.gsfc.nasa.gov/support/STC_APR96/qualtiy/stc_qual.html.

Jensen, F.V. Bayesian Networks and Decision Graphs. USA: Springer-Verlag, 2001.

Keil, M., Cule, P. E., Lyytinen, K., Schmidt, R.C. (1998). A Framework for Identifying Software Project Risks. Communications of the ACM, 41(11):76-83.

Kitchenham, B., Linkman, S. (1997). Estimates, Uncertainty, and Risk. IEEE Software 14(3):69-74.

Kontio, J., Englund, H., Basili, V. (1996). Experiences from an Exploratory Case Study with a Software Risk Management Method. Technical Report UMIACS-TR-96-75, Institute for Advanced Computer Studies and Department of Computer Science, University of Maryland, College Park, Maryland 20742, USA.

Leveson, N. G. SAFEWARE*:* System Safety and Computers. Reading, Mass.: Addison-Wesley, 1995.

MySQL AB :: Developer Zone. (n.d.). Retrieved July 31, 2007 from http://www.mysql.org.

Neopolitan, R.E. (2004). Learning Bayesian Networks, First Edition. Prentice Hall, Upper Saddle River, New Jersey 07458, USA.

Ott, L.O. (1993). An Introduction to Statistical Methods and Data Analysis, Fourth Edition. Wadsworth Publishing Company, Belmont, California 94002, USA.

Software Technology Support Center (2005). Understanding Risk Management. Crosstalk, 18(2):4-7.

Van Scoy, R. L. (1992). Software Development Risk: Opportunity, Not Problem. Technical Report CMU/SEI-92-TR-30, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania 15213, USA.

Williams, R. C., Pandelios, G. J., Sandra, S. G. (1999). Software Risk Evaluation (SRE) Method Description (Version 2.0). Technical Report CMU/SEI-99-TR-029, Software engineering Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania 15213, USA.

THIS PAGE INTENTIONALLY LEFT BLANK

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California

3. Professor John Osmundson
   Naval Postgraduate School
   Monterey, California

4. Paul Kimmerly
   USMC Technology Services Organization
   Kansas City, Missouri

5. Professor Man-tak Shing
   Naval Postgraduate School
   Monterey, California

6. Professor Richard Riehle
   Naval Postgraduate School
   Monterey, California