SANDIA REPORT
SAND2001-3317
Unlimited Release
Printed October 2001

Hybrid Processing of Measurable and Subjective Data

J. Arlin Cooper

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185 and Livermore, California 94550

Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy under Contract DE-AC04-94AL85000.

Sandia National Laboratories

SAND 2001-3317
Unlimited Release
Printed October 2001

Hybrid Processing of Measurable and Subjective Data

J. Arlin Cooper
Airworthiness Assurance Department
Sandia National Laboratories
P.O. Box 8500
Albuquerque, NM 87185-0490

Robert J. Roginski
Consultant, Retired from Sandia National Laboratories

Key words: safety analysis, fuzzy mathematics, possibilistic mathematics, hybrid analysis

Abstract

Conventional systems surety analysis is basically restricted to measurable or physical-model-derived data. However, most analyses, including high-consequence system surety analysis, must also utilize subjective information. In order to address this need, there has been considerable effort on analytically incorporating engineering judgment. For example, Dempster-Shafer theory establishes a framework in which frequentist probability and Bayesian incorporation of new data are subsets. Although Bayesian and Dempster-Shafer methodology both allow judgment, neither derives results that can indicate the relative amounts of subjective judgment and measurable data in the results. The methodology described in this report addresses these problems through a hybrid-mathematics-based process that allows tracking of the degree of subjective information in the output, thereby providing more informative (as well as more appropriate) results.

In addition, most high consequence systems offer difficult-to-analyze situations. For example, in the Sandia National Laboratories nuclear weapons program, the probability that a weapon responds safely when exposed to an abnormal environment (e.g., lightning, crush, metal-melting temperatures) must be assured to meet a specific requirement. There are also non-probabilistic DOE and DoD requirements (e.g., for determining the adequacy of positive measures). The type of processing required for these and similar situations transcends conventional probabilistic and human factors methodology. The results described herein address these situations by efficiently utilizing subjective and objective information in a hybrid mathematical structure in order to directly apply to the surety assessment of high consequence systems. The results can also improve the quality of the information currently provided to decision-makers. To this end, objective inputs are processed in a conventional manner; while subjective inputs are derived from the combined engineering judgment of experts in the appropriate disciplines. In addition to providing output constituents (including portrayal of uncertainty) corresponding to combination of these input types, their individual contributions to the resultant uncertainty are determined and provided as part of the output information. Finally, the safety assessment is complemented by a latent effects analysis, facilitated by soft-aggregation accumulation of observed operational constituents.

Acknowledgment

Many people contributed to the results reported, through technical discussions, suggestions on reporting and presentation, and through funds-supporting efforts. Thanks to Perry D'Antonio, Dick Perry, Kathleen Diegert, John Covan, Ken Chen, and Mike Dvorack, all of Sandia National Laboratories, Scott Ferson, Applied Biomathematics, and Profs. Tim Ross, University of New Mexico, Don Wunsch, University of Missouri, George Klir, State University of New York at Binghamton, and Jim Luxhφj (Rutgers University). Programming was by Bob Roginski (consultant, retired from Sandia), and Tony Zimmerman (Gram, Inc.). Funding was provided by the U. S. Department of Energy through a Laboratory Directed Research and Development award and by the Federal Aviation Administration Aviation Safety Risk Analysis Section, Airport and Aircraft Safety Research Engineering and Development Division.

Contents

List of Figures

Summary of Milestones

There were three phases to the work, each corresponding to a fiscal year of LDRD (Laboratory Directed Research and Development) support. During the first fiscal year, development of specific mathematical tools that were considered candidates for possible incorporation into a complete coordinated software package was emphasized as the first phase of the work. Specific milestones were:

- ✓ The methodology for the contest (competing failure characteristics) tool was scheduled for completion on December 1, 1998. The completion date was met, but refinements in possibilistic processing continued until April 2001. The utility of the contest tool is to assess related and competing system failures (races).
- ✓ The fuzzy logic tool was scheduled for completion (and was completed) on March 1, 1999. This tool is considered useful where failure is dominated by the weakest component in a series of components or support is furnished by the most robust component in a parallel structure.
- ✓ The dependency tool was scheduled for completion (and was completed) on May 1, 1999. The utility of this tool is that it transcends conventional correlation analysis to account for subjective and possibilistic dependence.
- ✓ The constrained mathematics capability was scheduled for completion (and was completed) on July 1, 1999. This improved the accuracy over conventional techniques in portraying subjective uncertainty.
- ✓ A workshop was projected during September 1999, in order to formally brief the methodology developed to that point. It was carried out at the International System Safety Conference, August 1999.
- ✓ A peer-reviewed SAND report was scheduled for completion (and was completed) on September 30, 1999. The report addressed the initial Markov modeling portion of the LDRD effort [Ref. 1]. The Markov effort continued, and an updated report was issued in September 2001 [Ref. 2].

The second phase of the project emphasized selection of the most appropriate tools for the end-product software suite, and determination of the best implementation strategy.

- ✓ The tool that modifies Dempster-Shafer methodology to track the amount of subjectivity incorporated was scheduled to be complete by December 1, 1999, and was completed on time.
- ✓ The modified Bayesian analysis tool was scheduled for completion on March 1, 2000. A methodology was in place on schedule, but its applicability to the family of problems for which it was applied was not considered compelling, so it was omitted from the suite of approaches implemented in software. The methodology developed is described in this report. The intent of the tool was to improve previous assessments by incorporating new information in a Bayesian manner, but to add the possibilistic capability that more accurately represents subjectivity in a hybrid methodology [Ref. 3].
- ✓ The implication tool was scheduled for completion (and was completed) on May 1, 2000. This tool was to incorporate forms of implication that have recently been

developed for “Boolean-like” (soft-modified “and” and “or”) functions [Ref. 4]. Its applicability was not judged sufficient for it to become part of the software suite.

- ✓ The soft aggregation tool was scheduled for completion (and was completed) on July 1, 2000. An improvement was found, and it was modified in late CY 2000. This tool allows nonlinear aggregation of contributing factors to risk, as for example, those implicated in the ValuJet 592 accident.
- ✓ A second workshop was scheduled to be held in August or September 2000. It was conducted at the International System Safety Conference in September 2000.
- ✓ A second interim report was scheduled to be completed by September 30, 2000. In place of the report, a paper [Ref.5] was published in the Proceedings of the International System Safety Society Conference, September 2000.
- ✓ The expert elicitation tool was to be complete by September 30, 2000. It was completed on schedule. This provides the capability to capture expert judgment in a complete and consistent manner so that the results can be expressed as possibilistic mathematical functions.

The third phase of the project was aimed at finalizing tool development and completing hybrid combination of the tools into an end-product software package, named COSMET (Coordinated Objective/Subjective Mathematically Enhanced Tools).

- ✓ The multi-objective decision tool was to be complete by December 15, 2000. This tool was to provide a possibilistic weighting of factors contributing to a decision (e.g., judgment of the robustness of a system surety[1] program). The intent was also to contribute to systematically understanding various decisions that can be reached by various people who all have access to the same information. The targeted effort was subsumed by the Markov Tool effort, which greatly enriched the intended capabilities, and this was completed in September 2001 [Ref. 2].
- ✓ The hybrid combination methodology was to be complete by July 1, 2001. This was one of the most formidable tasks in the project, because the hybridization involved all tools developed, as well as possibilistic and probabilistic hybrid analysis. It was completed August 1, 2001.
- ✓ A final workshop was to be held in August or September 2001. The workshop was proposed for a conference taking place in September, but was accepted only as an alternate. A firm commitment for a similar workshop was obtained from the Society for Risk Analysis and will be held December 2, 2001.
- ✓ A final peer-reviewed SAND report was to be complete by September 30, 2001. There are actually two reports that will satisfy this milestone: this one and Ref. 2.

Thrust Areas for the Work

The major aims of the project as originally proposed for LDRD funding in 1998 are summarized here. These aims changed very little during the three-year project.

- ✓ The methodology is differentiated from existing (more conventional) methodologies by its unique handling of subjectivity through various possibilistic tools. The main

[1] The term surety is intended to encompass safety, security, and reliability.

goal associated with incorporating subjective inputs is to provide a firm mathematical basis for doing so. The approach supplements and partially incorporates conventional methodology, but depends on a hybrid analysis structure.

- ✓ The approach coherently and seamlessly integrates conventional methodology and a significant number of new tools in a hybrid analysis. This allows incorporation of new capabilities in a new software tool (COSMET) that improve on, without losing any of the advantages of, conventional methodology.
- ✓ It promotes validated models for sure system design by making a variety of mathematics-based tools available for customizing to various applications. Some of the potential applications are in the nuclear weapons surety assessment areas; air and rail transportation surety; and critical infrastructure (e.g., power, water) surety.
- ✓ The methodology is an appropriate and defensible way to assess human interactions with complex systems, where humans can be sources of inputs to systems or can be system assessors. In both cases, there is a need to incorporate subjectivity and uncorrelated dependence in describing effects analytically.
- ✓ The multi-object decision analysis tool, which was subsumed by the Markov Latent Effects Tool [Ref. 2], contributes to optimal surety strategies. This is an essential capability, because it isn't sufficient to limit assessment to a physical system without also addressing its risk management.

Potential Payoffs

Some indications of the potential payoffs of the work have been documented over the past three years.

- ✓ The large number of professional analysts (approximately 165 in 17 countries) who have requested the COSMET software and information about the methodology behind it demonstrates the interest and potential applications in many fields for the methodology[2].
- ✓ The (nontrivial) disjoint set algorithm we developed has also been specifically sought out by researchers. As one example, a software risk analysis package was found that incorrectly performed the disjoint set calculation. After this was pointed out to the author, he requested the LDRD-developed algorithm in order to remove the error from his software.
- ✓ The approach to constrained mathematics was cited by George Klir, Distinguished Professor of System Science at the State University of New York, as having the potential for various applications during a plenary address at an information-processing conference. At two other conferences, errors were found in papers presented by researchers whose work was based on unconstrained mathematics. Two journal articles have been published on this subject [Refs. 6, 7].
- ✓ An invalid "maximum entropy proof" presented by a researcher at a major international conference was identified because of the work on this project, and a method for addressing the error was demonstrated.
- ✓ Professor Tim Ross, UNM, wrote a letter to SNL personnel in support of this project. In the letter, he cited applications of the work at the Army Environmental Policy

[2] A list of these requestors is available.

Institute and at UNM, where two students are following up on the work, and it was also the basis for a study proposal he made to the National Science Foundation. This exemplifies the research synergism the project has created.

Description of the Overall Strategy

Most safety and security analysts first learn conventional probabilistic risk assessment approaches through problems analogous to flipping fair coins; drawing from well shuffled, complete card decks; picking balls out of evenly distributed collections in urns; and other objective problems, along with probability-distribution-based representation of uncertainty. Then these approaches are often applied to real-world situations for which the conventional methodology is a misapplication[3]. Since practical safety and security problems have relatively little objectivity, one must question the general lack of attention to the role of subjectivity. What is the probability of tossing "heads" with a coin that can be observed to be slightly bent? Is it more important to worry about averages or extremes? What about situations that have dependence (transcending correlation) that can only be identified subjectively? Can probability distributions accurately represent subjective uncertainty? These questions were addressed in this project.

The genesis of this project was that methodology for treating subjective uncertainty was essential, and it was clearly not appropriate to apply objective analysis to subjective phenomena. The literature contains many pro and con arguments that were made as this subject developed, but there isn't much doubt anymore that subjective uncertainty phenomena are best matched to subjective analysis methodologies. That was key to the approach taken in this project. The situation is represented in Fig. 1.

The "Conventional" Process: Conventional software uses data and models to produce output and probability distributions to describe both variability and uncertainty (non-data-based judgment).

The Hybrid Process: Uses conventional methodology, except that it replaces probabilistic uncertainty with engineering expertise about uncertainty. This is more applicable to ill-defined processes because it does not unjustifiably suppress extremes.

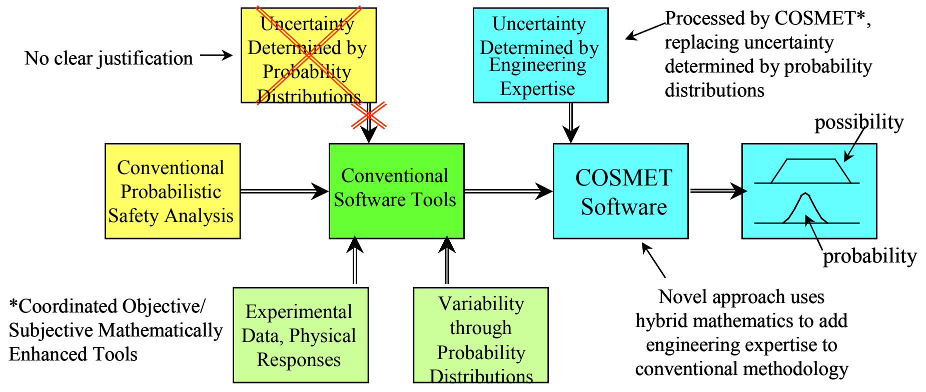
No clear justification
~~Uncertainty Determined by Probability Distributions~~
Uncertainty Determined by Engineering Expertise
Processed by COSMET*, replacing uncertainty determined by probability distributions
possibility
Conventional Probabilistic Safety Analysis
Conventional Software Tools
COSMET Software
probability
*Coordinated Objective/ Subjective Mathematically Enhanced Tools
Experimental Data, Physical Responses
Variability through Probability Distributions
Novel approach uses hybrid mathematics to add engineering expertise to conventional methodology

Figure 1. Incorporating Subjective Methodology for Subjective Uncertainty

[3] Examples will be shown.

The basic idea is that objective analysis techniques are appropriate for objective variability, but not for subjective uncertainty. The latter requires subjective analysis methodology. Where objective and subjective components co-exist[4], a hybrid analysis can be used to combine the two approaches into a coordinated assessment.

One of the traps inherent in thinking exclusively in terms of probability distributions is that the assumption of any particular distribution presupposes that the system behavior is in accordance with the distribution. Following is a demonstration that it is dangerous to assume that more information is known than the available data warrant. There are three (objective) processes in the demonstration, all of which produce numbers as outputs. The output for process "A" has a mean value of $3\frac{2}{3}$, and a range of 1 (minimum) to 5 (maximum). The output of process "B" has a mean value of $3\frac{1}{6}$, and a range of 3 (minimum) to 4 (maximum). The output of process "C" has a mean value of $3\frac{1}{3}$, and a range of 2 (minimum) to 6 (maximum). If a pair of the processes were "opponents" in a game of chance decided by high score, is the data given sufficient to decide on the most probable winner? Would A be favored over C because of its greater mean value? Would C be favored over B? Would the transitive property hold? For the processes shown in Fig. 2, the answer to all four questions is "No"!

A (3 2/3 ave.) beats B (3 1/6 ave.) 2/3 of the time
B (3 1/6 ave.) beats C (3 1/3 ave.) 2/3 of the time
C (3 1/3 ave.) beats A (3 2/3 ave.) 5/9 of the time

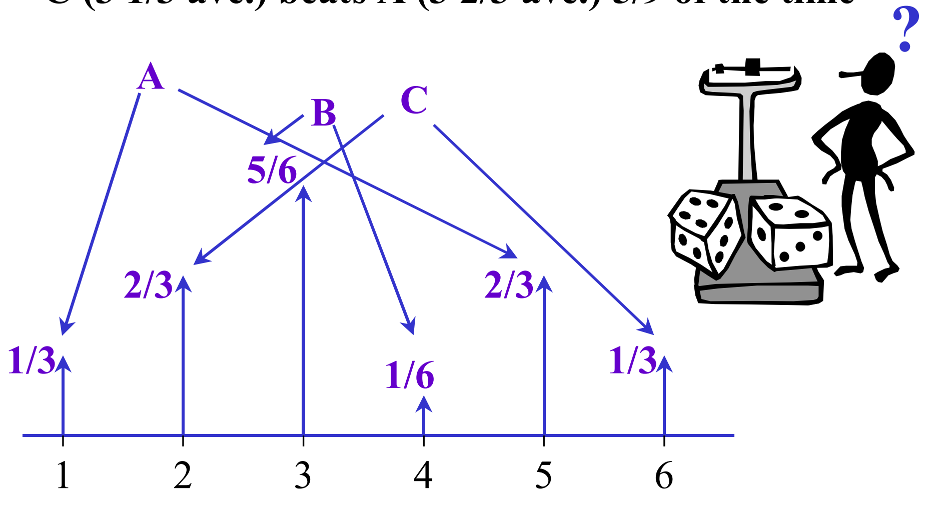

Figure 2. Three Processes Exemplified by Non-Transitive Dice

[4] A straightforward example is determination of the probability of throwing a large number with a fair die.

Anyone thinking that any of the four answers should be "Yes" has consciously or unconsciously added unwarranted assumptions to supplement the data available, possibly visualizing probability density functions, each with most of the weight in the vicinity of the mean. The dice-oriented processes shown in Fig. 2 have probability distributions, but they are (to most people, for the data given) of unexpected form. The corresponding implementations can use (for example) "fair" dice (six equally likely outcomes) with non-traditional number sets (two ones and four fives for A; five threes and one four for B; and four twos and two sixes for C). Subjectivity is inherent in this objective problem, because the detailed description of the processes is not known *a priori*. The subjective-oriented tools developed in this project actually discourage making unwarranted assumptions by preventing the use of more information than the available data offer.

Output Constituents

Three types of metrics concerning system surety were emphasized in this project and form the basis for the output modes. Two of these are related to the probability of system failure; one an objective measure based on measured data, and the other a subjective measure based on expert engineering judgment. The third is a measure of organization robustness with respect to surety, and is somewhat analogous to a "grade" that a student might receive in a class. Details on these three types of information are given in the following sections.

<u>Objective Measure of Probability of System Failure.</u> Objective data can help give information about the probability of system surety failure. Since it is common to have some objective variability associated with systems, the probability of failure can also be variable. This leads to the concept of a probability distribution to describe the variability in probability of failure. Most conventional risk analysis methods give a result similar to objective probability of system failure, representing variability and uncertainty by probability distributions. This concept is used in this project for one constituent of output, but is limited to objective variability.

<u>Subjective Measure of Probability of System Failure.</u> Subjective measures are almost always necessary in order to supplement objective measures. This requires expert judgment solicitation, which is converted to fuzzy or possibilistic measures of uncertainty about the probability of system surety failure. The possibility of uncertain failure probabilities is a second constituent of output. Possibilistic probability was used rather than direct possibility in order to better match purely objective measures as well as potential requirements. The hybrid analysis that links objective probability and subjective probability determines the amount of subjectivity, so that output displays of the two are linked by the proportionate amount of subjectivity and objectivity inherent in the results. The ability to generate these types of outputs was originally implemented in the PHASER (Probabilistic Hybrid Analysis System Evaluation Routine) software [Ref. 8], and was enriched in the COSMET software [Ref. 9]. This hybrid capability was a key feature of the overall project.

Measure of the System Organizational and Operational Surety Status. It would be very unlikely that system surety could be independent of the manner in which the system was operated, the restrictions placed on that operation, and the manner in which the operation was subjected to periodic surety reviews. These factors can be determined through a collection of metrics that form the basis for an "inspection" activity and transcend the measures of failure probability to give indicators about the surety health of the system operation. The aggregation of the data collected by inspection give a system surety "grade," and can also lead to useful information about how to improve the system and how to gain efficiency in the inspection activity. The basis for this approach was demonstrated in the Markov Latent Effects Tool software [Ref. 2], which became part of the COSMET suite. This type of information is the third output constituent.

Input Modes

There are several types of inputs that are used by COSMET. These are described briefly in the following sections.

Possibilistic/Fuzzy Inputs

Possibilistic/fuzzy inputs provide "zero" or "one" ordinate values at one to four specific abscissa positions. The actual entries specify the abscissa positions, from which the ordinate values are deduced by the software. A single abscissa entry is a scalar equal in value to the abscissa value (i.e., a delta function reaching ordinate value "one" at the entered abscissa value. A pair of entries defines an interval ranging from the first abscissa value (lower bound) to the second (upper bound). The resultant function transitions from ordinate value "zero" to ordinate value "one" at the lower bound, and returns to "zero" at the upper bound. A trio of entries results in a triangular function, with the base of the triangle beginning at the lowest abscissa value, linearly increasing to ordinate value "one" at the second abscissa value, and linearly decreasing back to "zero" at the third abscissa value. Four entries specifies a trapezoid, with the base of the trapezoid beginning at the first value, linearly increasing to ordinate value "one" at the second abscissa value, maintaining ordinate value "one" until the third abscissa value, and decreasing linearly to "zero" at the fourth abscissa value.

Possibilistic/Fuzzy Elicitation from Multiple Experts

The COSMET software enables combined judgment from multiple experts by allowing entries of multiple possibilistic/fuzzy inputs, each associated with an input sequence position and each associated with a "weight," where the sum of the weights for all involved experts must be one. Identification of the experts' names can be made in comment fields, but these are not provided as outputs. Where weights are not entered, the software assigns each weight the value $1/n$, where n is the number of experts. A subjectivity entry is associated with the inputs from each expert. This allows conversion onto an ordinate scale representing "level of presumption." The values allowed for are computed as a weighted average of the intermediate abscissa values entered by (or on behalf of) the experts.

Probabilistic Inputs

Since COSMET uses LHS (the Sandia National Laboratories Latin Hypercube Sampling routine), the probabilistic inputs are in the format allowed for by LHS (e.g., a mean and standard deviation for a Gaussian distribution).

"Inspection" Inputs

The Markov portion of COSMET utilizes possibilistic/fuzzy inputs corresponding to a "grade" or "score" with abscissa values between (and including) "zero" and "one."

Logic Equation Inputs

There are various possible sources of logic equations that can be utilized by COSMET. For example, equations corresponding to fault trees or event trees can be used. This is the basic structure used for probabilistic calculations. Logic equations can also be entered for the "Early Alert" portion of the Markov Tool.

Synopsis of Tools

This section provides a summary of the strategy behind the various tools in the COSMET tool set. Details have been reported previously for all except the last (Modified Bayesian Tool), so it is the only one for which complete mathematical treatment is given.

Contest Tool

The Contest Tool is intended to assess the probability that a "race" condition will be lost. For example, this is applicable to the assessment of the potential of a strong link (safety-protective component) failing prior to a weak link (safety-operation-critical component). Conventional probabilistic calculus solutions use the difference of probability density functions, along an abscissa representing a stressor to the race constituents, in order to derive a single-valued (no variability) number. Modifications to this conventional solution have been made to introduce uncertainty into the parameters of the constituent probability functions, but this is artificial where the probability distribution types are not certain (the most common situation). COSMET solves this problem by differencing possibilistic functions for the race constituents, where the possibilistic functions indicate the subjective uncertainty about the failure regime. The tool is applicable to any stressor entered by the user. The possibilistic result is converted into probabilistic terms, but the uncertainty about the result is preserved.

Extreme Minmax Tool

Safety assessment is usually predicated on considering "credible" inputs. Since accident histories demonstrate numerous cases of "incredible" inputs occurring, there is almost always a discomfort that something has been missed. COSMET allows for a composite overlay of inputs that may be sources of some worry, although thought to be incredible.

Fuzzy Logic Tool

The fuzzy logic tool (called in COSMET the "Link Minmax Tool") treats subjectively-dominated failure information about weak components in a structure of series components, any of which can degrade safety, and strong components in a structure of parallel components that support safety. The methodology is applicable to series components such a shift registers and parallel components such as redundant functions. This is analogous to weak-link failure in a chain of series links, and strong-link support in parallel chains.

Modified Dempster-Shafer Processing

The COSMET version of this approach allows users to enter interval or possibilistic estimates of uncertainty about a process (e.g., the yield rate of components on a production line) based on objective data available and subjective estimates about the prognosis. Also entered is the amount of relative subjectivity involved in the estimate. COSMET calculates possibilistic and probabilistic values based on this information.

The Markov Latent Effects Tool

The utility of the Markov Latent Effects Tool is to "grade" a process related to safety support. For example, an operation that has a satisfactory physical assessment (low probability of physical failure), but which is not being monitored by continuous assessment and is not supported by a strong safety culture, does not have a good safety prognosis. Also included are "Early Alert" warnings and Importance and Sensitivity pointers.

Modified Bayesian Tool

The Modified Bayesian Tool is not currently part of the COSMET suite. However, the mathematical methodology and software development for the tool is complete. The basic idea is that an objective and/or subjective base of information could be available, and this base could be modified in a Bayesian sense following the accumulation of additional information. While possibilistic/fuzzy information updates are inherently Bayesian and can be described subjectively, probabilistic Bayesian updates can be treated with conventional probabilistic mathematics.

There are two options for the possibilistic part: 1) The user can enter a possibilistic number (trapezoid, triangle, square, point), or 2) the user can accept a default. The default is 10^{-9}, 10^{-3}. The object is that the user might not have any idea when first starting, and might want to accept the default. After evidence begins to accumulate, the user would want to modify the startup value accordingly. The accumulated evidence is objective, but this tends to modify the subjective estimate used in the first part.

The second part is probabilistic. There are also two options for getting started on the probabilistic part: 1) The user can enter a probability density function in a prescribed form (see below), or 2) the user can accept a default.

The entry process for each of the two above parts has logarithmic abscissa values starting at 10^{-10}, and running to 10^{-0}. The ordinate values are called A_k. The default A_k values are 0 for abscissa values less than 10^{-9} and for abscissa values greater than 10^{-3}. They are 1/6 for all other *k*.

One other user entry needed is the "evidence" (This is the reason for the modified Bayesian Tool). The evidence is of the form of *x/y* (*x* failures in *y* trials).

For each abscissa value, the evidence is used to assess the binomial probability: $P_k = \dfrac{y!}{x!(y-x)!} p_k{}^x (1-p_k)^{y-x}$.

Then the Bayesian calculation is made:

$$Q_k = \frac{P_k A_k}{\sum_i P_i A_i}.$$

Next replace each A_{k} with the corresponding Q_{k}. This is the probabilistic Bayesian update.

Example Problem

The application of a combination of COSMET tools to a particular example problem helps illustrate the overall strategy. Consider safety-critical information that is used when the system is required to operate safety switches, but which is to be denied from the system when operation would not be safe (analogous to entering a code in an electronic lock and then requiring its erasure). The threat is that remnant switch position or information could compromise safety of the system. A fault tree is shown in Fig. 3.

The left-most branch of fault tree represents safety-switch testing, where the critical information has been inserted and for which the action must be reset. The first or-gate accounts for human failure to initiate the reset (Event A) or equipment failure to respond properly to the human-entered command (Event B). Event A is a possibilistic event and Event B is the result of a race between the human command and the equipment response. If the operation is to be checked under administrative procedures, this check must also fail (Event C). Event C is determined by utilizing the Dempster-Shafer modified approach with inputs from two experts.

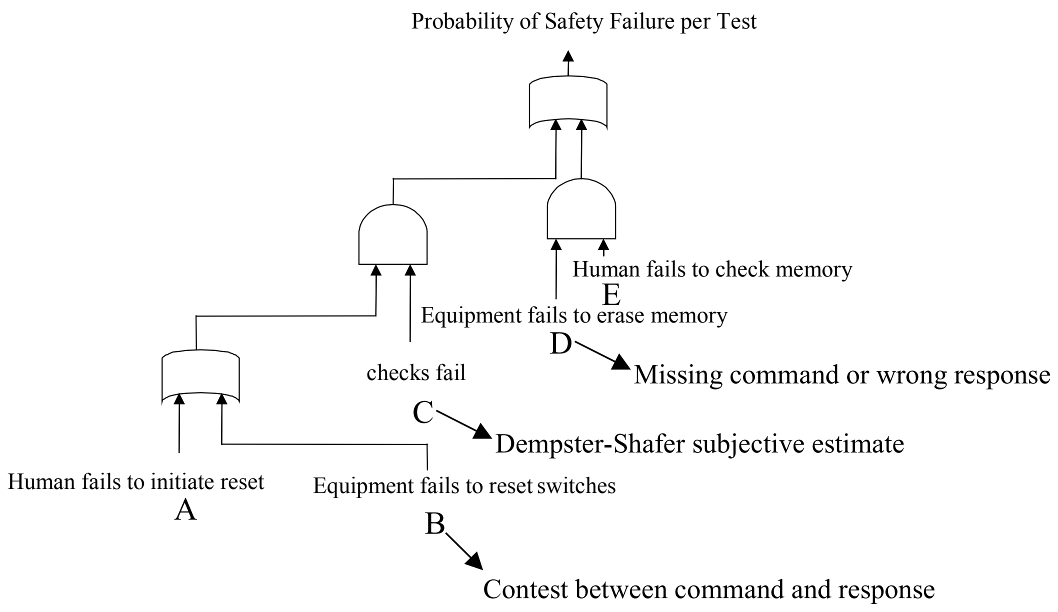
Probability of Safety Failure per Test
Human fails to check memory
E
Equipment fails to erase memory
D
Missing command or wrong response
checks fail
C
Dempster-Shafer subjective estimate
Human fails to initiate reset
A
Equipment fails to reset switches
B
Contest between command and response

Figure 3. Basic Structure of Example Problem

The right-most branch of the fault tree represents communication system storage of information during normal operation, which must be removed following the operation. The environment is assumed to contain relatively high temperature, which is of some concern. Event D represents failure of the equipment to erase the memory. This can be due to human failure to command erase (Event $\overline{N}$) or failure of the equipment to respond properly (Event $\overline{F}$). Since this can be handled by fuzzy logic, the tool used for Event D is Link Minmax, where the equation used is:

$$D = \overline{N} \mid \overline{F} \text{ (Fuzzy “or”)}$$

There must also be a failure of a human to check memory to verify erase (Event E). This is a possibilistic input.

From the fault tree, the overall Top Event equation in COSMET input format is:

T = A&C|B&C|D&E, where & represents “and,” | represents “or,” and & occurs before |.

The inputs used for the example problem are:

A = 10^-4,10^-3,10^-2 (triangular fuzzy number at exponential abscissa positions),

B = Contest:
    Stressor = time

Units = secs.
WL = 1,2,3
SL = 2,3,4

C = Dempster-Shafer:
Expert M: 10^-4,10^-3,10^-2 S = 0.9, w = 0.3
Expert N: 10^-4,10^-3 S= 0.5, w = 0.7
Where S is the subjectivity estimate associated with each expert, and w is the weight assigned to the expert.

D = Link Minmax:
Stressor = temperature
Units = deg. C
Applied stress = 30,50,70
N = 40,50.60
F = 40,50,60
L = N|F

E = 10^-4,10^-3,10^-2

Extreme minmax:
Min = 10^-8,10^-7,10^-6,10^-5
Max = 10^-4,10^-3,10^-2,10^-1

The COSMET execution takes place in two phases. The first phase runs any tool executions necessary prior to the top-event solution. In the example problem, the Contest, Modified Dempster-Shafer, and Link Minmax are run during phase 1. Since these tools have been described elsewhere (see, for example, Appendix A), the phase 1 results are not shown here. The second phase uses the phase 1 results (and any other events not requiring pre-processing) to solve the top-event equation. This example problem has no conventional probabilistic part to it, and it has no Markov scoring for organizational attributes. However, it does demonstrate the hybrid combination of the COSMET tool set.

The COSMET execution outputs begin with an echo of the top-event equation in the form of a "Cutset File."

```
T =
A * C +
B * C +
D * E .
```

The second COSMET output is an echo of the inputs and the results of the first-phase execution (Input File).

```
CUTSETFILE            = REP-CS.DAT    $ Associated Cutset File
DEBUGINFO             = Yes           $ Add Debugging Info. to QA File
ECHOCUTSETFILE        = Yes           $ Turn off Echo if file is large
PRECISERESULT         = Yes           $ Precise Top Event requested
```

```
Fuzzy-Event:   A                     $ A new event
   1.0E-4  1.0E-3  1.0E-2
End-Fuzzy-Event


Fuzzy-Event:   B                     $ Contest B Results from 1st execution
   1.000E-6  3.162E-5  1.000E-3
End-Fuzzy-Event


Fuzzy-Event:   C                     $ D-S Results from 1st execution
   1.000E-4  2.674E-4  2.026E-3  3.700E-3
End-Fuzzy-Event


Fuzzy-Event:   D                     $ Link Min/Max Results from 1st execution
   1.00E-6  1.00E-3  1.00
End-Fuzzy-Event


Fuzzy-Event:   E                     $ A new event
   1.0E-4  1.0E-3  1.0E-2
End-Fuzzy-Event


E-MinMax:      E                     $ Extreme MinMax Definition
   Minimum  =  1.0E-8  1.0E-7  1.0E-6  1.0E-5
   Maximum  =  1.0E-4  1.0E-3  1.0E-2  1.0E-1
End-MinMax
```

A table of Importance and sensitivity calculations for each of the three cutsets is derived:

```
Table No. 1I  (CS Importance Table)
     NCSETS =    3  MXSETS =  100
  ------------------------------
  No.      IXIMPO      RIMPOR
   1.         3      2.767232E-05
   2.         1      2.466276E-06
   3.         2      3.981072E-03
  ------------------------------


    Table No. 3S  (Event Sensitivity Table)
         NTEVTS =    5  MXEVNT =  100
 -------------------------------------------
 No. EVTNAM            IXSENS      SENSIT
  1. A                    4      7.087797E-02
  2. B                    5      2.406685E-03
  3. C                    3      7.341243E-02
  4. D                    1      0.930871
  5. E                    2      0.930871
 -------------------------------------------
```

The next table shows the results for the top event of the tree.

```
              Table No. 8  (Final Result Table)
            TOPEVT = T                TEORDR = 4
            TEAFAC =    0.00000       TEBFAC =    1.00000
-------------------------------------------------------------
  No.    RESULT         No.    RESULT         No.    RESULT
   1.  1.019999E-08      2.  1.275846E-06      3.  3.089996E-06
   4.  1.004026E-02
-------------------------------------------------------------
```

The last table shows the Extreme Min/Max results.

```
                Table No. 46  (E-MINMAX Result Table)
          MMENAM = E                 (No.  1)    NORMME = 4
   --------------------------------------------------------------
     No.    RESMME          No.    RESMME          No.    RESMME
      1.  1.000000E-08       2.  1.000000E-07       3.  1.000000E-02
      4.  0.100000
   --------------------------------------------------------------
```

The composite of the final two tables demonstrates the expected results and the variation possible in credibility variation.

Conclusions

The methodology developed (and the COSMET implementation of that methodology) is an effective supplement to conventional methodologies in that a complex high consequence operation can be sub-divided into parts, and each part can be matched to the most appropriate COSMET tool. Conventional tools are not precluded, but the tool set is much more flexible in matching problem parts to a particular tool. Then all of the results are brought together in a hybrid analysis that provides all of the information to a decision-maker that is warranted by the input information.

References

1. Cooper, J. A., "Markov Modeling with Soft Aggregation for Safety and Decision Analysis," Sandia National Laboratories Report SAND99-2461, September 1999

2. Cooper, J. A. "The Markov Latent Effects Approach to Safety and Decision-Making," Sandia National Laboratories Report SAND2001-2229, September 2001

3. Cooper, J. A., and Kathleen V. Diegert, "Improving analytical Understanding Through the Addition of Information: Bayesian and Hybrid Mathematics Approaches," *Proceedings of the 4th International Conference on Probabilistic Safety Assessment and Management,* Vol. 2, September 1998

4. Cooper, J. A., and Timothy J. Ross, "An Investigation of New Mathematical Structures for Safety Analysis," Sandia National Laboratories Report SAND97-2695, November 1997

5. Cooper, J. A., A. J. Johnson, and P. Werner, "Hybrid Safety Analysis Using Functional and Risk Decompositions," *Proceedings of the 2000 International System Safety Conference,* September 2000

6. Cooper, J. A., S. Ferson, and D. K. Cooper, "Constrained Mathematics for Calculating Logical Safety and Reliability Probabilities with Uncertain Inputs," *Journal of System Safety*: Vol. 36, No. 1, January 2000

7. Cooper, J. A., "Constrained Mathematics Evaluation in Probabilistic Logic Analysis," *Reliability Engineering and System Safety Journal*, Vol. 60, No. 3, June 1998

8. Cooper, J. A., "A Theoretical Description of Methodology in PHASER (Probabilistic Hybrid Analysis System Evaluation Routine), Sandia National Laboratories report SAND96-0022, January 1996

9. Cooper, J. A., "Hybrid Blends of Non-Traditional Safety and Reliability Analysis Tools," *Proceedings of the Tenth European Safety and Reliability Conference,* A. A. Balkema/Rotterdam/Brookfield, September 1999

Glossary

Hybrid Analysis: This term describes combining analyses of various types (e.g., probabilistic calculus and fuzzy mathematics) for one purpose (e.g., determining probability of an occurrence per unit time or per exposure).

Markov Latent Effects Model: A system model that can be used for cause and effect illumination, root cause analysis, and safety analysis. It is based on a concept wherein the causes for inadvertent operational actions are traced back through latent effects to the possible reasons undesirable events may have occurred. The approach is described in detail in Ref. 2. The Markov Latent Effects Model differs substantially from Markov processes, where events do not depend explicitly on past history, and Markov chains of arbitrary order, where dependence on past history is completely probabilistic.

Soft Aggregation: A mathematical accumulation of evidence about an attribute that is nonlinearly accumulated so as to prevent reaching a limiting value.

Surety. Basically encompasses safety, reliability, and security, but also can include other "ilities," such as quality and, certifiability.

System: An interacting collection of entities intended to achieve a common goal.

Uncertainty. Lack of complete knowledge about an outcome, generally associated with subjectivity.

Variability. Outcomes that vary according to prescribed physical laws, such as the outcome of the roll of a fair die, generally associated with objective descriptions of behavior.

Appendix A: COSMET User's Manual

COSMET

(Coordinated Objective/Subjective Mathematically Enhanced Tools)

User's Manual

Version 2.00

Robert J. Roginski August 6, 2001

Part I

General Information

This document defines several features that have been added or revised from Version 1.00 of the COSMET software program. This document supersedes the COSMET Version 1.00 User's Manual dated July 14. 1997. All features are still specified in a user-prepared file known as the Keyword Input File.

Version 2.00 of COSMET can process three different types of data. These are (1) logic (and/or) equations, (2) COSMET possibilistic mathematics-based structures, and (3) Markov modeling structures. However, only one data type can be processed in any given execution. The program performs extensive checking to ensure that the user's Keyword Input File does not contain statements associated with more than one of these three types.

Other documents are referenced in this User's Manual. Where changes have been made, this document will contain the updated information. Information is presented in five parts. Part I describes general program information. Part II contains information that describes the Keyword Input File. Part III describes the logic-oriented statements. Part IV describes the COSMET structures and related keyword statements. Part V describes the Markov Modeling structures and their associated keyword statements. Addendum A lists the keywords that have become obsolete in this version.

The executable program (COSMET.exe) will read the Keyword Input File and generate results in Part 2 of the QA Output File. Part 1 of this file will echo the user's Keyword Input File along with any error diagnostics that may occur. In most cases, the diagnostic will immediately follow the keyword statement found to be in error. The QA Output File will have the same base name as the Keyword Input File, but will have a file extension of .QAF.

To execute COSMET

1. Copy the executable file COSMET.exe to the desired folder (directory).
2. Prepare the Keyword Input File in accordance with the keyword syntax defined in this document. The Keyword Input File should be located in the same folder unless the executable file COSMET.exe has global accessibility via the system PATH setting. Consult your operating system reference manual for details regarding the PATH statement.

3. At the command prompt, execute statement "COSMET *KeywordFile*" where *KeywordFile* is the path/file name of the user's Keyword Input File.

4. If one or more fatal errors exist, the user must edit the Keyword Input File to correct all reported problems and re-execute as previously described.

5. When a successful execution is indicated on the CRT, the results can be found in Part 2 of the QA Output File. Because these results are presented in a self-explanatory fashion, an example of the QA Output File is not presented in this document.

Problem Size

The current limits of COSMET (Version 2.00) are defined in Table 1. The table is shown in three segments, one for each of the three supported data types. A fatal error diagnostic will be written to the QA Output File if any of these limits is exceeded. In this case, execution will be immediately terminated. The limits shown are subject to change with each subsequent release of the program.

Logic Limits

Basic Events	100
Basic Event References (All Cutsets)	800
Basic Event Experts (All Events)	500
Cutsets	100
LHS Sampled Events	100
LHS Observations	5000

LHS File Paths	16

COSMET Structure Limits

Contest Definitions	10
Contestants (Total)	20
Dempster Shafer Modified Models	20
Dempster Shafer Experts (Total)	100
Extreme Min/Max Definitions	10
Link Min/Max Definitions	10
Link Min/Max Responses (Total)	100

Markov Structure Limits

Modules	100
Module Inputs (All Modules)	1000
Module Experts (All Modules)	5000
Dependency Group Definitions	300
Early Alert Equations	15
Early Alert Equation Symbols	750

Table 1. COSMET Version 2.00 Limits.

<u>Machine Compatibility</u>

Version 2.00 of COSMET runs on an IBM or compatible personal computer (PC) executing one of the MS Windows 95, 98, NT, 2000 or ME operating systems. At least 128 MB of installed Random Access Memory (RAM) is recommended. Additional memory may be required if the Markov FILTER feature is used in conjunction with large problems (50-100 modules). A diagnostic message will be issued if sufficient memory is not available for runtime memory allocation. All memory requirements are subject to change with each new release of the program.

Part II

<u>Keyword Input File</u>

COSMET obtains its execution options and instructions from a series of keyword statements in a file whose pathname is specified on the program execution command line. If no pathname is specified, the program will alert the user and immediately terminate. If the pathname is only a filename, the program will expect to find the file in the user's current directory. Some keywords set flags while others indicate that a value (numeric or character) is to be read. In this case, a missing or illegal value will cause the program to abort execution and to write a fatal error message to the QA Output File. Keywords currently recognized by COSMET are described (or referenced) in Parts III, IV and V. Keywords that have become obsolete in this version are listed in Addendum A for user reference.

General Considerations

The Keyword Input File consists of column-independent records (or lines) that can be up to 2000 characters in length. The records consist of tokens that are delimited by at least one space or tab character or any combination of these two characters. A complete keyword statement must begin with one of the recognized keywords and must adhere to the syntax defined for that keyword. Any statement beginning with a token that is not a recognized keyword will result in a fatal diagnostic written to the QA Output File.

Some keyword command tokens may be followed by an optional '=' (equal sign) for improved readability. Use of the '=' is illustrated in many of the examples that follow. Numeric tokens (values) may also use a ',' (comma) as a delimiter.

The keywords currently recognized are described in Parts III, IV and V. The keywords shown there are all in upper case, but the program will accept any combination of upper or lower case, as long as the spelling agrees. This is also true for structure (events, contests, modules, etc.) names, module input names and character string constants (such as Yes and No) that are required to define the various execution options.

Comments

Comments can be placed anywhere in the Keyword Input File using any of three different methods. A trailing comment may be added to a line by appending a '$' at the end of the desired line. In this case, however, the '$' must be preceded by at least one space or tab character so that is not interpreted as part of the previous token. The program will then ignore the '$' and everything that follows it on that line. Whole line comments are allowed if a '$' is the first non-blank, non-tab character on a line. However, the '$' must not start beyond Column 30 for internally technical reasons. The third method of commenting is the use of totally blank lines that may contain only spaces or tab characters.

Continuation Lines

Continuation lines are also possible in the Keyword Input File. Any line can be continued with a continuation line character ('#' or '%') as the last character on the line. The continuation character must be preceded by at least one space or tab character. When the program detects the continuation character as the last token on the line, it will read the next line from the file and continue processing tokens.

Continuations are particularly useful if the user must prepare the input file with a text editor having a line size limit, which is smaller than 2000 characters. Two thousand characters is the maximum line size allowed. Trailing comments are legal on continuation lines as long as there is at least one tab or space character between the continuation character and the '$' that indicates the beginning of the trailing comment.

Numeric Constants

Numeric constants are required input at several places in the Keyword Input File. The required constant may be either an integer or a real number. In places where an integer value is expected, the decimal point, with or without trailing digits, must not be contained in the constant. If a real value is expected, the program will accept the value in any of three formats: fixed point, floating point (E-format), or integer. In this case, the value will be converted to a Fortran double precision real variable regardless of the format used.

Statement Errors

All keyword statements are checked for validity before COSMET begins processing. All parameters following the keyword are also checked for validity. Any parameter that is not recognized or does not fall within the expected range will also generate a fatal error diagnostic. Some errors encountered are not fatal and therefore will not cause execution to terminate, but will generate a warning diagnostic. Both fatal and warning diagnostics will be found in Part 1 (echo of Keyword Input File) of the QA Output File. Some diagnostics will immediately follow the erroneous statement while others will appear at the end of Part 1.

Part III

Logic Related Keyword Statements

The logic related keywords supported by COSMET are the same as those presented in the PHASER User's Manual, Version 2.10, printed September 1996. This is also true of keywords referenced in the PHASER User's Manual Version 2.20 Addendum, dated December 1996.

Several keywords associated with logic processing are supported as described (in detail) in the two documents mentioned above. These supported keywords are listed below for user reference. Many of the logic-oriented keywords have become obsolete in this version of COSMET. These keywords are listed in Addendum A.

CUTSETFILE

DEPENDENCY-GROUP:
ECHOCUTSETFILE
PRECISERESULT

Table 2. Supported Logic Related Keywords in COSMET 2.00.

The LHS keywords described in the PHASER documents are also supported in COSMET Version 2.00. However, LHS itself has been modified so that the limit of 200,000 total values (mentioned in the PHASER 2.20 Addendum) can easily be increased at run time. The latest version of the LHS documentation describes how this is accomplished.

The only other logic related statement that has been revised in COSMET 2.00 is the Fuzzy Event Definition Statement. In addition to the syntax described on Page 27 of the PHASER User's Manual, a new syntax is also available. This syntax allows the user to combine the input values of two or more "experts" for any fuzzy or possibilistic event. This combination of judgments forms the composite input values that will be used in all subsequent fuzzy/possibilistic calculations for a particular event. An example of this new syntax is shown below with line numbers shown for user reference.

```
1)  Fuzzy-Event:   D
2)     1.0E-4   1.0E-2  A=0.3           W=0.3
3)     1.0E-5   1.0E-1  A=0.3           W=0.7
4)  End-Fuzzy-Event
```

Line 1 indicates the beginning of the definition of the Fuzzy Event whose name is D. The event name specified must adhere to the same rules as described in the PHASER User's Manual for the older syntax. The event probabilities and the A-factor are now supplied for each "expert" providing input to this event. In the above example, there are two experts each providing minimum and maximum values (square fuzzy numbers). However, the new syntax also allows optional weight values to appear at the end of these records (lines) for the respective "experts".

The weight of an "expert" will determine how much or little influence the "expert" will have when calculating the event's composite input values. If equal weighting for all "experts" is desired, then all weight factors for that event can be omitted. However, if a single weight factor is specified, then all weight factors for all "experts" in that particular event must be specified and sum to 1.0 within ±1.0E-6. Any number of "experts" can be specified for an event as long as the total "expert" limit for all events is not exceeded. This limit is defined in the first segment of Table 1.

Obviously, Line 4 terminates the definition of the Fuzzy Event D. It should be noted that although the older syntax is still valid, only one "expert" could be specified if this syntax is detected.

Part IV

COSMET Related Keyword Statements

Several of the keywords pertaining to COSMET structures are the same as those presented in the previous version of this document (COSMET User's Manual, Version 1.00, dated July 14, 1997). However, as previously stated, the Version 1.00 manual is superseded by the one you are now reading. Keywords that remain valid are explained here. Keywords no longer supported are listed in Addendum A for user reference.

- -

CONTEST: Specifies the beginning of a multi-line contest definition statement. This allows the user to determine the possibility of a weaklink device surviving a stronglink device. The following file fragment illustrates the use of this feature (line numbers have been shown for user reference). The user should note that lines two through six can be specified in any order as long as they are all found within the contest definition (block).

```
1) CONTEST:   C   WL   SL
2)    STRESSOR   =   Temperature
3)    UNITS      =   "Degrees F"
4)    STRESSORLIMITS    =    -100   1000   100
5)    WL      =   700   800   900
6)    SL      =   800   900  1000
7) END-CONTEST
```

Line one specifies the beginning of a contest designated as "C". The weaklink and stronglink designators are assigned the names "WL" and "SL" respectively. The weaklink designator must be specified first in a contest definition.

Line two specifies the stressor that is applied to both the weaklink and the stronglink. Line three indicates the units to be used for the stressor specified. As shown in Line three, the text associated with the stressor or units should be enclosed in double quotes if it contains one or more spaces or tabs. The case-sensitive text specified in these two lines will be used (by COSMET) to form a suitable X axis label. This can be verified in the resultant plot shown below.

Line four is completely optional; it can be specified by the user to control the plot limit values of the abscissa (X axis). From left to right these values respectively are the minimum, maximum and increment values for the abscissa. The limit values are assumed to be in the units specified by Line three. If "STRESSORLIMITS" is not specified, COSMET will calculate and use limits that are based on the weaklink/stronglink inputs and their calculated difference.

Lines five and six respectively define the weaklink and stronglink failure values. Each line can specify from one to four values in the form of a fuzzy number that should be based on the best expert judgment data available. The first token of each line is a designator that associates the values that follow to either the weaklink or the stronglink. An error will be reported in the user's QA Output file if this token fails to match either the weaklink or the stronglink designator defined in Line one.

The token "END-CONTEST" in Line seven terminates the current contest definition and enables COSMET to accept other statements, including multi-line statements. This line is required by COSMET.

For reasons explained in Part III, contest results cannot be plotted in this version of COSMET. However, the contest results can be found in Table No. 33 of the user's QA Output File. Please note that the generic Keyword Statement "DEBUGINFO = YES" must be present in the Keyword Input File in order to generate this table.

~~~~~~

E-MINMAX: Specifies the beginning of a multi-line extreme Min/Max definition statement. The extreme Min/Max tool allows the user to determine and display the extremes of the top event of a logic tree by considering the "incredible" inputs. The following file fragment illustrates the use of this feature (line numbers have been shown for user reference). The user should note that lines two and three can be specified in either order as long as they are both found within the extreme Min/Max definition (block).

```
1) E-MINMAX:   E
```
~~~~~~

```
2)    MINIMUM =  1.0E-8   1.0E-7   1.0E-6
1.0E-5
3)    MAXIMUM =  1.0E-5   1.0E-4   1.0E-3
1.0E-2
4)  END-MINMAX
```

Line one specifies the beginning of an extreme Min/Max designated as "E".

Lines two and three respectively define the minimum and the maximum credibility inputs. These are displayed along with the top event of the tree. If the analyst has specified reasonably credible inputs, they should bracket (i.e., enclose) the top event as shown in the example plot that follows.

The token "END-MINMAX" in Line four terminates the current Extreme Min/Max definition and enables COSMET to accept other statements, including multi-line statements. This line is required by COSMET.

For reasons explained in Part III, extreme Min/Max results cannot be plotted in this version of COSMET. However, the extreme Min/Max results can be found in Table No. 46 of the user’s QA Output File. Please note that the generic Keyword Statement “DEBUGINFO = YES” must be present in the Keyword Input File in order to generate this table.

~~~~~~

L-MINMAX: Specifies the beginning of a multi-line link Min/Max definition statement. The link Min/Max tool allows the user to determine and display the result of two or more responses that are combined according a user-defined equation. The following file fragment illustrates the use of this feature (line numbers have been shown for user reference). The user should note that lines two through nine can be specified in any order as long as they are all found within the link Min/Max definition (block).

```
1)  L-MINMAX:   L
2)    STRESSOR    =   Temperature
3)    UNITS       =   "Degrees C"
4)  $$$$$$   STRESSORLIMITS     =  -200   1000
200
5)    APPLIED-STRESS     =     8      9    10
6)    RESPONSE       A   =    10     12     20
7)    RESPONSE       B   =     5     15
8)    RESPONSE       C   =     5     10     12
20
9)    L                  =    A   &   B   |   C
```
~~~~~~

```
10) END-MINMAX
```

Line one specifies the beginning of a link Min/Max designated as "L".

Line two specifies the stressor that is applied to the resultant response (the combination of user-supplied responses). Line three indicates the units to be used for the stressor specified. As shown in Line three, the text associated with the stressor or units should be enclosed in double quotes if it contains one or more spaces or tabs. The case-sensitive text specified in these two lines will be used (by a future version of COSMET) to form a suitable X axis label for plotting purposes.

Line four is completely optional. If specified by the user, it will control the plot limit values of the abscissa (X axis). From left to right these values respectively are the minimum, maximum and increment values for the abscissa. The limit values are assumed to be in the units specified by Line three. If "STRESSORLIMITS" is not specified, COSMET will calculate and use limits that are based on the applied stress input and the calculated resultant response. In the example shown above, Line four has no effect on the plot because it is commented out.

Lines six through eight specify the user-supplied response definitions that are to be combined to form the resultant response. The user can specify as many response definitions as necessary as long as the total response limit (for all link Min/Max definitions) is not exceeded (refer to the second segment of Table 1 for this value). As shown, the name of the response follows keyword "RESPONSE" in these definitions. Each response name must be unique within the link Min/Max definition. However, the same response name may also appear in <u>other</u> link Min/Max definitions.

The response values follow the optional '=' (equal sign) in the response definition statement. Each statement can specify from one to four values (in units of "stressor") for which COSMET will assume uniformly distributed values of membership. For example, specifying three stressor values implies corresponding membership values of 0.0, 0.5 and 1.0 respectively. These values must be specified in increasing order or a fatal error diagnostic will be written to the user's QA Output File.

Line nine specifies the equation that describes how the user-supplied responses are to be combined to form the resultant response. The first token in the line is the name of the link Min/Max definition. Specifying this name is optional, but is permitted for the sake of clarity and completeness of the equation. The '=' (equal sign) that follows must not be specified if the link Min/Max name is omitted. The operators '&' and '|' in the equation respectively indicate the minimum and maximum operations. These operations are performed on the user-specified operands (which must be valid response names) to determine the resultant response curve.

The applied stress and the resultant response are then combined (using methodologies described in Ref. 1) to determine the possibility of failure.

For reasons explained in Part III, link Min/Max results cannot be plotted in this version of COSMET. However, the link Min/Max results can be found in Table Nos. 56 and 57 of the user's QA Output File. Please note that the generic Keyword Statement "DEBUGINFO = YES" must be present in the Keyword Input File in order to generate these tables.

DEMPSTER-SHAFER-MOD: Specifies the beginning of a multi-line Dempster Shafer modified model definition statement. This feature allows the user to determine an adjusted (wider) range of uncertainty at the highest level of presumption (1.0) by varying the amount of subjectivity specified for each "expert" contributing to the input. The following file fragment illustrates the use of this feature (line numbers have been shown for user reference).

```
1)  DEMPSTER-SHAFER-MOD:    DS-Test-No-1
2)     0.1   0.2   0.3            S=0.9   W=0.2
3)     0.3   0.4   0.5   0.6   S=0.5   W=0.3
4)     0.4   0.5   0.6   0.7   S=0.1   W=0.2
5)     0.4   0.8                  S=0.5   W=0.2
6)     0.6                        S=0.3   W=0.1
7)  END-DEMPSTER-SHAFER-MOD
```

Line one specifies the beginning of a Dempster Shafer modified model designated as "DS-TEST-NO-1".

Lines two through six are the records associated with the five "experts" providing input to this model. As indicated, each record begins with 1-4 values that define a fuzzy shape for each "expert". The subjectivity value for each "expert" immediately follows the "S=" in each respective record. This subjectivity value must be in the range of 0.0 to 1.0. The weight factor for each "expert" will immediately follow the "W=" in each respective record.

The weight of an "expert" will determine how much or little influence the "expert" will have when calculating a composite input for the model. If equal weighting for all "experts" is desired, then all weight factors for that model can be omitted. However, if a single weight factor is specified, then all weight factors for all "experts" in that particular model must be specified and sum to 1.0 within ±1.0E-6. Any number of "experts" can be specified for a model as long as the total limit for all models is not exceeded. This limit is defined in the second segment of Table 1.

The token "END-DEMPSTER-SHAFER-MOD" in Line seven terminates the current definition and enables COSMET to accept other statements, including multi-line statements. This line is required by COSMET.

For reasons explained in Part III, the results of Dempster Shafer modified models cannot be plotted in this version of COSMET. However, these results can be found in Table Nos. 75 and 76 of the user's QA Output File. Please note that the generic Keyword Statement "DEBUGINFO = YES" must be present in the Keyword Input File in order to generate these tables.

~~~~~~

Part V

Markov Related Keyword Statements
~~~~~~

The Markov related keywords supported by COSMET are identical to those presented in the Markov User's Manual, Version 2.00, and dated May 15, 2001. For this reason the Markov related keywords will not be described in this document. However, a few differences exist that need to be explained here.

1. When using the COSMET code to perform a Markov analysis, the user should execute the statement "COSMET *KeywordFile*" as described on Page 24 of this document. The Markov User's Manual instructs the user to execute the statement "MARKOV *KeywordFile*".

2. The Markov executable program will always normalize the Importance and Sensitivity results, both the primary and the secondary. The normalization techniques used are described in the Markov User's Manual. On the other hand, the COSMET executable program does <u>not</u> normalize these results. They appear as calculated in Table Nos. 67 and 67A of the user's QA Output File. Once again, please note that the generic Keyword Statement "DEBUGINFO = YES" must be present in the Keyword Input File in order to generate these tables.

Reference

1. Cooper, J. Arlin, "Hybrid Processing of Measurable and Subjective Data," Sandia National Laboratories Report SAND01-3317, October 2001.

Addendum A

Obsolete Keywords

A number of the keywords described in previously mentioned documents are no longer supported in COSMET 2.00. For the most part, these are plot-oriented keywords, that have been disabled in anticipation of a future graphical Windows Interface[5]. They are listed below for user reference. If any of these keywords are encountered in the user's Keyword Input File, a warning diagnostic will be written to the QA Output File and the entire keyword statement will be ignored.

CUTSETLIMIT	
DELAY	
MAXDECIMALS	
MAXPROBABILITY	(Obsolete in PHASER 2.20)
MINPROBABILITY	(Obsolete in PHASER 2.20)
OMITTOPEVENT	
NLINES	(No Longer Required)
OUTPUTDEVICE	
PLOTCOMPOSITE	
PLOTCUTSETS	
PLOTEVENTS	
PLOTIMPORTANCE	
PLOTSENSITIVITY	
PLOTTOPEVENT	
SOLVECUTSETEQUATION	(No Longer Required)

Table 3. Obsolete Logic Related Keywords in COSMET 2.00.

For the same reason, a number of the COSMET related keywords previously supported are also obsolete. However, modifications of other COSMET features have eliminated the need for the WEIGHTED-SUM: and the SYSTEMRESULT Keyword Statements.

PLOTCONTESTS	
PLOT-E-MINMAX	
PLOT-L-MINMAX	
PLOTWEIGHTEDSUMS	
SYSTEMRESULT	(No Longer Required)
WEIGHTED-SUM:	(No Longer Required)

[5] This future Windows Graphical User Interface (GUI) will generate user-specified plots and will store all plotting parameters in a separate file. The user will select plotting options using the standard Windows check boxes and radio buttons.

Table 4. Obsolete COSMET Related Keywords in COSMET 2.00.

Appendix B. Markov User's Manual

MARKOV

Modeling Tool

User's Manual

Version 2.00

Robert J. Roginski May 15, 2001

Part I

General Information

This document defines several features that have been incorporated into a Markov modeling software program. These features are specified in a user-prepared file known as the Keyword Input File.

The executable program (Markov.exe) will read the Keyword Input File and generate Markov results in Part 2 of the QA Output File. Part 1 of this file will echo the user's Keyword Input File along with any error diagnostics that may occur. In most cases, the diagnostic will immediately follow the keyword statement found to be in error. The QA Output File will have the same base name as the Keyword Input File, but will have a file extension of .QAF.

To execute the Markov program

1. Copy the executable file Markov.exe to the desired folder (directory).

2. Prepare the Keyword Input File in accordance with the keyword syntax defined in this document. The Keyword Input File should be located in the same folder unless the executable file Markov.exe has global accessibility via the system PATH setting. Consult your operating system reference manual for details regarding the PATH statement.

3. At the command prompt, execute statement "MARKOV *KeywordFile*" where *KeywordFile* is the path/file name of the user's Keyword Input File.

4. If one or more fatal errors exist, the user must edit the Keyword Input File to correct all reported problems and re-execute as previously described.

5. When a successful execution is indicated on the CRT, the Markov results can be found in Part 2 of the QA Output File. Because these results are presented in a self-explanatory fashion, an example of the QA Output File is not presented in this document.

6. If post-processing and/or plotting of Markov results is desired by the user, the Capture File will contain the same results in a format better suited as input to a user-written program. The CAPTURE keyword statement (described in Part III) must be specified in the user's Keyword Input File in order to generate the Capture File.

Problem Size

The current limits of the Markov Modeling Tool (Version 2.00) are defined in Table 1. A fatal error diagnostic will be written to the QA Output File if any of these limits is exceeded. In this case, execution will be immediately terminated. The limits shown are subject to change with each subsequent release of the program.

Modules	100
Module Inputs (All Modules)	1000
Module Experts (All Modules)	5000
Dependency Group Definitions	300
Early Alert Equations	15
Early Alert Equation Symbols	750

Table 2. Markov Tool Version 2.00 Limits.

Machine Compatibility

Version 2.00 of the Markov tool runs on an IBM or compatible personal computer (PC) executing one of the MS Windows 95, 98, NT, 2000 or ME operating systems. At least 128 MB of installed Random Access Memory (RAM) is recommended. Additional memory may be required if the FILTER feature is used in conjunction with large problems (50-100 modules). A diagnostic message will be issued if sufficient memory is not available for runtime memory allocation. All memory requirements are subject to change with each new release of the program.

Part II
Keyword Input File

The Markov tool obtains its execution options and instructions from a series of keyword statements in a file whose pathname is specified on the program execution command line. If no pathname is specified, the program will alert the user and immediately terminate. If the pathname is only a filename, the program will expect to find the file in the user's current directory. Some keywords set flags while others indicate that a value (numeric or character) is to be read. In this case, a missing or illegal value will cause the program to abort execution and to write a fatal error message to the QA Output File. Keywords currently recognized by the Markov tool are described in Part III.

General Considerations

The Keyword Input File consists of column-independent records (or lines) that can be up to 2000 characters in length. The records consist of tokens that are delimited by at least one space or tab character or any combination of these two characters. A complete keyword statement must begin with one of the recognized keywords and must adhere to the syntax defined for that keyword. Any statement beginning with a token that is not a recognized keyword will result in a fatal diagnostic written to the QA Output File.

Some keyword command tokens may be followed by an optional '=' (equal sign) for improved readability. Use of the '=' is illustrated in many of the examples that follow. Numeric tokens (values) may also use a ',' (comma) as a delimiter.

The keywords currently recognized are described in Part III. The keywords shown there are all in upper case, but the program will accept any combination of upper or lower case, as long as the spelling agrees. This is also true for module names, input names and character string constants (such as Yes and No) that are required to define the various execution options.

Comments

Comments can be placed anywhere in the Keyword Input File using any of three different methods. A trailing comment may be added to a line by appending a '$' at the end of the desired line. In this case, however, the '$' must be preceded by at least one space or tab character so that is not interpreted as part of the previous token. The program will then ignore the '$' and everything that follows it on that line. Whole line comments are allowed if a '$' is the first non-blank, non-tab character on a line. However, the '$' must not start beyond Column 30 for internally technical reasons. The third method of commenting is the use of totally blank lines that may contain only spaces or tab characters.

Continuation Lines

Continuation lines are also possible in the Keyword Input File. Any line can be continued with a continuation line character ('#' or '%') as the last character on the line. The continuation character must be preceded by at least one space or tab character. When the program detects the continuation character as the last token on the line, it will read the next line from the file and continue processing tokens. Continuations are particularly useful if the user must prepare the input file with a text editor having a line size limit, which is smaller than 2000 characters. Two thousand characters is the maximum line size allowed. Trailing comments are legal on continuation lines as long as there is at least one tab or space character between the continuation character and the '$' that indicates the beginning of the trailing comment.

Numeric Constants

Numeric constants are required input at several places in the Keyword Input File. The required constant may be either an integer or a real number. In places where an integer value is expected, the decimal point, with or without trailing digits, must not be contained in the constant. If a real value is expected, the program will accept the value in any of three formats: fixed point, floating point (E-format), or integer. In this case, the value will be converted to a Fortran double precision real variable regardless of the format used.

Statement Errors

All keyword statements are checked for validity before the Markov tool begins processing. All parameters following the keyword are also checked for validity. Any parameter that is not recognized or does not fall within the expected range will also generate a fatal error diagnostic. Some errors encountered are not fatal and therefore will not cause execution to terminate, but will generate a warning diagnostic. Both fatal and warning diagnostics will be found in Part 1 (echo of Keyword Input File) of the QA Output File. Some diagnostics will immediately follow the erroneous statement while others will appear at the end of Part 1.

Part III

Keyword Statements

The keyword statements described in Part III allow the user to define Markov modules and set program execution parameters. Unless indicated otherwise, a default value will be assumed for any missing keyword statement. These default values are also defined in this section.

CAPTURE [=] *CaptureFile*

Specifies *CaptureFile* as the name of the file that will contain the results of each execution of the Markov tool. The Capture File will be created if it does not currently exist. If it does exist, a block of information containing the results of the current execution is appended to this file.

Example: `CAPTURE = D:\Test1\Capture-File.dat`

The parameter *CaptureFile* may specify a full path of 50 characters in length. If the user specifies only a filename, the file must be found in the current directory.

The CAPTURE keyword statement is mandatory if the user also specifies the FILTER keyword statement.

The order and format of results written to the Capture File are described in Addendum A.

- -

EARLY-ALERT:

Specifies the beginning of a multi-line early alert equation definition. The early alert feature allows the user to determine threshold ranges by the user-specified combination of minimums and/or maximums of two or more module inputs and/or outputs. The following file fragment illustrates the use of this feature (line numbers have been shown for user reference).

```
1)    EARLY-ALERT:
2)       E3  =  X9  &  (  X76  |  X78  |  X86  )
3)    END-ALERT
```

Line one specifies the beginning of the early alert definition. Line two specifies the equation for the early alert whose name is E3. In the above example, the maximum corresponding values of X76, X78 and X86 are first determined by specifying the '|' operator. The result of E3 is then determined by taking the minimum values of X9 and the corresponding values of the previous operation. Line three specifies the end of the early alert definition.

Each operand (input or output name) and each operator ('=', '&', '|', '(' and ')') in the equation must be delimited by at least one space or tab character. Although not shown in the above example, parentheses may be nested to explicitly define any desired order of evaluation.

Although the '=' (equal sign) is optional in the early alert equation, it is strongly recommended for clarity.

The above example shows a single line equation, but multiple lines are permitted in the early alert equation. If more than one line is necessary, the continuation line characters '#' and '%' described in Part II can be used. However, they are not actually required here, because the mandatory END-ALERT statement will signify the end of the equation.

Any module that defines an operand (input or output name) referenced in the early alert equation must precede the early alert definition in the user's Keyword Input File.

~~~~~~

FILTER [=] Yes/No

Instructs the program to request or suppress filtering of all previously captured Markov results. Filtered results include: (1) Primary Inputs, (2) Module Outputs, (3) Primary Importance Values, (4) Primary Sensitivity Values, (5) Secondary Importance Values, (6) Secondary Sensitivity Values, and (7) Early Alerts.

These filtered results will be written to a file that is always named FILTERED-MARKOV-DATA.DAT. The user should be cautioned that any existing file bearing this name would be overwritten with each execution. The format of this file is described in Addendum B.

Example:	`FILTER = Yes`
Default:	`FILTER = No`

The filter feature reads the Capture File as input. For this reason, the CAPTURE keyword statement is mandatory if the FILTER keyword statement is specified. Also, the Capture File must contain at least two data segments (i.e., results of two Markov executions) with unique dates.

The filter feature requires all execution segments of the Capture File to be identical in structure. This means that they should all have the same number of defined modules,
~~~~~~

inputs, inputs per module and early alert definitions. The filtering process will be aborted and a diagnostic will be issued to the QA Output File if a change in structure is detected between executions. To avoid this problem, the user should not specify either the CAPTURE or FILTER keyword statements until the Markov structure is well established. The weights and the Markov structure should not change between executions if trend plots and/or filtering are desired. The resulting different outputs written to the Capture File will become a source of input to the filtering process.

- -

MODULE: Specifies the beginning of a multi-line Markov module definition. The following file fragment defines two modules that illustrate the various components of a module (line numbers are shown for user reference). Explanations of these components will follow.

```
 1)    MODULE:   Management
 2)       INPUT:   Policy           0.1
 3)                0.8    0.9       W=1.0
 4)       END-INPUT
 5)       INPUT:   Compliance       0.1
 6)                   0.5     0.6     0.7     0.8
W=0.8
 7)                        0.5       0.7       0.9
W=0.2
 8)       END-INPUT
 9)       INPUT:   Turnover         0.05
10)                0.3    0.4
11)                0.3
12)       END-INPUT
13)       DEPENDENCY-GROUP:         0.5
14)          INPUT:   Culture          0.35
15)                   0.7    0.8
16)          END-INPUT
17)          INPUT:   Communication     0.2
18)                   0.3    0.4
19)          END-INPUT
20)          INPUT:   Training         0.2
21)                   0.4    0.6
22)          END-INPUT
23)       END-GROUP
24)    END-MODULE
25)
26)
27)    MODULE:   Processes
28)       INPUT:   Guidance         0.2
29)                0.9
30)       END-INPUT
31)       INPUT:   Config-Control   0.2
32)                0.9
33)       END-INPUT
34)       INPUT:   Internal-Review  0.2
35)                0.6    0.8
36)       END-INPUT
37)       INPUT:   External-Review  0.2
38)                0.7    0.8
39)       END-INPUT
```

```
40)       INPUT:   Management       0.2
41)    END-MODULE
```

Line 1 indicates the beginning of the definition of the module to be known as Management. The module name specified must consist of 1 to 16 case-insensitive alphanumeric characters beginning with an alpha (a-z) character. The module name must also be unique (i.e., not previously defined as a module or an input).

Lines 2, 5 and 9 respectively indicate the beginnings of the independent primary inputs Policy, Compliance, and Turnover. The input names specified must adhere to the same rules as module names. The last items found on these lines are the required weight values for the respective inputs. The weight of an input will determine how much or little influence the input will have when calculating the module's output. Within any given module, the weights of all defined inputs must sum to 1.0. This includes both primary and secondary inputs, both independent and dependent (those defined within a "DEPENDENCY-GROUP:" block described later).

As shown, these three inputs contain one or more records (lines) corresponding to the number of "experts" contributing to the respective input[6]. Each expert's input record consists of 1 to 4 values defining a possibilistic number. This possibilistic number will be singular, square, triangular or trapezoidal based on the number of values specified. An optional weight factor (following "W=") is specified for expert records of inputs Policy and Compliance. This was not necessary in the case of input Policy because only a single "expert" was indicated. On the other hand, the definition of input Compliance indicates that the judgment of Expert No. 1 should be weighted higher than that of Expert No. 2. If different weights are to be used, then a weight for each expert contributing to the input must be specified. If individual weights are specified, they must also sum to 1.0 for that input. If equal weighting of all experts is desired, then all expert weight values can be omitted for the input. If more than one expert is specified, they are combined to form a

[6] A module definition may specify any number of inputs as long as the total number of inputs in all defined modules does not exceed the limit defined in Table 1. In like fashion, a primary input definition may specify any number of "experts" as long as the limit defined in Table 1 is not exceeded.

composite possibilistic value that is used in all subsequent calculations.

The token "End-Input" found at Lines 4, 8 and 12 respectively terminates the definitions of primary inputs Policy, Compliance, and Turnover.

Line 13 specifies the beginning of a dependency group definition[7]. This group contains three members that are dependent on each other. The amount of dependency is specified at the end of Line 13. This dependency value must be in the range of 0.0 to 1.0. The token "End-Group" found at Line 23 terminates the dependency group definition. The user should note that the syntax of inputs defined within a dependency group is exactly the same as that of their independent counterparts, both primary and secondary. The only difference is that indentation is used to illustrate their hierarchy within the module structure. However, this indentation is not actually required by the program.

The token "End-Module" found at line 24 terminates the definition of the module known as Management.

The definition of the module known as Processes begins with Line 27 and ends with Line 41. This module has five inputs, the last of which is a secondary input. At Line 40, notice that the name specified as the secondary input is the name of the previously defined module Management. This simply means that the output of Module Management is used as an input to Module Processes. This linking of names is used to create the latent effects structure inherent in Markov modeling. If the case-insensitive name specified is not the name of a defined module, a fatal diagnostic will be issued to the QA Output File.

Obviously, a secondary input cannot be used in the calculation process unless the module it references is evaluated first. For this reason, the ordering of module definitions within the Keyword Input File is significant. Simply stated, a module cannot be referenced as a secondary input unless its definition precedes the definition of the module attempting to reference it. The program strictly enforces this rule to preclude the possibility of

[7] A module definition may specify any number of dependency groups as long as the total number of dependency groups in all defined modules does not exceed the limit defined in Table 1.

feedback[8]. Because modules are evaluated in the order of their definition, this rule also ensures that the output of each module is available before it is required in any subsequent evaluation. The preceding example defines the two modules in the required order.

In the example, the "End-Input" terminator does not follow the secondary input definition at Line 40. This terminator is optional for secondary inputs. The only other required parameter for a secondary input is its weight factor, which immediately follows the referenced module name on the same line.

Although not shown in the preceding example, secondary inputs are also allowed within a dependency group definition. Once again, the syntax is exactly the same, but indentation (as shown) is highly recommended to illustrate hierarchy within the module.

[8] Feedback exists whenever a module uses its own output as an input, either directly or indirectly. This condition is not allowed or supported by the Markov tool.

Addendum A

<u>Capture File Format</u>

As previously stated, the Capture File will collect the Markov results for each execution if the keyword statement "CAPTURE = *CaptureFilename"* is specified in the Keyword Input File. An example of a single execution will be shown to describe the format of each block of information written to the Capture File. However, keep in mind that a block of results is appended to the Capture File each time the program is executed. This can result in a very large Capture File, especially if many modules and inputs are defined.

Although shown separately in this Addendum for ease of reference, the date header and all nine information blocks are written to the Capture File as a contiguous stream of text without intervening blank lines or comments. Also, if more than a single execution has been written to the Capture File, the date/time stamp for subsequent executions will <u>immediately</u> follow the early alert information block of the preceding execution. These items and all other information blocks will be described in their order of appearance in the Capture File.

If required, the user can write a program to post-process or plot any or all data contained in the Capture File. Any information block that is not of interest can easily be ignored when the file is read. This is possible because the size of each block is either specified in a line preceding the block, or the block itself is delimited by known tokens (described later in this Addendum) that the program can use as keywords.

Each execution block written to the Capture begins with a one-line date/time stamp header that identifies the time of execution. The following is an example.

```
05/02/2001 08:56:15
```

The first block of information contains primary input information. The first line in the block indicates the number of primary input records that are to follow.

```
   17   Primary Inputs
POLICY              0  0.100 0.800 0.800 0.900 0.900
COMPLIANCE          0  0.100 0.500 0.620 0.700 0.900
TURNOVER            0  0.050 0.300 0.300 0.350 0.400
CULTURE             1  0.350 0.700 0.700 0.800 0.800
COMUNICATION        1  0.200 0.300 0.300 0.400 0.400
TRAINING            1  0.200 0.400 0.400 0.600 0.600
GUIDANCE            0  0.200 0.900 0.900 0.900 0.900
CONFIG-CONTROL      0  0.200 0.900 0.900 0.900 0.900
INTERNAL-REVIEW     0  0.200 0.600 0.600 0.800 0.800
EXTERNAL-REVIEW     0  0.200 0.700 0.700 0.800 0.800
EXPERIENCE          0  0.300 0.700 0.700 0.800 0.800
SKILL               0  0.300 0.800 0.800 0.900 0.900
PROCESS-COMPLI.     0  0.200 0.300 0.300 0.500 0.500
DOD-CONTROLS        0  0.300 0.400 0.400 0.600 0.600
HUMAN-RISK          0  0.100 0.300 0.300 0.700 0.700
NORMAL-EN-RISK      2  0.100 0.100 0.100 0.200 0.200
ABNORMAL-EN-RISK    2  0.200 0.400 0.400 0.600 0.600
```

Each primary input record has seven fields of information. The first field is the name of the primary input converted to upper case. The second field is the input's group membership identifier. A value of 0 (zero) in this field indicates no group affiliation (an independent input). The third field is the weight of the input as specified by the user in the Keyword Input File. The last four fields represent the possibilistic number assigned to the input by the user, except that they are expanded to four values. The inputs in this block are listed in the order of their appearance in the user's Keyword Input File.

The second block of information contains secondary input information. The first line in the block indicates the number of secondary input records that are to follow. The first of three fields is the name of the input converted to upper case. Note that duplicate names are possible in this field because a module can be a secondary input to one or more <u>subsequently defined</u> modules. The inputs in this block are listed in the order of their appearance in the user's Keyword Input File.

```
    5   Secondary Inputs
MANAGEMENT          0  0.200
MANAGEMENT          0  0.100
PROCESSES           0  0.100
PROCESSES           0  0.100
IMPLEMENTATION      0  0.200
```

The second field is the input's group membership identifier. A value of 0 (zero) in this field indicates no group affiliation (an independent input). The third field is the weight of the input as specified by the user in the Keyword Input File.

The third block of information contains module output information. The first line in the block indicates the number of module output records that are to follow. The first of eight fields is the name of the output converted to upper case. The outputs in this block are listed in the order in which they are defined in the user's Keyword Input File.

```
    4   Outputs of Modules
MANAGEMENT         6   0   1  0.4317 0.4479 0.5918 0.6213
PROCESSES          4   1   0  0.7567 0.7600 0.8377 0.8420
IMPLEMENTATION     3   2   0  0.6701 0.6725 0.8008 0.8037
DOD-USE            4   2   1  0.4072 0.4082 0.6467 0.6480
```

The second field is the number of primary inputs to this module. The third field is the number of secondary inputs to this module. The fourth field is the number of dependency groups defined in this module. The last four fields represent the possibilistic number calculated for the output of this module.

The fourth block of information contains dependency group information. The first line in the block indicates the number of dependency group records that are to follow. The dependency groups in this block are listed in the order in which they are defined in the user's Keyword Input File.

```
    2   Dependency Groups
0.5000 0.3050 0.3050 0.3900 0.3900
0.3000 0.0720 0.0720 0.1160 0.1160
```

The first field is the amount of dependence for the group as specified by the user in the Keyword Input File. The last four fields represent the possibilistic number calculated for the output of this dependency group. These values are subsequently used in the calculation of their corresponding module's output.

The fifth block of information contains primary importance information. The first line in the block indicates the number of primary input records that are to follow. The first of three fields is the name of the primary input converted to upper case. The primary inputs in this block are listed in the order in which they are defined in the user's Keyword Input File.

```
   17   Primary (Overall) Normalized Importance Values
POLICY             3.455613E-02   14
COMPLIANCE         2.760202E-02   17
TURNOVER           6.503563E-03   12
CULTURE            9.965209E-02   11
COMUNICATION       6.669787E-02   15
TRAINING           7.228051E-02    7
GUIDANCE           0.168386        8
CONFIG-CONTROL     0.168386       16
INTERNAL-REVIEW    0.125147       10
EXTERNAL-REVIEW    0.135714       13
EXPERIENCE         0.394912        9
SKILL              0.450182        4
PROCESS-COMPLI.    0.127388        6
DOD-CONTROLS        1.00000        5
HUMAN-RISK         0.344236        1
NORMAL-EN-RISK     0.165296        2
ABNORMAL-EN-RISK   0.571738        3
```

The second field is the importance relative to the final output (i.e., the output of the last module defined) calculated by the program for this input[9]. The third field is an index that provides sort information in descending order of importance. For example, in the preceding list, the <u>most</u> important input is No. 14, DOD-CONTROLS. The <u>least</u> important input is No. 3, TURNOVER. Note that these importance values are normalized by dividing each one by the value calculated for the most important input.

The sixth block of information contains primary sensitivity information. The first line in the block indicates the number of primary input records that are to follow. The first of three fields is the name of the primary input converted to upper case. The primary inputs in this block are listed in the order in which they are defined in the user's Keyword Input File.

```
   17   Primary (Overall) Normalized Sensitivity Values
POLICY             6.207908E-03   14
COMPLIANCE         1.310759E-02   16
TURNOVER           1.380865E-02   17
CULTURE            1.777889E-02   15
COMUNICATION       4.606734E-02   13
```

[9] A zero primary importance value indicates a primary input that does not output to the last defined module, either directly or indirectly. This will occur if the user fails to "connect" it to the last module as either a direct or an indirect input. A direct input would be a primary input to that module. An indirect input would be any primary input that begins a "chain" of modules, the last of which is defined as a secondary input to that module.

```
TRAINING           2.022789E-02   11
GUIDANCE           1.471833E-02   12
CONFIG-CONTROL     1.471833E-02    5
INTERNAL-REVIEW    4.112867E-02    9
EXTERNAL-REVIEW    3.489564E-02   10
EXPERIENCE         0.101811        6
SKILL              6.397718E-02    4
PROCESS-COMPLI.    0.151469        7
DOD-CONTROLS        1.00000        8
HUMAN-RISK         0.355323        3
NORMAL-EN-RISK     0.633254        2
ABNORMAL-EN-RISK   0.492858        1
```

The second field is the sensitivity relative to the final output (i.e., the output of the last module defined) calculated by the program for this input[10]. The third field is an index that provides sort information in descending order of sensitivity. For example, in the preceding list, the <u>most</u> sensitive input is No. 14, DOD-CONTROLS. The <u>least</u> sensitive input is No. 1, POLICY. Note that these sensitivity values are normalized by dividing each one by the value calculated for the most sensitive input.

The seventh block of information contains secondary importance information. The first line in the block is a header that indicates the number of sub-blocks (modules) that follow. The first line in a sub-block indicates the number of secondary input records that are to follow for that particular sub-block. Within a sub-block, the first of three fields is the name of the secondary input converted to upper case. The modules in this block are listed in the order in which they are defined in the user's Keyword Input File. The inputs are listed in the order in which they are defined within their corresponding modules.

```
    4   Secondary (Module) Normalized Importance Values
    6   Inputs for Module No.   1
POLICY             0.115633       4
COMPLIANCE         9.290943E-02   6
TURNOVER           2.230074E-02   5
CULTURE            0.317030       1
COMUNICATION       0.217468       2
TRAINING           0.234658       3
    5   Inputs for Module No.   2
GUIDANCE           0.244910       1
CONFIG-CONTROL     0.244910       2
INTERNAL-REVIEW    0.182306       4
EXTERNAL-REVIEW    0.197620       3
MANAGEMENT         0.130253       5
    5   Inputs for Module No.   3
EXPERIENCE         0.334298       2
SKILL              0.381509       1
PROCESS-COMPLI.    0.107754       5
MANAGEMENT         6.836047E-02   3
PROCESSES          0.108079       4
    6   Inputs for Module No.   4
DOD-CONTROLS       0.276107       1
HUMAN-RISK         9.504582E-02   4
PROCESSES          0.151784       6
IMPLEMENTATION     0.273563       3
NORMAL-EN-RISK     4.563943E-02   2
ABNORMAL-EN-RISK   0.157861       5
```

[10] A zero primary sensitivity value indicates a primary input that does not output to the last defined module, either directly or indirectly. This will occur if the user fails to "connect" it to the last module as either a direct or an indirect input. A direct input would be a primary input to that module. An indirect input would be any primary input that begins a "chain" of modules, the last of which is defined as a secondary input to that module.

The second field is the importance relative to the module's output (i.e., the output of the module of which it is an input) calculated by the program for this input. The third field is an index that provides sort information in descending order of importance. For example, in Module No. 4 of the preceding list, the <u>most</u> important input is No. 1, DOD-CONTROLS. The <u>least</u> important input is No. 5, NORMAL-EN-RISK. Note that these importance values are normalized by dividing each one by the sum of all calculated importance values.

The eighth block of information contains secondary sensitivity information. The first line in the block is a header that indicates the number of sub-blocks (modules) that follow. The first line in a sub-block indicates the number of secondary input records that are to follow for that particular sub-block. Within a sub-block, the first of three fields is the name of the secondary input converted to upper case. The modules in this block are listed in the order in which they are defined in the user's Keyword Input File. The inputs are listed in the order in which they are defined within their corresponding modules.

```
    4   Secondary (Module) Normalized Sensitivity Values
    6   Inputs for Module No.   1
POLICY              5.192225E-02    5
COMPLIANCE          0.110309        6
TURNOVER            0.116283        4
CULTURE             0.150257        3
COMUNICATION        0.399891        2
TRAINING            0.171338        1
    5   Inputs for Module No.   2
GUIDANCE            8.812551E-02    5
CONFIG-CONTROL      8.812551E-02    3
INTERNAL-REVIEW     0.246603        4
EXTERNAL-REVIEW     0.209160        1
MANAGEMENT          0.367986        2
    5   Inputs for Module No.   3
EXPERIENCE          0.245902        3
SKILL               0.154336        1
PROCESS-COMPLI.     0.366499        4
MANAGEMENT          0.162944        2
PROCESSES           7.031984E-02    5
    6   Inputs for Module No.   4
DOD-CONTROLS        0.334534        1
HUMAN-RISK          0.118868        5
PROCESSES           4.750725E-02    6
IMPLEMENTATION      0.122367        4
NORMAL-EN-RISK      0.211845        2
ABNORMAL-EN-RISK    0.164878        3
```

The second field is the sensitivity relative to the module's output (i.e., the output of the module of which it is an input) calculated by the program for this input. The third field is an index that provides sort information in descending order of sensitivity. For example, in Module No. 4 of the preceding list, the <u>most</u> sensitive input is No. 1, DOD-CONTROLS. The <u>least</u> sensitive input is No. 3, PROCESSES. Note that these sensitivity values are normalized by dividing each one by the sum of all calculated sensitivity values.

The ninth and final block of information written to the Capture File contains the early alert information. The first line of the block is a header that indicates the number of early alerts that follow. The first line of each early alert specifies the name of the alert converted to upper case. This name is followed by four values that represent the possibilistic number calculated for the result of this early alert. The early alerts are listed in the order in which they are defined within the user's Keyword Input File.

```
    3   Early Alert Equations
A1               0.1963 0.1992 0.3275 0.3299
Begin
   A1  =  (  ABNORMAL-EN-RISK  )  &  (  IMPLEMENTATION  )
End
A2               0.2000 0.2000 0.3000 0.3000
Begin
   A2  =  (  EXPERIENCE  )  |  (  SKILL  )
End
A3               0.6000 0.6500 0.7000 0.7000
Begin
   A3  =  (  CULTURE  )  |  (  TURNOVER  )
End
```

The early alert equation follows the line containing the calculated results. As shown above, each equation is delimited by the tokens "Begin" and "End". Delimiters are required because the equation will be written using as many lines as necessary for completion.

Addendum B

Filter File Format

As indicated in Part III, presence of the keyword statement "FILTER = Yes" in the user's Keyword Input File will instruct the program to generate a file of filtered Markov results. Once again, the Capture File is read as input for the filtering process, and the name of the Filter File is always FILTERED-MARKOV-DATA.DAT. It is also worth repeating that any existing version of this file will be overwritten.

The filtering process is simply the averaging of corresponding result values over the range of unique dates found in the date/time headers of the Capture File. If two or more dates are identical, then results for that date are averaged before any actual filtering begins. Interim days missing in the chronological sequence are interpolated before filtering begins. In all cases, the filtering process ignores the time field found in the date/time header; it serves only to indicate the time of execution if the program is run more than once on the same day.

Although shown separately in this Addendum for ease of reference, the main file header and all four information blocks are written to the Filter File as a contiguous stream of text without intervening blank lines or comments. Also, if more than a single plot segment has been written to the Filter File, the segment header for subsequent segments will immediately follow the early alert information block of the preceding segment. These items and all other information blocks will be described in their order of appearance in the Filter File.

The first line written to the Filter File is the main file header. It defines the number of plot segments that follow in the file. Although they are referred to as plot segments, the information can be used for any type of post-processing operation desired.

```
     1  (No. of Plot Segments to follow)
```

The next line in the file is a segment header. It describes the range of dates that were averaged in determining the results that will follow in the segment. This line is written using a fixed column format. This makes it easy for any program that must read the Filter File to extract any required field from this header.

```
Plot Segment    1 averages    11 Days:  05/02/2001 - 05/12/2001
```

The first block of information contains primary input information. The first line in the block indicates the number of primary input records that are to follow. Each record contains seven fields associated with each primary input.

```
   17  Pri-Inputs  (1)   (2)   (3)   (4)    Imp.      Sens.
POLICY           0.800 0.800 0.900 0.900  3.46E-02  6.21E-03
COMPLIANCE       0.500 0.620 0.700 0.900  2.76E-02  1.31E-02
```

```
TURNOVER         0.300 0.300 0.350 0.400  6.50E-03  1.38E-02
CULTURE          0.700 0.700 0.800 0.800  9.97E-02  1.78E-02
COMUNICATION     0.300 0.300 0.400 0.400  6.67E-02  4.61E-02
TRAINING         0.400 0.400 0.600 0.600  7.23E-02  2.02E-02
GUIDANCE         0.900 0.900 0.900 0.900  1.68E-02  1.47E-02
CONFIG-CONTROL   0.900 0.900 0.900 0.900  1.68E-02  1.47E-02
INTERNAL-REVIEW  0.600 0.600 0.800 0.800  1.25E-02  4.11E-02
EXTERNAL-REVIEW  0.700 0.700 0.800 0.800  1.36E-02  3.49E-02
EXPERIENCE       0.700 0.700 0.800 0.800  3.95E-02  1.02E-02
SKILL            0.800 0.800 0.900 0.900  4.50E-02  6.40E-02
PROCESS-COMPLI.  0.300 0.300 0.500 0.500  1.27E-02  1.51E-02
DOD-CONTROLS     0.400 0.400 0.600 0.600  0.10      0.10
HUMAN-RISK       0.300 0.300 0.700 0.700  3.44E-02  3.55E-02
NORMAL-EN-RISK   0.100 0.100 0.200 0.200  1.65E-02  6.33E-02
ABNORMAL-EN-RISK 0.400 0.400 0.600 0.600  5.72E-02  4.93E-02
```

The first field is the name of the primary input converted to upper case. The next four fields represent the possibilistic number assigned to the input by the user, except that they are averaged for the time period shown in the plot segment header. As indicated by the column labels in the plot segment header, the sixth and seventh fields respectively are the primary importance and sensitivity values averaged for each primary input.

The second block of information contains module output information. The first line in the block indicates the number of module output records that are to follow. The first of five fields is the name of the output converted to upper case.

```
    4  Module Outputs
MANAGEMENT       0.432 0.448 0.592 0.621
PROCESSES        0.757 0.760 0.838 0.842
IMPLEMENTATION   0.670 0.673 0.801 0.804
DOD-USE          0.407 0.408 0.647 0.648
```

The last four fields of the module output block represent the possibilistic number values averaged for each output of the module.

The third block of information contains the averages of the secondary importance and sensitivity values of each input to each module. The first line in the block is a header that indicates the number of sub-blocks (modules) that follow.

```
    4  Module Groups of Secondary Imp. & Sens.
    6  Values of:    Imp.    Sens.   for MANAGEMENT
POLICY            1.16E-02  5.19E-02
COMPLIANCE        9.29E-02  1.10E-02
TURNOVER          2.23E-02  1.16E-02
CULTURE           3.17E-02  1.50E-02
COMUNICATION      2.17E-02  4.00E-02
TRAINING          2.35E-02  1.71E-02
    5  Values of:    Imp.    Sens.   for PROCESSES
GUIDANCE          2.45E-02  8.81E-02
CONFIG-CONTROL    2.45E-02  8.81E-02
INTERNAL-REVIEW   1.82E-02  2.47E-02
EXTERNAL-REVIEW   1.98E-02  2.09E-02
MANAGEMENT        1.30E-02  3.68E-02
    5  Values of:    Imp.    Sens.   for IMPLEMENTATION
EXPERIENCE        3.34E-02  2.46E-02
SKILL             3.82E-02  1.54E-02
PROCESS-COMPLI.   1.08E-02  3.66E-02
MANAGEMENT        6.84E-02  1.63E-02
PROCESSES         1.08E-02  7.03E-02
    6  Values of:    Imp.    Sens.   for DOD-USE
DOD-CONTROLS      2.76E-02  3.35E-02
HUMAN-RISK        9.50E-02  1.19E-02
PROCESSES         1.52E-02  4.75E-02
IMPLEMENTATION    2.74E-02  1.22E-02
NORMAL-EN-RISK    4.56E-02  2.12E-02
ABNORMAL-EN-RISK  1.58E-02  1.65E-02
```

The first line in a sub-block indicates the number of input records that are to follow for that particular sub-block. As indicated above, the name of each module appears at the end of this line. Within a sub-block, the first of three fields is the name of the input converted to upper case. Note that this name can represent either a primary or a secondary input, because both types affect the secondary importance and sensitivity values. One obvious exception to this is the first defined module, which cannot have secondary inputs. The second and third fields respectively represent the secondary importance and sensitivity values averaged for each input.

The fourth and final block of information written to the Filter File contains the early alert information. The first line of the block is a header that indicates the number of early alerts that follow. The first line of each early alert specifies the name of the alert converted to upper case. This name is followed by four averaged values that represent the possibilistic number calculated for the result of this early alert.

```
    3  Early Alerts
A1              0.196 0.199 0.328 0.330
A2              0.200 0.200 0.300 0.300
A3              0.600 0.650 0.700 0.700
```

The example used in this Addendum spans a time period smaller than the filtering constant, and therefore generates only a single plot segment. However, in general, several plot segments are normally generated.