2008-09

Computer aided method for system safety and reliability assessments

Roycroft, Steven Michael.

Monterey, California. Naval Postgraduate School

http://hdl.handle.net/10945/3966

NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

COMPUTER AIDED METHOD FOR SYSTEM SAFETY AND RELIABILITY ASSESSMENTS

by

Steven Michael Roycroft

September 2008

Thesis Advisor: Mark M. Rhoades
Co-advisor: Louis C. Huang

Approved for public release; distribution is unlimited

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE	*Form Approved OMB No. 0704-0188*

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.

1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** September 2008	**3. REPORT TYPE AND DATES COVERED** Master's Thesis

4. TITLE AND SUBTITLE Computer Aided Method for System Safety and Reliability Assessments	**5. FUNDING NUMBERS**
6. AUTHOR(S) Roycroft, Steven M.	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000	**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) SMC/CV, BGen Susan Mashiko, USAF Los Angeles AFB, CA 90245	**10. SPONSORING/MONITORING AGENCY REPORT NUMBER**

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.	**12b. DISTRIBUTION CODE** A

13. ABSTRACT (maximum 200 words)

The objective of this study was to determine a methodology for implementing feedback loops into a logical, automated, computer assisted probability assessment tool. A tool exists called the GO program, which allows for systems to be modeled in a block diagram or schematic format and then analyzed in a structured manner to determine the probabilities of outcome events. The challenge was to incorporate a method for analyzing feedback loops. Given the difficulty involved with using computer code to analyze feedback loops, reliability engineers would normally create two separate models. To allow for a single model to be used and achieve consistent and repeatable results, a methodology for creating multiple layers of feedback loops in increasing complexity has been analyzed for use with the GO program. Monte Carlo simulations for each of these representative models have been constructed and analyzed to validate the adequacy of the GO program to effectively create probabilities of event success and failure. With the demonstrated ability of the GO program to correctly model feedback loops, it clears a path for the Department of Defense to investigate the benefits of adopting a standardized approach for the analyses of complex systems.

14. SUBJECT TERMS Reliability, Safety Assessments	**15. NUMBER OF PAGES** 95
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UU

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

COMPUTER AIDED METHOD FOR SYSTEM SAFETY AND RELIABILITY ASSESSMENTS

Steven M. Roycroft
Captain, United States Air Force
B.S., North Carolina State University, 2000

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN SYSTEMS ENGINEERING

from the

**NAVAL POSTGRADUATE SCHOOL
September 2008**

Author: Steven M. Roycroft

Approved by: Mark M. Rhoades
Thesis Advisor

Louis C. Huang, Ph.D.
Co-Advisor

David H. Olwell, Ph.D.
Chairman, Department of Systems Engineering

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

The objective of this study was to determine a methodology for implementing feedback loops into a logical, automated, computer assisted probability assessment tool. A tool exists called the GO program, which allows for systems to be modeled in a block diagram or schematic format and then analyzed in a structured manner to determine the probabilities of outcome events. The challenge was to incorporate a method for analyzing feedback loops. Given the difficulty involved with using computer code to analyze feedback loops, reliability engineers would normally create two separate models. To allow for a single model to be used and achieve consistent and repeatable results, a methodology for creating multiple layers of feedback loops in increasing complexity has been analyzed for use with the GO program. Monte Carlo simulations for each of these representative models have been constructed and analyzed to validate the adequacy of the GO program to effectively create probabilities of event success and failure. With the demonstrated ability of the GO program to correctly model feedback loops, it clears a path for the Department of Defense to investigate the benefits of adopting a standardized approach for the analyses of complex systems.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF FIGURES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF TABLES

EXECUTIVE SUMMARY

The purpose of this study was to determine a methodology for implementing feedback loops into a logical, automated, computer assisted probability assessment tool. The GO program allows for systems to be modeled in a block diagram or schematic format and then analyzed in a structured manner to determine the probabilities of outcome events. The challenge was incorporating a method for analyzing feedback loops. Feedback loops are difficult to analyze with computer code because feedback loops often have elements in the loop that are not independent events. Dependent events, when encountered, need to be accounted for in advance when they are incorporated within a block or schematic diagram. When processing data, computer code cannot logically process a failure, which represents the feedback, and a return to the beginning sequence to try a different path, which represents the loop. Given the difficulty involved with using computer code to analyze feedback loops, reliability engineers would normally create two separate models. One model would be used to determine the system probability and another model to estimate the feedback loops. These models would then be combined in the best manner possible given the background and experience of the engineer.

To allow for a single model to be used and achieve consistent and repeatable results, a methodology for creating multiple layers of feedback loops in increasing complexity has been analyzed for use with the GO program. Monte Carlo simulations for each of these representative models have been constructed and analyzed to validate the adequacy and accuracy of the GO program to effectively create probabilities of event success and failure when dependent feedback loop are considered as part of a single modeling effort and analysis of a system.

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

I want to thank several people who were instrumental in my completion of this educational journey. I would not be doing any justice if I were not to mention the dedication, guidance, encouragement and mentoring received from Mr. Tyrone Jackson. I can not begin to enumerate the different ways he contributed to my ability to successfully navigate my way through the thesis. In addition, I would be remiss if I did not mention my great appreciation to both Mr. Mark Rhoades and to Dr. Louis Huang, my thesis advisor and co-advisor respectively. Without their combined aid and sponsorship, I may still be in the hunt for a challenging thesis topic. In my time at SMC, I had several supervisors. Major Gene Cummins was my first supervisor overlapping my time with Naval Postgraduate School and suffice it to say, he provided endless encouragement to me and allowed for my seemingly endless requests for leave so I may work on the thesis (or homework or studying or fill in the blank for any education related need). Major Steven Lindemuth inherited the leave requests when he became my next supervisor and I am grateful he too understood the demands I was faced with as I attempted to balance my day job with school and family. Finally, I must say thank you to SMC for sponsoring this coursework and providing an avenue for myself and many others to embark on an educational journey that was at times quite exciting and as well quite challenging.

My last paragraph is dedicated to my son Nicolas and my wife Barbara who have experienced the costs of working full time and trying to do school simultaneously from the family side of the equation. Thank you for your sacrifices; the time you selflessly gave to me as it was projects, telephone conferences with groups and partners, study time, homework time, reviews, tests, reports and sometimes just one slow morning or afternoon so that I can clear my head and move on to the next item. I recognize that I was not alone in this effort, both of you were there with me, loving, supporting, understanding and allowing me to have this terrific opportunity. Without both of you supporting me, this would have been a difficult task indeed and I am blessed and thankful for your help and understanding.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

A. BACKGROUND

GO, an event sequence analysis tool, has applications in the disciplines of reliability, availability, safety and risk assessment. The GO Program was first developed in FORTRAN program code and ran on VAX IBM machines, it was later further updated to operate on personal computers. The GO program was initially developed by Kaman Sciences Corporation with funding provided by the US Army in 1967. Input to the GO Program is a functional and logical model called the GO Model. This model is a textual description of an event sequence diagram. Event sequence diagrams are constructed using different types of modeling elements. These modeling elements represent the functions of components and assemblies, including simple and complex components, logical OR and AND gates, contacts, signal generators, etc. These modeling elements also include probabilities of occurrence via the systematic processing of the modeled system by the GO Program. The GO Program evaluates each of the user-defined and user-assigned signal states of each of the modeled elements and will produce tables displaying the frequency or probability of each modeled element to be in the defined signal state. GO software is used to reduce any diagnostic problem to a deterministic solution, when the cause of a system malfunction is inferred from sensor information. GO analysis techniques are used to identify the cause, or potential causes, of system events or failures. System safety and reliability engineers are the typical users of this type of information and would most directly benefit from the data generated from GO analyses. As such, the input data would be developed and provided by safety and reliability engineers.

B. PURPOSE

The purpose of this study was to determine a methodology for implementing feedback loops into a logical, automated, computer assisted probability assessment tool. The GO program allows for systems to be modeled in a block diagram or schematic format and then analyzed in a structured manner to determine the probabilities of outcome events. The challenge was incorporating a method for analyzing feedback loops. Feedback loops are difficult to analyze with computer code as feedback loops will often have elements in the loop which are not independent events. These dependent events, when encountered, need to be accounted for in advance when they are incorporated within a block or schematic diagram. When processing data, computer code cannot logically process a failure, which represents the feedback, and a return to the beginning sequence to try a different path, which represents the loop. Given the difficulty involved with using computer code to analyze feedback loops, reliability engineers would normally create two separate models. One model would be used to determine the system probability and another model to estimate the feedback loops. These models would then be combined in the best manner possible given the background and experience of the engineer.

To allow for a single model to be used and achieve consistent and repeatable results, a methodology for creating multiple layers of feedback loops in increasing complexity has been created for use with the GO program. Monte Carlo simulations for each of these representative models have been constructed and analyzed to validate the adequacy and accuracy of the GO program to effectively create probabilities of event success and failure when dependent feedback loop are considered as part of a single modeling effort and analysis of a system.

C. RESEARCH QUESTIONS

Can the GO program be revised to allow concurrent performance of safety and reliability assessments on dynamic systems which include output feedback control functions? To be able to answer this question, a foundation of the operation of the GO program had to be established; starting first with a simple and straightforward example of how the GO computer code processes data and arrives at a probability of success and failure. This example would be supported by an accepted method of calculating the same probability of success and failure manually. After this initial step, a feedback loop example with a single feedback element would then be introduced. To further validate the GO analyses, a Monte Carlo simulation would be designed to independently verify the results of the analyses. Next, a feedback loop example with two feedback elements and another with three feedback loops would be processed by GO and verified by Monte Carlo simulations. These investigations into increasingly complex model would determine the adequacy of the methodology used for designing and implementing feedback loops for concurrent performance of safety and reliability assessments on dynamic systems.

D. BENEFITS OF STUDY

There are multiple benefits to be derived from performing this study and verifying the results. Listed below are ten specific benefits from using the GO program and the methodology described in this paper to implement feedback loops to allow for concurrent performance of safety and reliability assessments on dynamic systems.

1. There are no commonly known commercial reliability modeling tools available that are driven by a functional model. Performing reliability modeling using mathematical models introduces errors when function-level probabilities of occurrence are involved.

2. There are no commonly known commercial FMECA (Failure Modes and Effects Criticality Analysis) tools available that are driven by a functional model. Performing FMECA manually using worksheets is likely to introduce errors when complex systems are involved.

3. There are no commonly known commercial event tree analysis tools available that are driven by a functional model. Performing event tree analysis manually using worksheets introduces errors when latent operating modes are involved.

4. There are no commonly known commercial fault tree analysis tools available that are driven by a functional model. Performing fault tree analysis manually using logic symbols introduce errors when complex systems are involved.

5. There are no commonly known commercial engineering tools available that integrate reliability modeling, FMECA, event sequence analysis, and fault tree analysis. Performing these analyses separately using different methods is labor-intensive and introduces errors when trying to integrate the results.

6. There is no commonly known commercial suite of integrated safety/reliability tools available. One of the benefits in a future study is the research involved in

defining an "enhanced GO" tool which allows integrating safety and reliability analyses via output which includes function-level failure cut-sets and function-level probabilities of success.

7. There are no commonly known commercial reliability modeling tools available that are driven by a functional model which is compliant with an industry approved standard. The AIAA S-102 Mission Assurance Standards Working Group is interested in coordinating the further development of an "enhanced GO" to create an Open Source tool which is driven by a functional model that is compliant with the S-102 Functional Diagram Modeling standard.

8. There are no commonly known commercial reliability modeling tools available which allow feedback to be included in the model. One of the topics in this study is the research involved in defining an "enhanced GO" which allows feedback to be included in the reliability model. GO automatically generates the reliability model from the functional model.

9. There are no commonly known commercial reliability modeling tools available which provide output in a format which is compliant with an industry approved standard. The AIAA S-102 Mission Assurance Standards Working Group is interested in coordinating the further development of an "enhanced GO" to create an Open Source tool which provides output which is compliant with the XML data element descriptions defined in the S-102 standards.

10. In the past 25 years, very few commercial reliability analysis tools were developed that are driven by functional models and which can perform multiple analyses. One of these tools is called MultiLinx, which was used in the NASA Crew Return Vehicle (CRV) program between 1998 and 2003. This tool was not marketed in the public domain after the CRV program ended. The other tool is called eXpress, and it is currently used by many companies to analyze the testability of a design. The eXpress tool is not intended to be used in safety design or reliability design assessments. An "enhanced GO" could become the affordable Open Source alternative which supports both safety design and reliability design assessments.

E. SCOPE

The scope of this thesis is the identification and validation of revisions to the GO program that would allow concurrent performance of safety and reliability assessments on dynamic systems that include output feedback control functions. This study will directly support Government reviewed and approved analyses methodologies which can then be shared with other government agencies and industry partners. This revision to GO provides a no cost alternative to evaluate the combined safety and reliability assessments of complex systems.

F. CHAPTER SUMMARY

This study begins with a literature review of the Event Sequence Analysis GO program and then a review of Fortran program code. Then appropriate test methodologies of sample systems and the generation of reports were developed and performed. The results of these tests guided the development of the program code.

Chapter II provides a review of the GO program and takes the reader through a history of the development. It also discusses how GO is used by engineers for analyses applications. Chapter III discusses the systems engineering disciplines of system safety and reliability engineering and details how the disciplines are related and the overlap which exists between them. Chapter IV delves into the research and analyses necessary to incorporate feedback loops into a model for use with the GO program. Chapter V describes how GO may be utilized by both reliability and system safety engineers in the performance of these discipline areas as a common analyses tool. Chapter VI presents conclusions and areas for further research.

THIS PAGE INTENTIONALLY LEFT BLANK

II. EVENT SEQUENCE ANALYSIS GO PROGRAM REVIEW

A. THE DEVELOPMENT OF GO

Bill Gately, Larry Williams and Don Stoddard, while working for Kaman Sciences Corporation, were the initial developers of the GO program. Their work began in 1967 with funding from the United Stated Army. Since 1967 the GO codes have been developed and refined continually. Several innovations, improved features and enhanced functionality of the GO program were developed in the 1970s, including data consistency checks, the development of new operators, and the incorporation of a data type called a supertype. The supertype operator allows a user to create complex operators that can be reused when developing block diagrams and translating the blocks into code.

When GO was first developed, the use and acceptance of GO was widespread as GO allowed for unique formulations and also enabled efficiencies in the modeling approach. As the number of users increased and applications of the GO program grew, Kaman Sciences Corporation realized a need to continue with updates and modifications to the code. Providing support and maintaining standardized versions became a necessity to support the growing number of experienced users.

Since the inception of GO in 1967 there have been 24 versions of the code created as identified in the table below.

VERSIONS OF THE GO CODES		
DATE	**NAME**	**CHARACTERIZATION**
APR 68	GOMAR68	Eleven Logical Operators, Hash Addressing
JUN 68	GOJUN68	---
APR 69	GOAP69	Type 9 & 11 Kind Data Changed, Sensitivity Runs, Format-free Data, Modular Programs, Time Points up to 9999 permitted
MAY 69	GOMAY69	Use of two computer words to store more active signals and handle larger problems
AUG 70	GOAUG70	---
1971	RANGO	Randomized GO, Component Beta Distributions
1972	GOCHK72	---
APR 74	GOAPR74	Data Checks, Signal Table
1974	XGO	100 Active Components, Automatic Signal Deletion, Extensive Error Checking, Perfect Component Case, Automatic Array Size Optimization, PMIN Reset, Types 5 & 11 Combined
1975	Version B	---
16 FEB 76	Version C	New Operators 11-15, Multiple Type 8 Delays, GO1 Signal Table, Developed with Public Funds
26 APR 76	Version D	Supertypes GO1, GO2, GO3, Printouts Modified, GO1 Signal Table, Developed with Public Funds
NOV 76	Version E	---
11 JAN 77		Preliminary Fault Finder
3 MAY 77	GO	Fault Finder SYSFILE, FF1, FF2
30 NOV 77	GO	Types 15 & 17
3 MAR 78		Version D as documented for EPRI, Master Program GOFF, Data Decks Control Sequence, Alpha Descriptors, New Type 4
1 DEC 78	GOFF	Program FG and GO4 Created
17 AUG 79	GO	Efficiency Update, New Program Structure, Procedures and CLISTS, LIBRARY GORUN
1 OCT 79	GO	Effect Evaluation EE1, EE2, EE3
20 MAR 80	GO	CDC Version Documented for EPRI
20 MAY 80	GO	IBM Version Documented for B&R, UP&L, EPRI
30 DEC 80	GO	IBM Version Enhanced at UCC, Dallas, Descriptors, Facility to Alter Array Sizes, Explanation of Use
1 SEP 82	KSCGO	Version 1.0 Proprietary to KSC, Both VAX and CYBER Versions, Developed directly from GO Version C 1976

Table 1. Versions of the GO Codes (From Kaman Sciences Corp., 1983).

Over the years, GO has been used in a variety of systems for analyses. Disciplines such as reliability, availability, safety and risk assessment have benefited from studies with GO in systems such as satellites, missiles, conventional and nuclear munitions, power plants, processing plants and a variety of other systems. GO benefits the user by the way it handles event development and the probability of occurrence associated with the events.

B. GO ANALYSES, PROCEDURES AND OPERATORS

To better understand the benefits of GO, an explanation of the principles of the different operators and how to encode these operators into a model for analysis is needed. To begin, GO operates in a sequence of eight programs. These programs are identified as GO1, GO2, GO3, FF1, FF2, EE1 EE2 and EE3. This study focuses on the application and use of GO1, GO2 and GO3. Some explanatory information will be presented regarding the FF and EE series, but no detailed information will be provided. The GO sequence 1-3, when executed, is used to find the probabilities of the different systems states of the modeled system. The FF series is a fault finder sequence of analyses. FF 1-2 is used to determine the sets of operators needed which will cause a selected event. The EE series 1-3, which is not being evaluated in this paper, is used to determine the effects on probabilities of system states with respect to uncertainties.

In each of the series of analyses (GO, FF or EE), the programs must be run in the numbered order. The results of each of the series are used in subsequent programs. One benefit of this sequential execution of the program is that once the data sets are generated from the GO series, the FF or EE series may be run multiple times without having to re-run the previous series. This allows for different events to be analyzed by the subsequent series with a single operation of a previous series.

GO1, GO2 and GO3 operate with data entered by the user. The GO1 input file contains the model data of operators. GO2 contains any associated probability information related to the operators. GO3 sets analysis parameters. Each of these input files have data validation checks performed when the associated GO execution program is executed. The checks are done to find most typing or logical errors. Although the programs are effective in finding most types of modeling element errors, care must be taken to prevent logical errors which are consistent with incorrect parameter inputs. As an example, the user may input data which defines two states for a signal generator, one in which the signal generated is a value of “0” 80 percent of the time and a value of “1” 20 percent of the time. If the user incorrectly assigns the percentages, GO will perform a check which will determine there are no typing or logical errors for the entry. Although the data are logically correct, the resulting analyses will produce accurate results for the inaccurately entered data. The GO program will first process the operator records with the GO1 program. GO1 will produce an operator file for use with the next phase of the program. GO2 will read and check the input file which contains the “kind” records with the probability data and will create a combined operator and kind file. The creators of the GO Program use the term “kind” to define references to different types or “kinds” of modeled elements. For example, a switch may be modeled within GO and depending on the manufacturer or type, the switch may have different probabilities of success and failure. The user will assign a kind value to each individual element within the model which has a probability of success and failure defined. If when modeling a system, the user repeats the application of a previously defined or kind element, the kind number does not need to be repeatedly referenced within the code. If a new “kind” element is used, a unique kind number will be assigned to the element and the associated probability data defined accordingly. With the operator and kind file created, GO3 is used to evaluate the event tree and will create a file which contains the results of the analysis. A discussion of parameter inputs will be made later in this paper.

GO operates with seventeen logical operators called types in modeling systems. These types are used so that systems may be effectively modeled in operation or with interactions within the system. A list of the seventeen operators is shown in a diagram in Figure 1. This figure shows all the types of operators and also shows elements of operator inputs which would be required when modeling the element in the GO code. Any inputs into an operator are called stimuli and given the nomenclature of S_1, S_2, …, S_n. The outputs are called responses and have the nomenclature of R_1, R_2, …, R_n. These inputs and outputs are random variables. When using GO models, the random variables are referred to as signals.

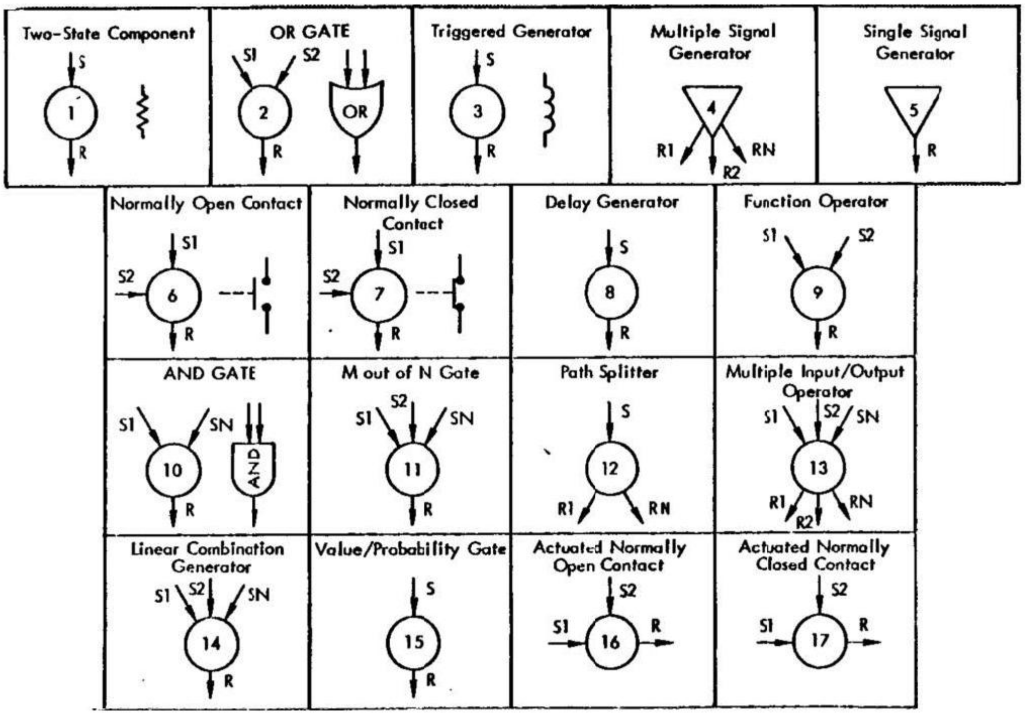


Figure 1. GO Type Operators (From Kaman Sciences Corp., 1983).

Each operator is used to model responses from equipment or of human actions within a system. Each of these operators may have probabilities associated with it and will evaluate the differing random variable inputs (S) and will generate the appropriate response (R).

An example of how the type operators are used follows. A type 1 operator is used to logically represent an element that will either perform or fail to perform the function given a correct input to the type operator. The type 2 operator is an OR gate and it will function just as a logical OR gate and will generate a response with the correct inputs.

Type 1	Two State Component
Type 2	OR Gate
Type 3	Triggered Generator
Type 4	Multiple Signal Generator
Type 5	Signal Generator
Type 6	Normally Open Contact
Type 7	Normally Closed Contact
Type 8	Increment Generator
Type 9	Function Operator
Type 10	AND Gate
Type 11	m out of n Gate
Type 12	Path splitter
Type 13	General Purpose, Multiple Input, Multiple Output Operator
Type 14	Linear Combination Generator
Type 15	Value/Probability Gate-Generator
Type 16	Actuated Normally Open Contact
Type 17	Actuated Normally Closed Contact

Table 2. GO Type Operators.

To successfully create a model in GO, the user must complete several steps.

1. Learn how the system is configured and actually operates.
2. Define system success and failure criteria.
3. Identify the system events about which information is sought.
4. Represent system elements with standardized GO operators.
5. Combine the inputs and outputs of operators representing system elements into a GO model portraying successful system operation.
6. Obtain the probabilistic data for component response. (Kaman Sciences Corporation., & Electric Power Research Institute, 1983)

The GO program uses a methodology of translating functional or block diagrams into a code and then generates the event development with the probability of success and failure associated with the events. To best illustrate the benefits of using GO methodology, first an understanding of what the industry standard definitions of reliability is required.

MIL-STD 721 defines reliability as: (1) The duration or probability of failure-free performance under stated conditions and (2) The probability that an item can perform its intended function for a specified interval under stated conditions.

IEEE defines reliability as “the characteristic of an item expressed by the probability that it will perform a required mission under stated conditions for a stated mission time” (IEEE Std 577-1976).

GO may be used to determine a system’s reliability performance measures. To best illustrate the ability of GO, a simple example will be used which demonstrates calculating reliability probabilities using a representative model and manual calculations. A familiar model is one in which a comparison is made between two systems, one which operates with a single generator and the other with two generators. Functional diagrams of a single system and a two generator system are displayed in Figures 2 and 3.

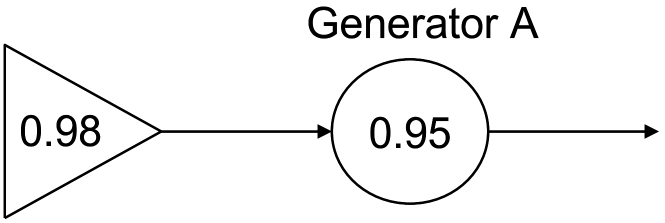


Figure 2.   Simple Reliability Model with One Generator.

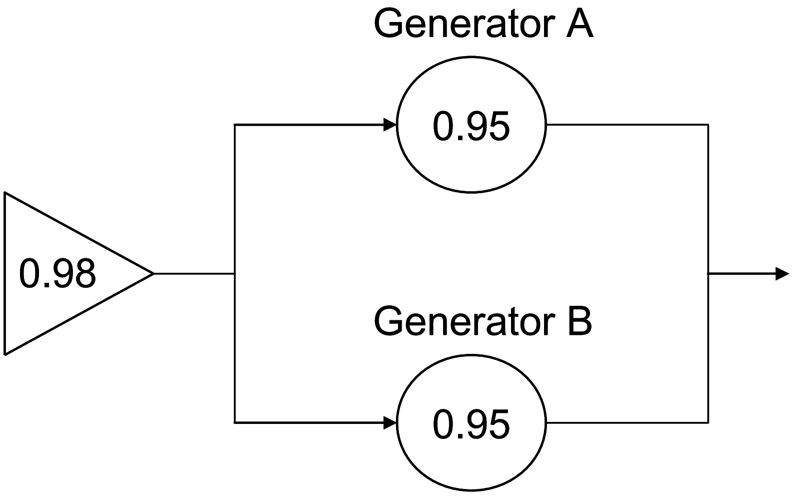


Figure 3.   Simple Reliability Model with Two Generators.

This system starts with a start signal with a probability of success of 98% which leads into the single generator in Figure 2 or the two generators of Figure 3. Both generators in Figure 3 are positioned in parallel providing that if one generator were to fail, the second generator would be available to continue the operation. All generators in both Figures 2 and 3 have a probability of success of 95%. To calculate the overall probability of success for the system in Figure 2, multiply the probability of success of the first signal times the probability of success of the generator. Multiplying 98% and 95% yields a probability of success of 93.1%. When two components operate in parallel, the probability calculation changes. The calculation of the parallel generators uses the following formula: PA + PB - (PA x PB). Substituting the values of 95% for both generators into the formula results in 0.95 + 0.95 - (0.95 x 0.95) which results in 1.90 - 0.9025, with a final result of 0.9975. This parallel combination of generators with a 95% probability of success for both generators yields an overall probability of success of 99.75%. To complete the calculation of a two generator system as depicted in Figure 3, the probability of success for the first signal is multiplied by the probability of success of the parallel generators. Multiplying 98% and 99.75 percent yields a probability of success of 97.755%.

The results of the manual calculations of the parallel generator system will now be compared to the results of a two generator system as modeled with the GO program. Following the demonstration of the results generated by the GO program is an explanation of the code which was used to generate the GO results. To begin, Figure 4 shows how a functional model would be developed for use with GO. The reader will note the numbering used in each of the symbols used in Figure 4. The first number (or in the case of the type 2 OR gate, the only number) represents the type of operator being used, while the second number is a uniquely assigned identifier called a kind number. This numbering convention helps the user identify and track the components when translating from a graphical or schematic model to the code which will be used as inputs to the GO program. Additional information related to operator types and kinds will be provided in more detail further in this section.

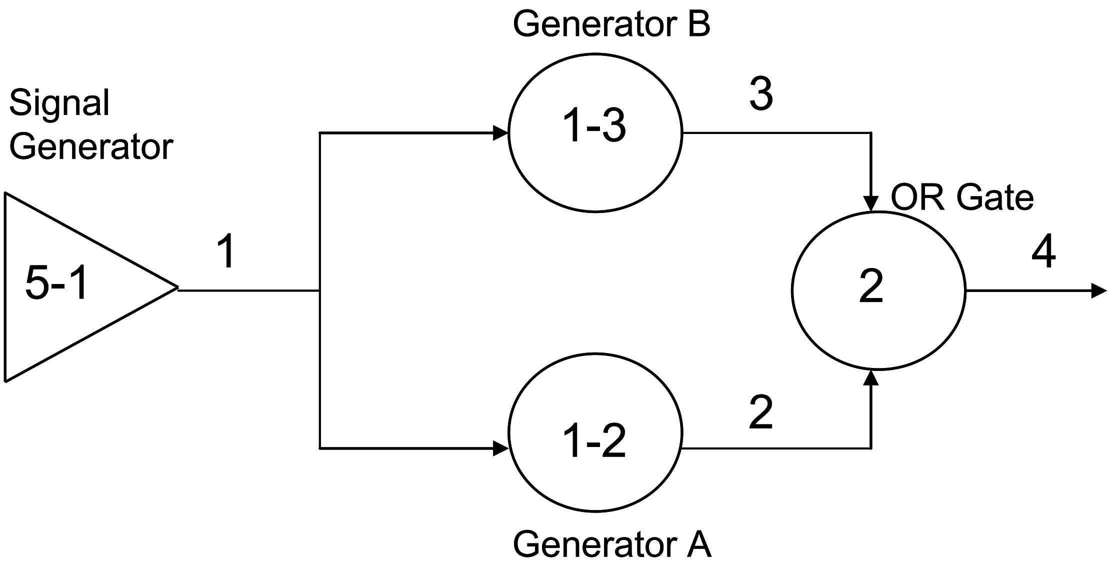


Figure 4. Simple Reliability Model with Two Generators as Modeled with GO.

GO uses several different operators in the computer code to model function or behavior of the elements which make up a system. The first signal is represented by a type 5 operator which is a signal generator. Generators A and B are represented by a type 1 operator. The last operator used in this model is a type 2 which is an OR gate. The OR gate reads the signals present on the inputs and according to the logic of an OR gate will generate an output. When this model of a two generator system operating in parallel is processed by the GO program, it generates a table of results which is presented below in Table 3, where the numbers shown on the left are the states of the system and the numbers along the top of the table are the signals. State 1 is the operational state and State 4 shows the failed state. As you can see, there is good agreement between the operational reliability results shown under signal 4 and the manual calculation.

VAL.	1	2	3	4
----	------------	------------	------------	------------
1	0.97999996	0.93099993	0.93099993	0.97754997
4	0.02000000	0.06900001	0.06900001	0.02245000

Table 3. GO Probability of Events.

C. PROGRAMMING IN GO

Each of the operator types requires modeling data to be input according to the type being used. Table 4 below shows each operator type and the different modeling information required for modeling the type in GO code.

Data 1 Operator Type	Data 2	Data 3	Data 4	Data 5	Data 6
1	K	S	R		
2	0	n	$S_1 \ldots S_n$	R	
3	K	S	R		
4	K	n	$R_1 \ldots R_n$		
5	K	R			
6	K	S_1	S_2	R	
7	K	S_1	S_2	R	
8	K	S	R		
9	K	S_1	S_2	R	
10	0	n	$S_1 \ldots S_n$	R	
11	m	n	$S_1 \ldots S_n$	R	
12	K	S	m	$R_1 \ldots R_n$	
13	K	n	$S_1 \ldots S_n$	m	$R_1 \ldots R_n$
14	K	n	$S_1 \ldots S_n$	R	
15	K	S	R		
16	K	S_1	S_2	R	
17	K	S_1	S_2	R	
K: KIND NUMBER					
S: STIMULUS (INPUT)					
R: RESPONSE (OUTPUT)					

Table 4. GO1 Operator Type Data.

Table 5 contains the input code and data which were used to create the GO1 input code as represented in Figure 4.

Line 1	GO1 DATA FOR PARALLEL RELIABILITY SYSTEM
Line 2	,,,,,4/
Line 3	5 1 1 $ SIGNAL GENERATOR
Line 4	1 2 1 2 $ GENERATOR A
Line 5	1 3 1 3 $ GENERATOR B
Line 6	2 0 2 2 3 4 $ OR GATE
Line 7	0 1 2 3 4 $ FINAL SIGNALS
Line 8	EOR

Table 5. GO1 Code for Parallel Reliability System.

Line 1 identifies the title of the GO1 code. Line 2 can be edited to change default system parameters. The number 4 is a user defined value identifying what value the failure state for the model should be. The number 4 was chosen in this model, but it could have been any number value which was not being used as a value for a signal within the system. For the purposes of this study, default system parameters for the GO program will not be discussed other than the assigned failure state value, using the system defaults is sufficient for the purposes of analyses of GO and demonstrating an effective use of feedback loops.

Line 3 of the code describes the operator as a type 5 operator and is assigned a kind (K) of 1, a response (R) of 1 and following the $ character is a description or nomenclature of Generator. The kind number is used to identify the unique characteristics of this type operator. In addition a different kind number should be assigned to each operator type used in a model as the kind number is linked from data included in the GO1 input to probability data which would be included with GO2 input data. If multiple type 5 operators are used and the characteristics are the same in the use of this type within the code, only one type 5 operator with an assigned kind would need to be defined within the GO1 input code. On the other hand if multiple type 5 operators are used and there are characteristics of the type 5 operator that differed on one or more operators, then a unique kind number would be assigned to each of the different type operators. The response (R) of 1 identifies that a single response will be generated as an output signal. The nomenclature following the $ symbol identifies this type 5 operator as a generator.

Line 4 of the code describes the operator as a type 1 operator and is assigned a kind (K) of 2, a stimulus (S) of 1, a response (R) of 2 and following the $ character is a description or nomenclature of Generator A. The stimulus of 1 indicates there is an input stimulus which comes from signal 1 (as shown in Figure 4); the response (R) of 2 identifies the response will be output onto signal 2. The nomenclature following the $ symbol identifies this operator as Generator A.

Line 5 of the code describes the operator as a type 1 operator and is assigned a kind (K) of 3, a stimulus (S) of 1, a response (R) of 3 and following the $ character is a description or nomenclature of Generator B. The stimulus of 1 indicates there is an input stimulus which that comes from signal 1; the response (R) of 3 identifies the response will be output onto signal 3. The nomenclature following the $ symbol identifies this operator as Generator B.

Line 6 of the code describes the operator as a type 2 operator and is assigned a kind (K) of 0, the number of inputs (n) is set as 2, a stimulus (S1) of 2, a stimulus (S2) of 3, a response (R) of 4 and following the $ character is a description or nomenclature of OR Gate. This kind value is different from any of the other kind values which have been assigned so far. OR gates as well as AND gates (type 10) when used in GO modeling are considered to be type operators with perfect kinds. When the code encounters a type operator with a perfect kind, the probability of success for the operator is 100 percent. The data type n of 2 identifies this OR gate has two stimulus inputs. The stimulus of S1 of 2 indicates there is an input stimulus which will come from signal 2, the stimulus Sn (S2) of 4 indicates the second and last input stimulus will come from signal 4. The response (R) of 5 identifies the response will be output onto signal 5. The nomenclature following the $ symbol identifies this operator as an OR gate.

Line 7 of the code starts with a zero which is an indicator to the program the final signals will be identified. This is where the safety or reliability engineer will identify which signals the code will output the probability information associated with the individual signals. In this use, all signals (1, 2, 3 and 4) used in the model will have the associated probability data output. The nomenclature following the $ symbol identifies the line as the Final Signals line of code. Line 8 is the terminator of EOR. This is an indicator to the GO program the inputs of GO1 are complete and the last line of the record has been reached.

As indicated in the explanations of the GO1 input code, there are probability data associated with some of the operators used in GO1. The probability data are included in the GO2 input file and are presented in Table 6.

Data 1	Data 2 Operator Type	Data 3	Data 4	Data 5	Data 6	Data 7	Data 8
K	1	P_g	P_b				
K	3	P_g	P_b	P_p			
K	4	n	m	$V_{11}…V_{1n}$	P_1		
				$V_{21}…V_{2n}$	P_2		
				⋮			
				$V_{m1}…V_{mn}$	P_m		
K	5	n	V_1	$P_1…V_n$	P_n		
K	6	P_g	P_b	P_p			
K	7	P_g	P_b	P_p			
K	8	n	D_1	$P_1…D_n$	P_n		
K	9	n	X_1	$Y_1…X_n$	Y_n		
K	12	m	$P_1…P_m$				
K	13	n	m	N			
	$VS_{11}…VS_{n1}$	M1					
		$VR_1…VR_m$	P_{11}				
		⋮					
		$VR_1…VR_m$	P_{M11}				
	⋮						
	$VS_{1N}…VS_{nN}$	M_N					
		$VR_1…VR_m$	P_{1N}				
		⋮					
		$VR_1…VR_m$	P_{MnN}				
K	14	n	$a_1…a_n$	a_0			
K	15	V_1	V_2	V_3	V_4	P_1	P_2
K	16	P_1	P_2	P_3			
K	17	P_1	P_2	P_3			
K: Kind Number				b: Probability Bad			
P: Probability				p: Probability Premature			
V: Value				n: Number of Inputs; 0<=n<=10			
D: Delay				m: Number of Outputs; 1<=m<=10			
g: Probability Good				N: Number of Output Sets			

Table 6. GO2 Kind Data.

Table 7 contains the input code and data which were used to create the GO2 input code as represented in Figure 4.

Line 1	GO2 DATA FOR PARALLEL RELIABILITY SYSTEM
Line 2	0/
Line 3	1 5 2 1 0.98 4 0.02 $ SIGNAL GENERATOR
Line 4	2 1 0.95 0.05 $ GENERATOR A
Line 5	4 1 0.95 0.05 $ GENERATOR B

Table 7. GO2 Code for Parallel Reliability System.

Line 1 identifies the title of the GO2 code. Line 2 can be edited to change default system parameters. Line 3 lists Kind 1 for the type 5 operator. The number 2 represents the n number of outputs. The number 1 is the first user defined value V1 of the output signal which will be created by the type 5 operator. The value will occur with a probability percentage P1 of 0.98. The next number is the value V2 (Vn) 4, which is for this model the failure value chosen by the user in Line 2 of the GO1 input code. The V2 value will occur with a probability of 0.02. The nomenclature following the $ symbol identifies this operator as a Generator.

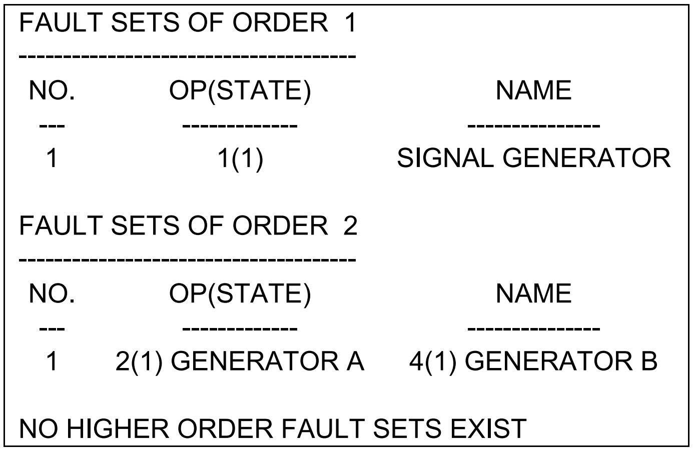
Although each of the remaining lines could be explained number by number, the intent behind this information was to familiarize the reader with the method by which a system is modeled graphically for GO and how the graphical information is then interpreted into the GO1 and GO2 code. The remaining lines for the GO2 table follow the same pattern as described in Table 6 for inputting data to support the GO program analyses. For reference, the code for GO3 is included below in Table 8. As one can see, the information necessary for creating the GO3 input is minimal. Line 1 identifies the title of the GO3 code. Line 2 can be edited to change the default system parameters. Line 3 is the terminator of EOR.

Line 1	GO3 DATA FOR PARALLEL RELIABILITY SYSTEM
Line 2	,,,1,,,,128/
Line 3	EOR

Table 8. GO3 Code for Parallel Reliability System.

GO has the capability of having a failure state value assigned. In this example, the success state was assigned the value of 1, the failure state was assigned the value of 4. Referencing Figure 3 above, GO produced the probability of success for each of the connectors used in the dual generator system. After the OR gate operator, signal 5 shows a probability of success of 0.97754991 which matches the probability of success results determined by manual calculations. With this foundation of how GO can be used as an analysis tool in place, later in this paper, a further exploration of using GO to analyze more complex systems will be developed.

In addition to being able to produce tables showing probabilities of event success and failure, GO can perform a fault finder analysis. GO uses algorithms to determine which functional or component failures would have to happen for an identified signal to experience a complete failure. An example of a fault finder analysis on signal 5 is given below in Table 9.



```
FAULT SETS OF ORDER  1
-------------------------------------
 NO.          OP(STATE)                  NAME
  ---          -------------           ---------------
   1                1(1)           SIGNAL GENERATOR

FAULT SETS OF ORDER  2
-------------------------------------
 NO.          OP(STATE)                  NAME
  ---          -------------           ---------------
   1      2(1) GENERATOR A     4(1) GENERATOR B

NO HIGHER ORDER FAULT SETS EXIST
```

Table 9. GO Fault Finder Results.

As can be seen, the signal generator is the only single component which would result in a complete failure at signal 5. If two failures were being investigated, there is one set of faults which would result in failure, both generator A and B failing would result in failure. This example is an easy one to evaluate and can be done visually. In more complex systems, having a computer perform the evaluation is useful as the analysis will be performed quickly and completely, especially if fault sets of higher orders are being evaluated or in cases where there are many components.

With this understanding of how GO operates to analyze a simple reliability model of two generators the next step is to introduce the reader to feedback loops and how GO can be utilized to perform an analyses which can support both reliability and system safety assessments with a single computerized tool. With the introduction of feedback loops, the capabilities of GO can be further expanded and used by a variety of users in the engineering community.

THIS PAGE INTENTIONALLY LEFT BLANK

III. SYSTEMS ENGINEERING AND THE ROLE OF SYSTEM SAFETY

A. INTRODUCTION

Systems engineering is defined by the International Council of Systems Engineering (INCOSE) website *incose.org* as, “Systems Engineering is an interdisciplinary approach and means to enable the realization of successful systems.” One of the mission assurance elements within systems engineering is the engineering discipline of system safety. *MIL-STD-882C* defines safety as, “Freedom from those conditions that can cause death, injury, occupational illness, or damage to or loss of equipment or property, or damage to the environment.” *MIL-STD-882C* defines system safety as, “The application of engineering and management principles, criteria, and techniques to optimize all aspects of safety within the constraints of operational effectiveness, time and cost throughout all phases of the system life cycle.” Often there is a misconception that system safety as an engineering entity is focused on protecting personnel from accidents by focusing on slips, trips, falls, and toxic chemicals/materials. In fact system safety incorporates the necessary requirements into product design to allow for safety to be an integral part of the design process and process hazard avoidance at every indenture level of the system.

System safety, as an engineering discipline, contributes to systems engineering objectives by applying a set of safety design requirements throughout all life cycle phases of a system, whether it is design, manufacturing, sustainment or at the end of service life. Safety design requirements follow a pre-established order of precedence. For example, *MIL-STD-882C* imposes a four-level order of precedence for safety design requirements. First in the *MIL-STD-882C* order of precedence is designing for minimum risk, which is accomplished by eliminating hazards through design and by reducing risk to acceptable levels throughout the design selection. Second in the *MIL-STD-882C* order of precedence is the incorporation of safety devices, e.g., power circuitry fuses and anomaly detection and response (ADR) software for fast-acting failure modes. Third in the *MIL-*

STD-882C order of precedence is providing warning devices. Finally, fourth in the *MIL-STD-882C* order of precedence is developing appropriate procedures and training. System safety also levies requirements on the system design philosophy in terms of fault tolerance requirements. For example, *MIL-STD-882C Appendix C, Section 70.1.1a*, requires mission-critical system functions to be able to continue operating after any single fault occurs within the system or within an external entity controlling the system. *MIL-STD-882C Appendix C, Section 70.1.1b* also requires safety-critical system functions to be able to continue operating after any dual independent faults occur within the system and/or within an external entity controlling the system. Due to the wide-spread practice of allowing high unit-value system designs, e.g., satellites and military aircraft, to minimize rather than eliminate all single-point failure modes and using mission-critical equipment for system "safing" functions, the practicality of verifying dual-fault tolerance is often limited to evaluating the failure of one mission-critical hardware, software, or procedural item and the failure of one responding safety-critical software item, e.g., an ADR algorithm. Due to multiple failures being involved, the safety-critical software item can be identified using the GO program, Fault Tree Analysis, or an equivalent "deductive" analysis methodology. Then software hazard analysis or software FMECA is performed to identify design, processing, or operating features that can either mitigate the root causes or compensate for the effects of each software functional failure mode.

B. OVERLAP BETWEEN SYSTEM SAFETY AND SYSTEM RELIABILITY ENGINEERING

In the last chapter, a simple example of how system reliability is calculated was presented. In this chapter, an explanation of the overlap which exists between system safety and reliability engineering will be provided. As discussed previously, system safety contributes to systems engineering through design influence and process hazard avoidance. For complex high unit-value systems even a minor design change can have a large impact on the cost and performance of the system. One of many systems engineering elements that contribute to the system design definition is reliability engineering. If a system has a requirement for the generation of power to have a mission reliability of 95%, the example single generator design configuration would not satisfy the reliability requirement, but the two generator design configuration would. If the same system has a system safety requirement for the generation of power to be dual-fault tolerant, the two generator design configuration would not satisfy the system safety requirement. One possible solution to the problem just described would be to introduce a third generator into the system.

It is important to emphasize that there are often overlapping requirements between the disciplines of system safety and reliability engineering. Aside from cost and performance, there are other design attributes which can be influenced by a design change. The reliability engineer will focus on the system design satisfying its mission reliability requirement through allocation of the system level reliability requirement to the subsystem level, assembly level, etc. In contrast, the system safety engineer will focus on the system design satisfying its dual-fault tolerance requirement for all safety-critical functions through reduction of the system design to a set of safety-critical functions. There is often a conflict between the reliability engineering effort and system safety effort, because the former has "flexibility" at the subsystem and lower levels as long as the system level design meets its quantitative requirements, while the latter must rigidly control "hazard risk" at relatively low functional hardware/software component levels. For example, hardware/software component single-point-failure-modes (SPFMs) have a greater impact on safety-critical design requirements (e.g., dual-fault tolerance) than on mission reliability-critical design requirements (e.g., single-fault tolerance). However, reliability engineers and system safety engineers are responsible for identifying and controlling some of the same types of faults, such as SPFMs. So there is a great deal of potential benefit from using common tools such as the GO program, among different engineering disciplines. These common tools would allow a complex system design to by simultaneously analyzed quickly in different ways and for detailed reports to be automatically generated to yield multiple solutions for further optimization or for identifying potential trade space within the system. With the increasing complexity of systems as technology advances, the need for the different engineering disciplines to work cooperatively to satisfy system requirements becomes more important. As such, the use of more common, powerful and robust computer aided tools becomes more important as well.

C. CHAPTER SUMMARY

There are many aspects within the disciplines of system safety and reliability which overlap. The example presented illustrates the relationship between the two disciplines and how the design implementation of one discipline may impact the other discipline in meeting an important requirement. The ability to use a common set of tools between the two disciplines would be helpful as each would receive benefit from tools which could be universally used, allowing for potential savings in work efforts and efficiencies in operations as a result.

The GO program is a tool that can be used for both system safety and reliability to analyze system design to ensure all requirements are being met and to assess the degree to which each requirement has been met. A fully defined process for incorporating feedback loops is necessary, however, so that GO can be fully support the needs of reliability engineers. Without a defined process for incorporating feedback loops, GO would continue to be beneficial for system safety engineers but would have limited benefit for reliability engineers. The limited benefit stems from the difficulty involved with using computer code to analyze feedback loops. When using a computer to analyze a circuit or a system, the calculations to arrive at a deterministic result are performed as the code processes through the system path. When feedback is involved, this causes a path, which had previously been analyzed, to require another iteration in the calculation process. Unfortunately, due to how reliability or probability calculations are performed, the previously used probability values can not be used and therefore require different calculations to arrive at the correct reliability or probability calculations. Currently reliability engineers typically use two separate models for determining the reliability of a system that has feedback. One model would be for the system and a separate model would be used to estimate the reliability involving the feedback loops. After the estimate was completed, the reliability engineer would then attempt to merge the two models in the best manner possible based on the engineer's own experience. One of the difficulties faced when separate models are created is the merging of the models. When a system is split to take into account different methods of modeling interactions and the associated probabilities of the interactions, there is the possibility of introducing error when the separate models are then combined. In addition, there are several different methods available for modeling interactions. One engineer may choose one method, while another engineer a different one. Given the opportunity for subjective choices to be made when modeling probabilities, this creates the opportunity for two qualified engineers to model the same system and come up with different results based on the modeling methodologies chosen. If GO could be used with feedback loops incorporated into the functional model, there would be no need to have two separate models created and then merged by the reliability engineer. In the next chapter of this paper, a fully defined methodology for incorporating feedback loops into a functional design is documented.

THIS PAGE INTENTIONALLY LEFT BLANK

IV. RESEARCH ANALYSIS

A. INTRODUCTION

In this section, the author will present a description of a single model based on an example problem, describe the functional behavior and describe how to incorporate a single feedback loop into the GO program for analysis. Once the single model has been described and each of the elements within the model documented, this paper will next introduce a standard model with two feedback loops as well as a complex model with three feedback loops. Each of these examples will serve to demonstrate what steps would be taken by an engineer to follow a standard methodology for performing reliability and safety analyses on systems.

Monte Carlo simulations for each of the examples were created. Monte Carlo simulations were chosen as a tool to use to validate the results of the GO program as Blanchard and Fabrycky state, ”Models and their manipulation (the process of simulation) are useful tools in systems analysis.” They further state, “A mathematical model employs the language of mathematics and, like other models, may be a description and then an explanation of the system it represents (Blanchard and Fabrycky, 2006). The Monte Carlo simulation was a natural selection to use for validating the model represented within the GO modeled code as GO is a logical and mathematical analyses application.

B. EXAMPLE PROBLEM

An engineer may be faced with a problem of how to model the behavior of a person faced with a set of actions to be taken. Each of the actions which could be taken will be called tasks. In the single model, the system is made up of tasks A, B and C. Each task has a probability of success and failure. The person performing the tasks has a number of choices which can be taken in accomplishing the tasks. In all cases, task A is to be accomplished first. In this study, it is preferred but not required that the tasks are accomplished in alphabetical order. The preferred order would follow that A is accomplished, then task B and with successful accomplishment of task B, then task C would be performed. Alternatively, task A could be performed and the choice could be made to skip task B and perform task C. Last, task A could be performed, task B attempted and a failure of task B experienced, which would then result in the person returning to perform task A again and then perform task C. In the single model, if a task failure is experienced at task A or task C, the system will result in a failure. It is important to note that this model could have been designed to allow the first task to be task B or task C, but allowing this complicates the model and does not aid in the purpose of demonstrating a methodology for incorporating feedback loops. The objective of this study was to determine a methodology for implementing feedback loops into a logical, automated, computer assisted probability assessment tool. There is not an intention to provide for a study which analyzes a comprehensive and exhaustive list of possible task path combinations. A feedback loop used in the context of this study represents an opportunity for an action which could be performed more than once and can be accounted for via the feedback loop methodology. In the case of this study, each task path which can be accomplished can be traced out in the functional block diagrams and the GO models.

Figures 5, 6, and 7 show the three models each in increasing complexity. These are functional models and, in the next section, will be shown in corresponding functional block diagrams that were used to create models with the GO program. As stated previously, the single model has tasks A, B and C. The standard model adds a task D and the complex model adds tasks D and E.

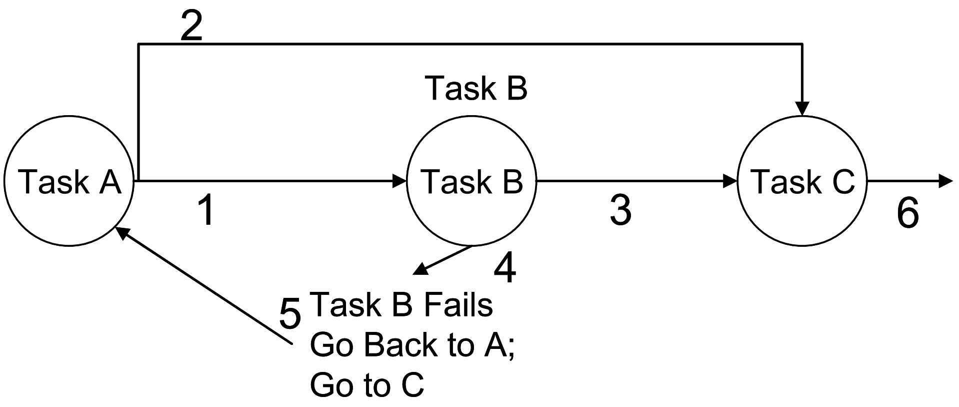


Figure 5. Functional Block Diagram of Single Model with Tasks A, B and C.

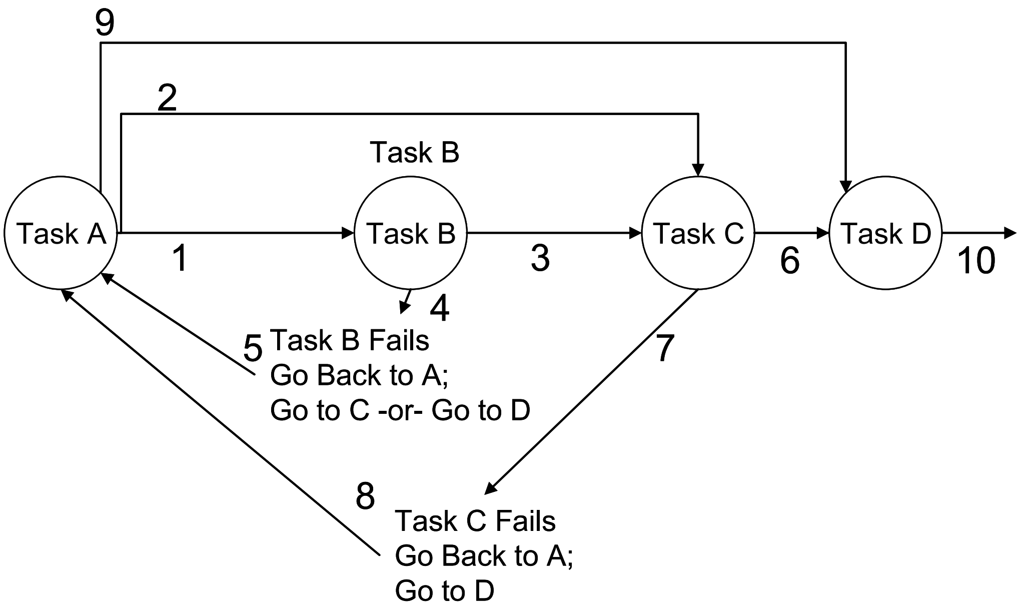


Figure 6. Functional Block Diagram of Standard Model with Tasks A, B, C and D.

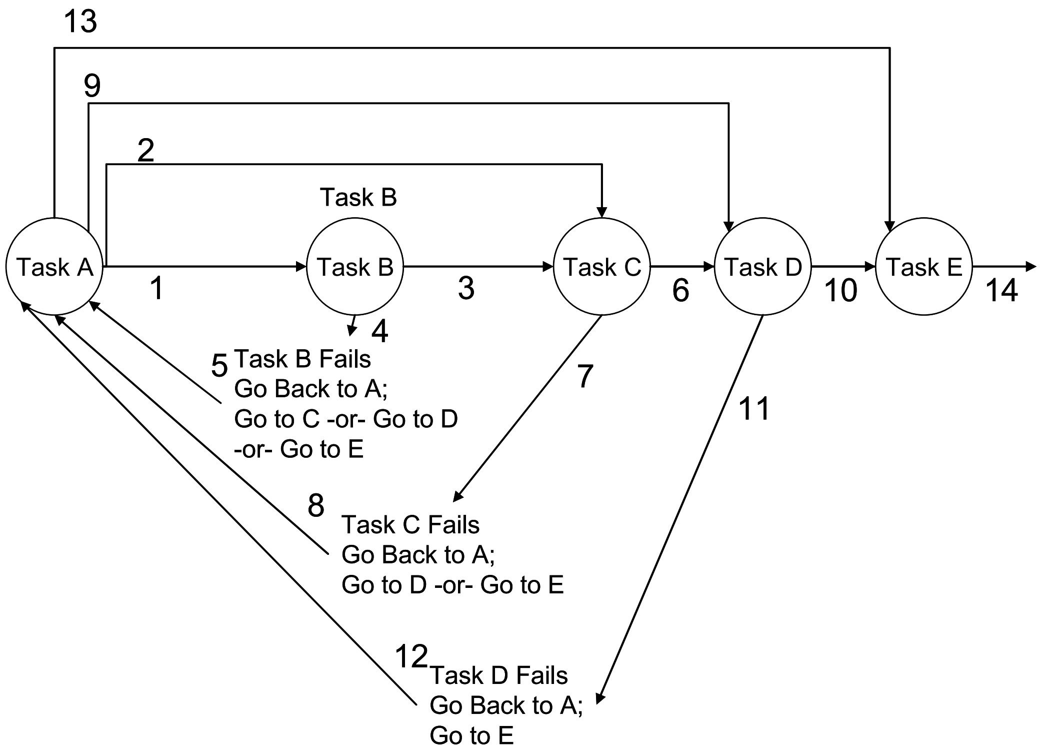


Figure 7. Functional Block Diagram of Complex Model with Tasks A, B, C, D and E.

C. EVENT SEQUENCE ANALYSIS (GO PROGRAM) SINGLE FEEDBACK LOOP IMPLEMENTATION

With an understanding of how to use the GO program, an engineer can take a functional block diagram and select the appropriate operators so that a representative model may be created to be analyzed by GO. In the case of the single model, the GO diagram is represented in Figure 8 below.

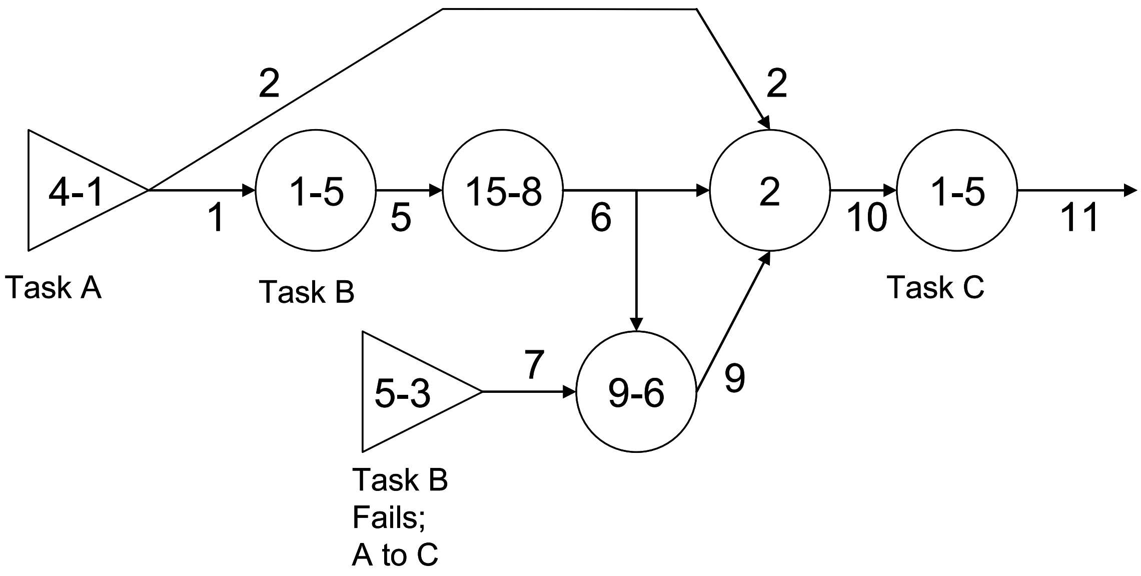


Figure 8. GO Diagram of Single Model with Tasks A, B, and C.

A summary table which shows all of the signals from 1 through 11 and the values which can be found on the signals is presented in Table 10. Each step shown in Figure 8 will be explained in the paragraphs below detailing how this GO model is structured and operates.

Signal:	**1**	**2**	**6**	**7**	**9**	**10**	**11**
Values:	0/10	1/10	3/10	4/10	4/10	1/3/4/10	1/3/4/10

Table 10. GO Diagram of Single Model Signals and Values Output.

Task A in the single model is represented by a type 4 operator which is a value generator. If task A is successful a value of zero or one is generated. If task A is not successful or a path is not chosen, a value of ten is generated. Signal 1 will either have a value of zero or ten. Signal 2 with either have a value of one or ten or both signal 1 and 2 will have a value of ten according to the assigned probability of success and failure for task A.

Task B and task C are both represented by a type 1 operator. If task B is selected and task B is successful, it will pass a value of zero to signal 5. There is an operator which follows task B which is a type 15 operator. This operator may change the value which is present at signal 5. If a value of zero is present, the type 15 operator will always change the value from zero to three. If the value is ten, a value of ten is passed to signal 6. The reason this type 15 operator is included is to allow for a trace of values to be established as the model progresses from task A through task C. This model could have been made with just a success or failure value being used, however, to better demonstrate how this model works for traceability, the type 15 operator was added and the value changed.

A type 5 operator is used in this model as the methodology to represent the beginning of the path in which task B is not successful and the person has returned to task A and then proceeded directly from task A to task C. The path task A to task C already exists, but the event of task B failing is a feedback that needs to be modeled separately from the first order choice of selecting task A and then task C. The type 5 operator generates a value of zero if it is successful according to the assigned probability of success and a value of ten otherwise. A type 9 operator incorporates the logic necessary for the system to recognize if task B has failed and if the choice has been made to perform task A and then go to task C. The type 9 will apply the assigned logic and according to the logic will either pass the value of four, indicating that task B had failed and the choice was made to return to task A and then attempt task C or the value of ten indicating the signal path 9 is failed. Either a value of four or a value of ten will be passed by the type 9 operator to signal 9.

Task A to task B follow the signal path of 1, 5 then 6. Task A to task C is path 2. The feedback loop of ‘B fails, go back to perform A and then C’ is signal path 7 then 9. Signals 2, 6 and 9 are all connected to an OR gate with the output of the OR gate connected to signal path 10. The OR gate reads the signals present and passes the lowest value signal on to the task C operator, which is the final operator in this model and is signal 11. The type 1 operator representing task C will succeed or fail according to the probabilities associated with the task. When this system is processed by the GO program, it generates a table of results. These results are the probability of event success and failure and are presented below in Table 11.

Value:	1	2	5	6	7	9	10	11
0	0.80000007	0.00000000	0.64000005	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000
1	0.00000000	0.10000001	0.00000000	0.00000000	0.00000000	0.00000000	0.10000001	0.08000001
3	0.00000000	0.00000000	0.00000000	0.64000005	0.00000000	0.00000000	0.64000005	0.51200002
4	0.00000000	0.00000000	0.00000000	0.00000000	0.80000007	0.28800002	0.20800000	0.16640002
10	0.20000002	0.90000004	0.36000001	0.36000001	0.20000002	0.71200002	0.05200000	0.24160000

Table 11. GO Probability of Events for Single Model.

This study is not meant to be a tutorial on how to use the GO program, but at the same time it is often beneficial to see the code which was used to produce the results as shown in Table 12. The GO program uses several input files which are read in individually and processed serially. There are two input files with the bulk of information required to be read in and processed by GO. The first file is named GO1 and the second is GO2. The code for GO 1 and GO2 are included in Table 12 below.

```
GO1 DATA FOR FEEDBACK LOOP
4 1 2 1 2 $ FIRST LVL A
1 5 1 5 $ SECOND LVL B
15 8 5 6 $ VALUE CHANGER
5 2 7 $ SECOND LVL B
9 6 6 7 9 $ TYPE 9 MODEL1
2 0 3 2 6 9 10 $ OR GATE
1 5 10 11 $ THIRD LVL C
0 1 2 5 6 7 9 10 11$ FINAL SIGNALS

GO2 DATA FOR FEEDBACK LOOP
   1   4  2 3  0 10 0.8
              10  1 0.1
              10 10 0.1 $ FIRST LVL A
   8   15 3 10 0 0 0 1 $ VALUE CHANGER
   2   5  2 4 0.8 10 0.2 $ B FAIL INPUT
   6   9  1 -6 -6 $ TYPE 9 MODEL1
   5   1  0.8 0.2  $ SECOND LVL B/C
```

Table 12. GO1 and GO2 Code Input Files for Single Model.

The probabilities of success and failure for each of the operator types are contained within the GO2 code. Of interest in this single model is that task A has a probability of success to go directly to task B set at 80%, which fits the description the preferred path to take is A, then B then C. The path to go from task A to task C is set at 10% and there is a failure probability of task A set at 10%. Both task B and task C are represented by the type 1 operator and they both have a probability of success equal to 80%. The type 15 operator will always have an output value of either three or ten and this is dependent strictly on the input to the type 15 operator. The type 5 operator has a probability of success equal to 80%. The type 2 operator, which is the OR gate, will always operate 100 percent of the time. There is no probability of success associated with this operator type. The last operator to discuss is the type 9 operator and the logic associated with it. In the GO2 subprogram, the logic required by the GO program to use the type 9 operator is identified by the number 1 and the pair of numbers, -6 and -6. Without going into the specific algorithm, the engineer who programs the code performs a set of additions and subtractions to determine under which conditions and which outputs will be allowed to be passed to signal 9. Once the algorithm is mastered, the engineer will find the type 9 operator very useful.

The last item to discuss in this section is the interpretation of the results. Table 11 contains the results which will be examined are the probabilities of events found on signal 11. With a working knowledge of the single feedback system, one can predict the values. For value zero, there is a 0% probability of this event at signal 11. This is easily confirmed knowing that value zero only appears on signal 1 and does not propagate further. Value one happens 8% of the time and makes sense as it is a first level choice between taking the preferred path the alternate path directly to task C. It should be expected that value three would occur most often as it represents the preferred path. Value four is 16.64% which represents the probability of task B failing and then returning back to task A and then attempting task C. This system as designed and with the probabilities of success and failure associated with it have an overall failure rate as expressed in value ten of 24.16%.

To independently support the results are as calculated by the GO program, a Monte Carlo simulation was set up with Microsoft Excel ™. Random numbers were generated and propagated into a table. The same logic derived from the functional diagram was applied to the Monte Carlo Simulation. A step by step explanation of the GO code and the relationship of GO code to each of the Excel cells code used in the simulation is explained in Appendix A. The Monte Carlo simulation can be found in Appendix B. Note the Monte Carlo simulation and results included in Appendix B have been truncated to only include the first five rows of randomly generated inputs. For the interested reader, increasing the elements to a higher number for increased fidelity of the random results can be accomplished by creating a one-variable data table with the number of experiments desired. Instructions on creating this one-variable data table may be found by using the Excel help function or index tool and following the instructions for creating one-variable data tables. Note that in this particular table set up, the table is row-oriented.

D. EVENT SEQUENCE ANALYSIS (GO PROGRAM) STANDARD FEEDBACK LOOP IMPLEMENTATION

Building on the foundation of a simple feedback loop, a standard feedback loop includes two opportunities for feedbacks to be looped into the analysis of a system. The functional block diagram and appropriate operators of this representative model is presented in the GO diagram in Figure 9 below.

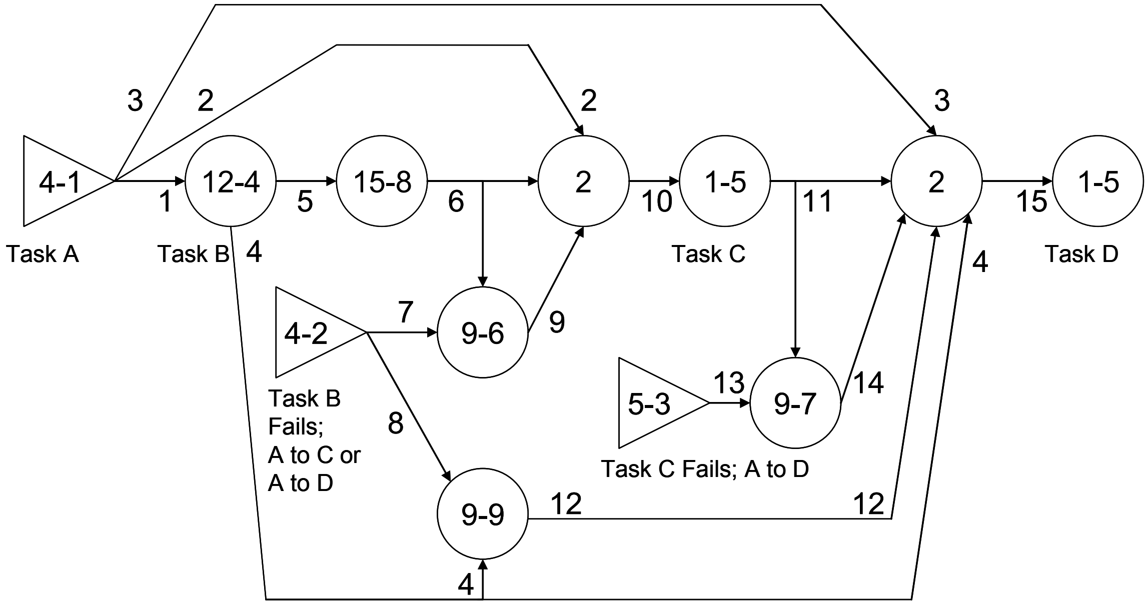


Figure 9. GO Diagram of the Standard Model with Tasks A, B, C and D.

Less detail will be presented in the operation of this model as the behavior of the model is largely similar to that of the simple model. The key differences between this model and that of the simple model is this model adds an additional task, task D and also has a second feedback to consider, namely what options become additionally available when a task is added. With these changes, additional operators needed to be added as well as operator types changed to allow for a representative model to be created. Task A has three paths, the additional path to take is one from task A directly to task D. A choice can be made after accomplishing task B to go directly to task D. If task B fails, there are now two options, either accomplish task A and go to task C or accomplishment of task A and go to task D. Last, there is now another option should task C fail. If task C should fail, there is now an option to return to task A and the go directly to task D.

Although each change made in the diagram could be discussed, the selection of which operators to use in the creation of a model is largely a choice made by the engineer. The important information to highlight is to identify what operator types are required to correctly model the feedback loop implementation. Of note, the type 5 operator is added to this model prior to the single input needed to model the failure of task C and the choice to perform task A and then go to task D. The change of the operator type for task B failing from a type 5 to a type 4 is made to satisfy the need to have multiple task choices available after task B. The additions and uses of the type 9 operators in this model are the same as in the simple model. The last operators added are the OR gates used to receive all possible paths which can be taken prior to task D as well as the type 1 operator used to model task D.

As was provided for the single model, the standard model also had a Monte Carlo simulation set up with Microsoft Excel ™. Random numbers were generated and propagated into a table. The same logic which is applied to the functional diagram was applied to the Monte Carlo Simulation. This simulation can be found in Appendix B. The Monte Carlo simulation for the various signals and the GO results match very well for this model also.

E. EVENT SEQUENCE ANALYSIS (GO PROGRAM) COMPLEX FEEDBACK LOOP IMPLEMENTATION

Continuing with the manner of presentation from section B, a complex feedback loop model is introduced. The functional block diagram and appropriate operators of this representative model is presented in the GO diagram in Figure 10 below.

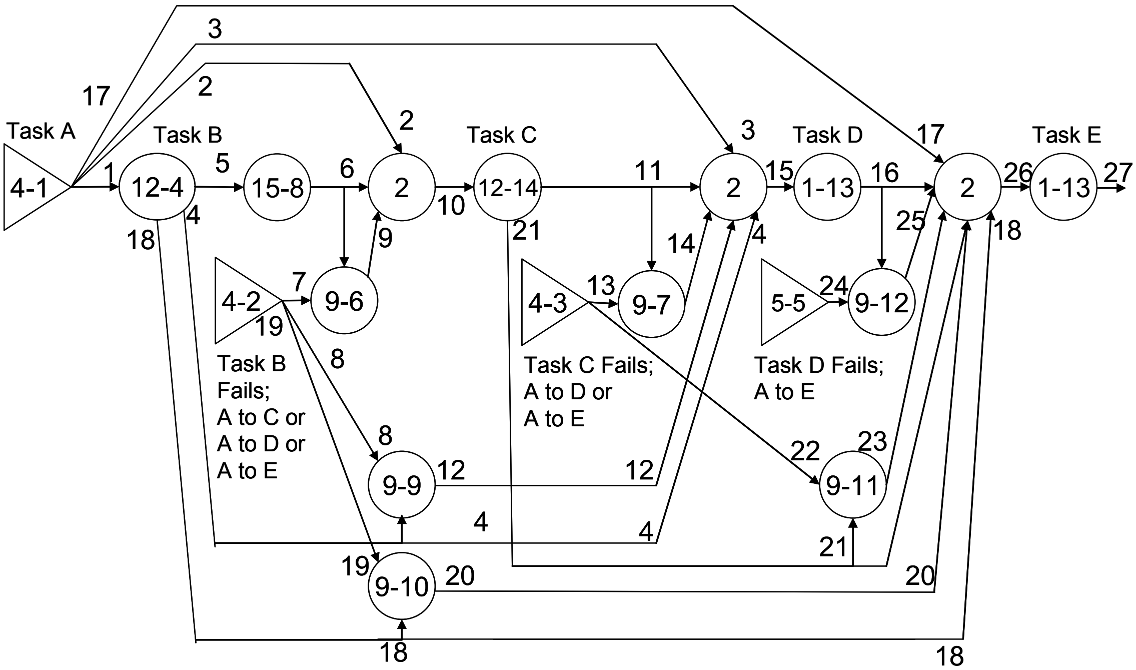


Figure 10. GO Diagram of the Standard Model with Tasks A, B, C, D and E.

The key differences between this model and that of the standard model is this model adds an additional task, task E and also has a third feedback to consider, naturally, the options which become additionally available when this newest task is added. With these changes, once again, additional operators are needed as well as the requisite operator types changed to allow for a representative model to be created. Task A has a total of four paths, this additional path to take is one from task A directly to task E. A choice can be made after accomplishing task B to go directly to task D or E. If task B fails, there are now three options, either accomplish task A and go to task C or D and now task E. Last, there is now another option should task D fail. If task D should fail, there is now an option to return to task A and the go directly to task E. At this point in this demonstration, these changes to system paths or choices are to be expected.

Note the type 5 operator is moved again in this model prior to the single input needed to model the failure of task D and the choice to perform task A and then go to task E. An additional choice is made for the type 4 operators for tasks B failing and task C failing is made to satisfy the need to have the additional task choice available after task B and C. Once again, the additions and uses of the type 9 operators in this model are the same as how the type 9 operators used in the standard and simple models. As expected, an OR gate is used to receive all possible paths which can be taken prior to task E as well as the type 1 operator used to model task E.

A Monte Carlo simulation was created with Microsoft Excel TM with random numbers generated and propagated into a table. The Monte Carlo simulation results for the various signals match the GO results. Both the codes for the Monte Carlo and the GO program are available for review in Appendix B.

F. CHAPTER SUMMARY

The feedback loop can be a complicated element to integrate into a GO block diagram. The approach for simplifying the integration is to analyze how many feedbacks are required and to select the appropriate operator types to fit the system. When advancing from a simple feedback loop with one feedback element and one path after the element to the most complex feedback loop described in this study, the steps required to select the operator type become clear. One feedback loop with one path after the feedback element requires a type 5 operator with a type 9 operator. One feedback loop with two or more paths after the feedback element would necessitate selecting a type 4 operator and for each path added and an additional type 9 operator included. Lastly, with two or more feedback loops integrated into a system, the engineer must determine if a type 5 operator is needed, which represents a single path after the feedback element, or if type 4 operators are needed, representing more than one path after the feedback element. The remaining aspects of integrating feedback loops involve adding the additional signals required so that all additional paths available in the circuit are represented in the GO block diagram.

THIS PAGE INTENTIONALLY LEFT BLANK

V. APPLICATION OF STUDY

A. INTRODUCTION

With the ability of the GO program to perform analyses for both reliability engineering and system safety, there are benefits and efficiencies that may be leveraged within systems engineering. System safety engineers have overlapping interests with engineering disciplines across those within systems engineering. Obviously there is a link between the disciplines of system safety and reliability. In addition, disciplines of design architecture, software engineering, mission assurance and specialty engineering to name a few would receive benefit from modeling elements of their disciplines with the GO program.

To best illustrate the benefits of using a common tool, such as the GO program, for multi-disciplinary work within systems engineering, a survey designed by Mr. Tyrone Jackson was sent out to mission assurance engineering experts in the field of defense systems engineering, manufacturing and development. The survey was designed to collect the opinions of the mission assurance experts regarding mission assurance deficiency analysis methodologies. The results of the survey were then correlated to determine areas of commonality.

There are three tables presented in this study which reference information presented as part of the survey. The first table is a list of deficiency analysis methodologies that are presented in Table 13 below. This list is included in the survey and it represents the set of deficiency analysis methodologies currently being applied by mission assurance experts. The methodology list also includes an opportunity for a tool or methodology not identified within the survey to be manually written in and identified by the survey taker, which is listed in the survey as item W. The second table contains lists of products grouped according to unit-values, from Category 1 through 5, that are presented in Table 14. These lists are included in the survey as Figure 3, and they represent products which are grouped according to the relative impact of their worst-case failure, in terms of potential human, environmental, and financial losses, in that order.

For example, the products in the Category 5 list represent products which have the highest unit-values, and the products listed in the Category 1 column represent systems which have the lowest unit-values. The third and final table contains the survey questions that are presented in Table 15. The survey questions relate the preferred mission assurance approaches among mission assurance experts who took the survey, with respect to their selection of particular deficiency analysis methodologies for particular unit-value categories of products. The results of the survey correlate the commonality of preferred approaches among mission assurance experts, as demonstrated by each selecting nearly all of the deficiency analysis methodologies for Category 5 unit-value products.

A. Functional Diagram Modeling
B. System Reliability Modeling
C. Component Reliability Predictions
D. Product Failure Mode, Effects and Criticality Analysis
E. Sneak Circuit Analysis
F. Design Concern/Rule Analysis
G. Parts Stress Derating Analysis
H. Worst Case Analysis
I. Human Error Predictions
J. Environmental Event Survivability Predictions
K. Anomaly Detection and Response/Failure Coverage Analysis
L. Maintainability Predictions
M. Operational Dependability/Availability Modeling
N. Hazard/Safety Analysis
O. Software Component Reliability Predictions
P. Process Failure Mode, Effects and Criticality Analysis
Q. Event Tree Analysis
R. Fault Tree Analysis
S. Fishbone Analysis
T. Similarity/Allocations Analysis
U. Parts Engineering Analysis
V. Stress and Damage Simulation Analysis
W. Other Deficiency Analysis Methodology:_________________________

Table 13.   List of Deficiency Analysis Methodologies Used in Survey.

Unit-Value Category 1 • Motorized/ manual hand tools • Fire arms • Explosive devices • Consumer electronics • Personal computers • Household appliances • Battery operated toys • Infant/ children toys	Unit-Value Category 2 • Automobiles/ trucks/ motorcycles • Recreational vehicles/ motor homes • Industrial electronics • Experimental/ kit aircraft • Computer servers • Farm equipment • Medical/ laboratory equipment • Factory machinery • Test equipment/ software • Mobil construction/ demolition/ mining equipment • Mobil communications equipment
Unit-Value Category 3 • Experimental satellites • Small private aircraft/helicopters • Commercial buses • Oil tankers/ rigs • Freighters • Ground-mobile/ ground-fixed military electronics • Ground-mobile/ ground-fixed conventional military weapons • Freight trains • Unmanned terrestrial exploration vehicles • Amusement park rides • Elevators/ escalators	Unit-Value Category 4 • Communications satellites • Fossil fuel/hydro-electric power plants • Water filtration plants • Short-range missiles/rockets • Commercial passenger aircraft/ helicopters • Military aircraft/ helicopters • Military drones/ unmanned vehicles • Ocean liners • Diesel-powered naval vessels • Passenger trains • Airborne military electronics • Extraterrestrial exploration vehicles
Unit-Value Category 5 • Defense satellites • Launch vehicles • Long-range missiles • Nuclear-powered naval vessels • Nuclear weapons • Nuclear power plants • Manned spacecraft • Space stations • Extraterrestrial manned habitats	

Table 14. List of Unit-Value Category Definitions.

1. Which of the deficiency analysis methodologies would you recommend be used to assess deficiencies of a Capability Level 1 product in Figure 3?

Highlight letters which correspond to methodologies applicable to Capability Level 1 products.

[A] [B] [C] [D] [E] [F] [G] [H] [I] [J] [K] [L] [M] [N] [O] [P] [Q] [R] [S] [T] [U] [V] [W]

2. Which of the deficiency analysis methodologies would you recommend be used to assess deficiencies of a Capability Level 2 product in Figure 3?

Highlight letters which correspond to methodologies applicable to Capability Level 2 products.

[A] [B] [C] [D] [E] [F] [G] [H] [I] [J] [K] [L] [M] [N] [O] [P] [Q] [R] [S] [T] [U] [V] [W]

3. Which of the deficiency analysis methodologies would you recommend be used to assess deficiencies of a Capability Level 3 product in Figure 3?

Highlight letters which correspond to methodologies applicable to Capability Level 3 products.

[A] [B] [C] [D] [E] [F] [G] [H] [I] [J] [K] [L] [M] [N] [O] [P] [Q] [R] [S] [T] [U] [V] [W]

4. Which of the deficiency analysis methodologies would you recommend be used to assess deficiencies of a Capability Level 4 product in Figure 3?

Highlight letters which correspond to methodologies applicable to Capability Level 4 products.

[A] [B] [C] [D] [E] [F] [G] [H] [I] [J] [K] [L] [M] [N] [O] [P] [Q] [R] [S] [T] [U] [V] [W]

5. Which of the deficiency analysis methodologies would you recommend be used to assess deficiencies of a Capability Level 5 product in Figure 3?

Highlight letters which correspond to methodologies applicable to Capability Level 5 products.

[A] [B] [C] [D] [E] [F] [G] [H] [I] [J] [K] [L] [M] [N] [O] [P] [Q] [R] [S] [T] [U] [V] [W]

Table 15. List of Deficiency Analysis Methodologies Survey Questions.

The results of the survey revealed there are extensive overlapping engineering design areas, requirements and analyses. Figures 11, 12 and 13 are examples of these overlapping areas with the figures taken directly from the results generated by the author of the survey. The author states, “Without detailed planning, this overlap may result in instances of duplication in effort or process escapement among different disciplines.” The results of this survey directly support the development and use of common use tools such as the GO program for inter-disciplinary application and use of analyses methodologies.

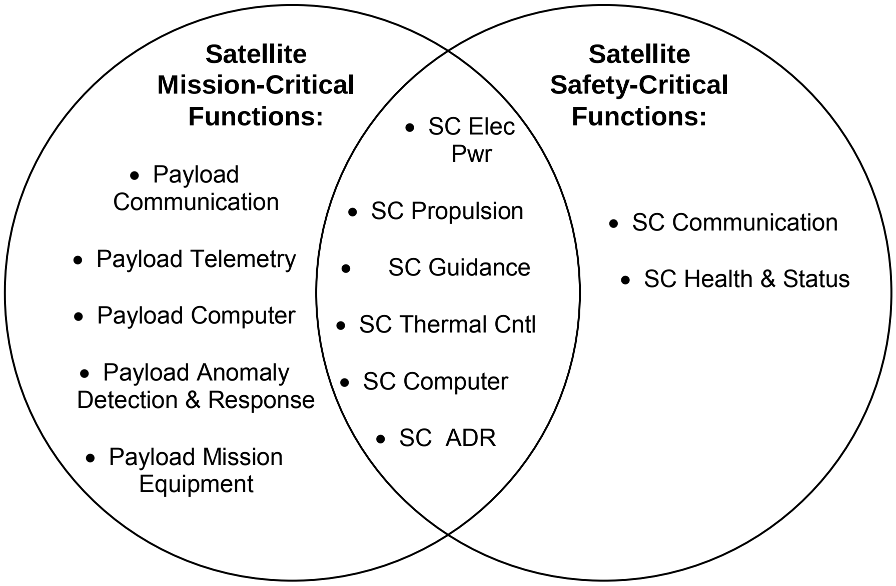


Figure 11. Example Overlap of Safety-Critical and Mission-Critical Design Attributes.

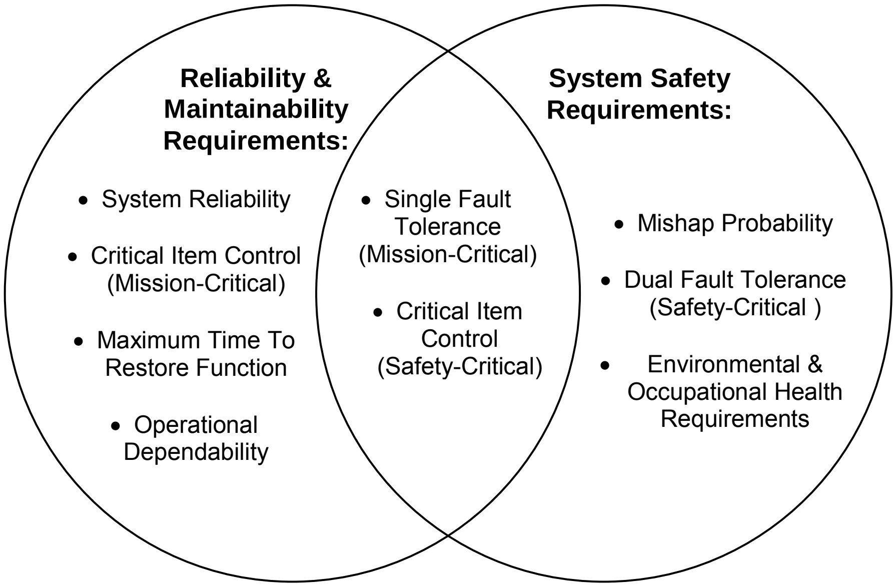

Reliability & Maintainability Requirements:
• System Reliability
• Critical Item Control (Mission-Critical)
• Maximum Time To Restore Function
• Operational Dependability
• Single Fault Tolerance (Mission-Critical)
• Critical Item Control (Safety-Critical)
System Safety Requirements:
• Mishap Probability
• Dual Fault Tolerance (Safety-Critical)
• Environmental & Occupational Health Requirements

Figure 12. Example Overlap of System Safety and Reliability / Maintainability Requirements.

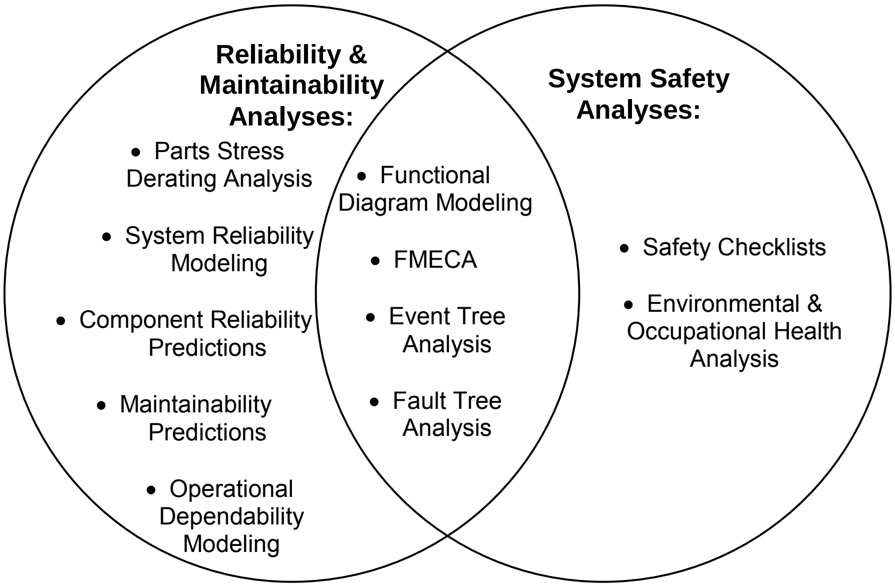

Reliability & Maintainability Analyses:
• Parts Stress Derating Analysis
• System Reliability Modeling
• Component Reliability Predictions
• Maintainability Predictions
• Operational Dependability Modeling
• Functional Diagram Modeling
• FMECA
• Event Tree Analysis
• Fault Tree Analysis
System Safety Analyses:
• Safety Checklists
• Environmental & Occupational Health Analysis

Figure 13. Example Overlap of System Safety and Reliability / Maintainability Analyses.

B. RECOMMENDATIONS

A better understanding of the benefits of using a systematic, structured, updateable and repeatable method of analyses provides a more complete assessment of the performance of a system as a whole as well as offer opportunities for identifying previously unknown risks. Beginning with architectural considerations and continuing through the final integration of system elements, there are benefits to having a common tool to be used across all disciplines for design selections and risk, reliability and safety analyses.

Eliminating or reducing the amount of rework from similar disciplines with differing requirements is an immediate application which could lead to reducing cost and schedule for complex and often cost intensive systems. Taking advantage of the inventory of computer aided tools currently available and finding and exploiting the synergies of easy to use tools will also immediately reduce the logistical footprint of multiple analyses techniques. With the ability of the GO program to perform analyses for both reliability engineering and system safety, these benefits and efficiencies may be leveraged within systems engineering efforts.

VI. CONCLUSIONS

A. KEY POINTS AND RECOMMENDATIONS

With the demonstrated ability of the GO program to correctly model feedback loops, it clears a path for the Department of Defense to investigate the benefits of adopting a standardized approach for the analyses of complex systems. The use of computerized tools aids greatly in assessing complex systems and the ability of the systems to meet requirements. In the introduction chapter of this study, ten points were made regarding the benefits of using the GO program as a common analyses tool across multiple disciplines. The list of benefits is repeated below.

1. There are no commonly known commercial reliability modeling tools available that are driven by a functional model. Performing reliability modeling using mathematical models introduces errors when function-level probabilities of occurrence are involved.

2. There are no commonly known commercial FMECA (Failure Modes and Effects Criticality Analysis) tools available that are driven by a functional model. Performing FMECA manually using worksheets is likely to introduce errors when complex systems are involved.

3. There are no commonly known commercial event tree analysis tools available that are driven by a functional model. Performing event tree analysis manually using worksheets introduces errors when latent operating modes are involved.

4. There are no commonly known commercial fault tree analysis tools available that are driven by a functional model. Performing fault tree analysis manually using logic symbols introduce errors when complex systems are involved.

5. There are no commonly known commercial engineering tools available that integrate reliability modeling, FMECA, event sequence analysis, and fault tree analysis. Performing these analyses separately using different methods is labor-intensive and introduces errors when trying to integrate the results.

6. There is no commonly known commercial suite of integrated safety/reliability tools available. One of the benefits in a future study is the research involved in defining an "enhanced GO" tool which allows integrating safety and reliability analyses via output which includes function-level failure cut-sets and function-level probabilities of success.

7. There are no commonly known commercial reliability modeling tools available that are driven by a functional model which is compliant with an industry approved standard. The AIAA S-102 Mission Assurance Standards Working Group is interested in coordinating the further development of an "enhanced GO" to create an Open Source tool which is driven by a functional model that is compliant with the S-102 Functional Diagram Modeling standard.

8. There are no commonly known commercial reliability modeling tools available which allow feedback to be included in the model. One of the topics in this study is the research involved in defining an "enhanced GO" which allows feedback to be included in the reliability model. GO automatically generates the reliability model from the functional model.

9. There are no commonly known commercial reliability modeling tools available which provide output in a format which is compliant with an industry approved standard. The AIAA S-102 Mission Assurance Standards Working Group is interested in coordinating the further development of an "enhanced GO" to create an Open Source tool which provides output which is compliant with the XML data element descriptions defined in the S-102 standards.

10. In the past 25 years, very few commercial reliability analysis tools were developed that are driven by functional models and which can perform multiple analyses. One of these tools is called MultiLinx, which was used in the NASA Crew Return Vehicle (CRV) program between 1998 and 2003. This tool was not marketed in the public domain after the CRV program ended. The other tool is called eXpress, and it is currently used by many companies to analyze the testability of a design. The eXpress tool is not intended to be used in safety design or reliability design assessments. An "enhanced GO" could become the affordable Open Source alternative which supports both safety design and reliability design assessments.

Each of the above numbered items may be integrated into a government approved approach for use throughout the entire lifecycle of a system. There are opportunities to improve the way the government and the contractor use the best practices available for acquiring systems of ever increasing cost and complexity. Common tools with universally available, documented, and understood methodologies for their use will improve the government’s oversight of defense acquisition systems and increase the contractor’s awareness of the impacts of design decisions and changes throughout the life cycle systems engineering process.

B. AREAS TO CONDUCT FURTHER RESEARCH

The GO program is currently designed to read in data files, perform analyses and write data files presented in a table format for review. Each of the files read are created by the engineers who have a high familiarity with the system being modeled. This technology has an opportunity to be updated with the power and complexity of the computers of today and the improved ease of use with graphical user interfaces. One initial area in which the ease of the use of this program can be greatly improved would be for a computer or software engineer to design a graphical user interface. Another benefit which may be gained is in possibly having the GO code updated to operate in code which is more advanced than FORTRAN, potentially allowing for greater application of the code or for obtaining intermediate levels of results of analyses. The interested reader will find all of the input files used to create the single, standard and complex models in Appendix C.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX A. MONTE CARLO SIMULATION AND GO CODES

Signal 1	GO Code	1 4 2 3 0 10 0.8 10 1 0.1 10 10 0.1 $ FIRST LVL A
	Explanation	Value of 0 with probability of 0.8 is a given property for the probability of this value on signal 1
		Value of 10 with a combined probability of 0.2 is a given property for the probability of this value on signal 1
	Excel Code	=IF(rand_num<sig_1_percent,0,10)
	Explanation	If the random number is less than the assigned signal 1 percent (0.80), the value is 0 else the value is 10.
	Math:	N/A based on random number generator
	Note:	There is an value of 10 on signal 1 and 10 on signal 2 10% of the time. This is done to allow for a condition for the value 4, which originates at signal 7 to be output at the OR gate The logic of the OR gate is that the signal with the lowest value will be passed at the gate

Table 16. GO1 and Excel Code Signal 1 for Single Model.

Signal 2	GO Code	1 4 2 3 0 10 0.8 10 1 0.1 10 10 0.1 $ FIRST LVL A
	Explanation	Value of 10 with probability of 0.1 is a given property for the probability of this value on signal 2.
		Value of 0 with a combined probability of 0.9 is a given property for the probability of this value on signal 2
	Excel Code	=IF(AND(rand_num>=0.8,rand_num<0.9),1,10)
	Explanation	If the random number is less than the assigned signal 2 percent (net 0.10), the value is 0 else the value is 10.
	Math:	N/A based on random number generator
	Note:	No notes for this signal

Table 17. GO1 and Excel Code Signal 2 for Single Model.

<table>
<tr><td rowspan="11">Signal 5</td><td>GO Code</td><td colspan="4">5 1 0.8 0.2 $ SECOND LVL C</td></tr>
<tr><td>Explanation</td><td colspan="4">The type 1 operator passes signals based on a predetermined success/failure rate</td></tr>
<tr><td>Excel Code</td><td colspan="4">=IF(rand_num5<0.8,signal_1,10)</td></tr>
<tr><td>Explanation</td><td colspan="4">If the random number is less than the assigned signal 5 percent (0.80), the value on signal 1 is passed else the output value is 10. Based on the random number generator, any random numbers less than signal 1's assigned percentage (0.80) will be passed.</td></tr>
<tr><td rowspan="6">Math:</td><td colspan="4">The multiplication is 0.80 times the probabilities of the values found on signal 10</td></tr>
<tr><td><b>Value</b></td><td><b>Signal 1</b></td><td><b>Multiply</b></td><td><b>Signal 5</b></td></tr>
<tr><td>0</td><td>0.80000007</td><td>Multiplied by 0.80</td><td>0.64000006</td></tr>
<tr><td>1</td><td>0.00000000</td><td>Multiplied by 0.80</td><td>0.00000000</td></tr>
<tr><td>3</td><td>0.00000000</td><td>Multiplied by 0.80</td><td>0.00000000</td></tr>
<tr><td>4</td><td>0.00000000</td><td>Multiplied by 0.80</td><td>0.00000000</td></tr>
<tr><td></td><td>10</td><td>0.20000002</td><td></td><td>0.36000001</td></tr>
</table>

<table>
<tr><td>Signal 5</td><td>Note:</td><td>The failure value is represented by the addition of 0.80 * 0.2 = 0.16 plus the 0.20 failure rate of the type 1 operator, totaling 0.36 An alternative way to get this number is to add all of the values for the signals 0-4 and subtract from 1 which will also yield the overall failure rate.</td></tr>
</table>

Table 18. GO1 and Excel Code Signal 5 for Single Model.

Signal 6	GO Code	8 15 3 10 0 0 0 1 $ VALUE CHANGER
	Explanation	Value of 3 with probability of 0.8 is a given property due to the inputs going to the type 15 operator.
		Value of 10 with probability of 0.2 is a given property due to the inputs going to the type 15 operator.
	Excel Code	=IF(signal_5=0,3,10)
	Explanation	If Signal 5 =0, the output is three else the output is 10.
	Math:	N/A based on inputs received which are determined by random numbers generated at Signal 1 and Signal 5
	Note:	Note: 15-8 has no probability associated with it. It simply has an output based on the input received. The code for this type operator is for any input of 0, the output is 3, otherwise the output is 10.

Table 19. GO1 and Excel Code Signal 6 for Single Model.

Signal 7	GO Code	2 5 2 4 0.8 10 0.2 $ B FAIL INPUT
	Explanation	Value of 4 with probability of 0.8 is a given property due to the inputs going to the type 5 operator
		Value of 10 with probability of 0.2 is a given property due to the inputs going to the type 5 operator
	Excel Code	=IF(rand_num2<sig_7_percent,4,10)
	Explanation	Similar to the type four operator on signals 1 and 2, if the random number (this time random number 2) is less than the assigned signal 7 percent (0.80), the value is 4 else the value is 10.
	Math:	N/A based on random number generator
	Note:	No notes for this signal

Table 20.   GO1 and Excel Code Signal 7 for Single Model.

Signal 9	GO Code	6 9 1 -6 -6 $ TYPE 9 MODEL1
	Explanation	Value of 4 is output when the input on signal 6 is 10 and the input on signal 7 is 4, otherwise the output is 10
	Excel Code	=IF(AND(signal_7=4,signal_6=10),4,10)
	Explanation	Excel populates this data based on the values available from the previous signals generated from the random numbers This will not produce an exact probability value, but on average, due to the nature of probability the average result will approximately equal the probability value.
	Math:	Based on logic assigned to type 9 operator
	Note:	No notes for this signal

Table 21.   GO1 and Excel Code Signal 9 for Single Model.

Signal 10	GO Code	No probability data for a type 2 operator which is an or gate.
	Explanation	The or gate will take the lowest value from the signals present
		This or gate receives signals from 2, 6 and 9 to pass on as signal 10.
	Excel Code	=MIN(signal_2,signal_6,signal_9)
	Explanation	Excel function MIN does the same operation as described for the type 2 operator for the GO program
	Math:	Signal 2 probability value is directly associated with percentage value assigned on line 2 of the code, with line 2 being 0.10 for value 1, the or gate signal is 0.1
		Signal 6 probability value is directly associated with percentage value assigned on line 1 of the code, with line 1 being 0.80 for value 0, the or gate signal is 0.80
		Signal 9 probability value is directly associated with percentage value assigned on line 3 of the code, with line 3 being 0.10 for value 10 to be present at the same time on signals 1 and 2, multiplied by the percentage assigned to the signal 7 operator, which is 0.80 The probability for signal 9 to be passed at the or gate is 0.80 * 0.10 = 0.08
		The last value passed at the or gate is 10 which only happens in this case when signals 1 and 2 are 10 and when signal 7 results in a 10 (in this case 20 percent)
		0.20 times 0.10 (signal 7 failure times signal 1 and 2 failure) = 0.02
	Note:	No notes for this signal

Table 22. GO1 and Excel Code Signal 10 for Single Model.

<table>
<tr><td rowspan="12">Signal
11</td><td>GO Code</td><td colspan="4">5 1 0.8 0.2 $ SECOND LVL C</td></tr>
<tr><td>Explanation</td><td colspan="4">The type 1 operator passes signals based on a predetermined success/failure rate</td></tr>
<tr><td>Excel Code</td><td colspan="4">=IF(rand_num3<0.8,MIN(signal_2,signal_6,signal_9),10)</td></tr>
<tr><td>Explanation</td><td colspan="4">If the random number is less than the assigned signal 11 percent (0.80), the value on signal 10 is passed else the output value is 10. Based on the random number generator, any random numbers less than signal 11's assigned percentage (0.80) will be passed.</td></tr>
<tr><td rowspan="7">Math:</td><td colspan="4">The multiplication is 0.80 * the probabilities of the values found on signal 10</td></tr>
<tr><td><b>Value</b></td><td><b>Signal 10</b></td><td><b>Multiply</b></td><td><b>Signal 11</b></td></tr>
<tr><td>0</td><td>0.00000000</td><td>Multiplied by 0.80</td><td>0.00000000</td></tr>
<tr><td>1</td><td>0.10000001</td><td>Multiplied by 0.80</td><td>0.08000001</td></tr>
<tr><td>3</td><td>0.64000005</td><td>Multiplied by 0.80</td><td>0.51200002</td></tr>
<tr><td>4</td><td>0.20800000</td><td>Multiplied by 0.80</td><td>0.16640002</td></tr>
<tr><td>10</td><td>0.05200000</td><td></td><td>0.24160000</td></tr>
<tr><td>Note:</td><td colspan="4">The failure value is represented by the addition of 0.80 * 0.052 = 0.0416 plus the 0.20 failure rate of the type 1 operator, totaling 0.2416 An alternative way to get this number is to add all of the values for the signals 0-4 and subtract from 1 which will also yield the overall failure rate.</td></tr>
</table>

Table 23. GO1 and Excel Code Signal 11 for Single Model.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX B. MONTE CARLO SIMULATIONS

RV1	RV2	RV3	RV4		Signal 1
0.72	0.56	0.71	0.08	Description	A to B signal 1, state 0/10
				Likelihood	80%
	Random Variable1	Random Variable2	Random Variable3	Random Variable4	RV 1
					=IF(rand_num<sig_1_percent,0,10)
Exp #	0.72	0.56	0.71	0.08	0
1	0.05	0.54	0.99	0.23	0
2	0.52	0.60	0.99	0.02	0
3	0.29	0.07	0.03	0.52	0
4	0.20	0.40	0.30	0.83	0
5	0.74	0.39	0.92	0.88	0

Table 24. Single Model Simulation Description for Signal 1.

Signal 2	Signal 5	Signal 6
A to C signal 2, state 1/10	B to C signal 2, state 1/10	B to C signal 6, state 3/10
10%	80%	N/A
RV 1	RV 2	N/A
=IF(AND(rand_num>=0.8,rand_num<0.9), 1,10)	=IF(rand_num2<0.8,signal_1, 10)	=IF(signal_5=0,3, 10)
10	0	3
10	0	3
10	0	3
10	0	3
10	0	3
10	0	3

Table 25. Single Model Simulation Description for Signals 2, 5 and 6.

Signal 7	Signal 9
Task B fails; A to C signal 7, state 4/10	Task B Fails; A to C signal 9, state 4/10
80%	N/A
RV 3	N/A
=IF(rand_num3<sig_7_percent,4,10)	=IF(AND(signal_7=4,signal_6=10),4,10)
4	10
10	10
10	10
4	10
4	10
10	10

Table 26. Single Model Simulation Description for Signals 7 and 9.

Signal 10	Signal 11
Or Gate to Task C signal 10 state 1/3/4/10	Task C signal 11, state 1/3/4/10
N/A	N/A
N/A	RV 4
=MIN(signal_2,signal_6,signal_9)	=IF(rand_num4<0.8,MIN(signal_2,signal_6,signal_9),10)
3	3
3	3
3	3
3	3
3	10
3	10

Table 27. Single Model Simulation Description for Signals 10 and 11.

RV1	RV2	RV3	RV4	RV5		Signal 1
0.29	0.27	0.76	0.04	0.99	Description	A to B signal 1, state 0/10
					Likelihood	80%
	Random Variable 1	Random Variable 2	Random Variable 3	Random Variable 4	Random Variable5	RV 1
						=IF(rand_num<sig_1_percent,0,10)
Exp #	0.29	0.27	0.76	0.04	0.99	0
1	0.17	0.60	0.86	0.05	0.02	0
2	0.84	0.64	0.35	0.32	0.28	10
3	0.64	0.74	0.11	0.96	0.22	0
4	0.96	0.62	0.07	0.23	0.99	10
5	0.38	0.12	0.30	0.90	0.91	0

Table 28. Standard Model Simulation Description for Signal 1.

Signal 2	Signal 3	Signal 4	Signal 5
A to C signal 2, state 1/10	A to D signal 3, state 2/10	B to D signal 4, state 0/10	B to C signal 5, state 0/10
10%	10%	60%	10%
RV 1	RV 1	RV 1	RV 1
=IF(AND(rand_num>=0.8,rand_num<0.9),1,10)	=IF(rand_num>0.9,2,10)	=IF(AND(rand_num<(sig_4_percent*sig_1_percent)),0,10)	=IF(AND(rand_num<((sig_4_percent*sig_1_percent)+(sig_5_percent*sig_1_percent)),rand_num>=(sig_4_percent*sig_1_percent)),0,10)
10	10	0	10
10	10	0	10
1	10	10	10
10	10	10	10
10	2	10	10
10	10	0	10

Table 29. Standard Model Simulation Description for Signals 2, 3, 4 and 5.

Signal 6	Signal 7	Signal 8	Signal 9
B to C signal 6, state 3/10	Task B fails; A to C signal 7, state 4/10	Task B fails; A to D signal 8, state 5/10	Task B Fails; A to C signal 9, state 4/10
N/A	80%	20%	N/A
N/A	RV 2	RV 2	RV 1
=IF(signal_5=0,3,10)	=IF(rand_num2<sig_7_percent,4,10)	=IF(rand_num2>=sig_7_percent,5,10)	=IF(AND(signal_6=10,signal_7=4),4,10)
10	4	10	4
10	4	10	4
10	4	10	4
10	4	10	4
10	4	10	4
10	4	10	4

Table 30. Standard Model Simulation Description for Signals 6, 7, 8 and 9.

Signal 10	Signal 11	Signal 12	Signal 13
Or Gate to Task C signal 10 state 1/3/4/10	Task C signal 11, state 1/3/4/10	Task B fails; A to D signal 12, state 5/10	Task C fails; A to D signal 13, state 6/10
N/A	80%	N/A	N/A
N/A	RV 3	N/A	RV 4
=MIN(signal_2,signal_6,signal_9)	=IF(AND(rand_num3<Q12),MIN(signal_2,signal_6,signal_9),10)	=IF(AND(signal_8=5,signal_4=10),5,10)	=IF(rand_num4<0.8,6,10)
4	4	10	6
4	10	10	6
1	1	10	6
4	4	10	10
4	4	10	6
4	4	10	10

Table 31. Standard Model Simulation Description for Signals 10, 11, 12 and 13.

Signal 14	Signal 15	Signal 16
Task C fails; A to D signal 14, state 6/10	Or Gate to Task D signal 15, state 0/1/3/4/5/6/10	Task D signal 16, state 0/1/2/3/4/5/6/10
N/A	N/A	80%
N/A	N/A	RV 5
=IF(AND(signal_13=6,signal_11=10),6,10)	=MIN(signal_3,signal_4,signal_11,signal_12,signal_14)	=IF(rand_num5<0.8,MIN(signal_3,signal_4,signal_11,signal_12,signal_14),10)
10	0	10
6	0	0
10	1	1
10	4	4
10	2	10
10	0	10

Table 32. Standard Model Simulation Description for Signals 14, 15 and 16.

RV1	RV2	RV3	RV4	RV5	RV6	RV7	
0.72	0.21	0.99	0.43	0.00	0.10	0.91	Description
							Likelihood
	Random Variable 1	Random Variable 2	Random Variable 3	Random Variable 4	Random Variable 5	Random Variable 6	Random Variable7
Exp #	0.33	0.52	0.36	0.99	0.69	0.70	0.51
1	0.06	0.58	0.08	0.92	0.48	0.13	0.73
2	0.86	0.05	0.32	0.20	0.36	0.43	0.35
3	0.92	0.01	0.59	0.11	0.46	0.47	0.44
4	0.17	0.81	0.33	0.02	0.37	0.95	0.76
5	0.76	0.54	0.92	0.79	0.99	0.26	0.79

Table 33. Complex Model Simulation Description for Random Variables.

Signal 1	Signal 2	Signal 3	Signal 17
A to B signal 1, state 0/20	A to C signal 2, state 1/20	A to D signal 3, state 2/20	A to D signal 17, state 7/20
70%	10%	10%	10%
RV 1	RV 1	RV 1	RV 1
=IF(rand_num<sig_1_percent,0,20)	=IF(AND(rand_num>=0.7,rand_num<0.8),1,20)	=IF(AND(rand_num>=0.8,rand_num<0.9),2,20)	=IF(rand_num>0.9,7,20)
0	20	20	20
0	20	20	20
20	20	2	20
20	20	20	7
0	20	20	20
20	1	20	20

Table 34. Complex Model Simulation Description for Signals 1, 2, 3 and 17.

Signal 4	Signal 5	Signal 18	Signal 6
B to D signal 4, state 0/20	B to C signal 5, state 0/20	B to E signal 18, state 0/20	B to C signal 6, state 3/20
50%	10%	10%	N/A
RV 1	RV 1	RV 1	N/A
=IF(AND(rand_num<(sig_4_percent*sig_1_percent)),0,20)	=IF(AND(rand_num<(sig_4_percent*sig_1_percent+sig_5_percent*sig_1_percent),rand_num>=((sig_4_percent*sig_1_percent+sig_5_percent*sig_1_percent)-(sig_5_percent*sig_1_percent))),0,20)	=IF(AND(rand_num<(sig_4_percent*sig_1_percent+sig_5_percent*sig_1_percent+sig_18_percent*sig_1_percent),rand_num>=(sig_4_percent*sig_1_percent+sig_5_percent*sig_1_percent)),0,20)	=IF(signal_5=0,3,20)
0	20	20	20
0	20	20	20
20	20	20	20
20	20	20	20
0	20	20	20
20	20	20	20

Table 35. Complex Model Simulation Description for Signals 4, 5, 6 and 18.

Signal 7	Signal 8	Signal 19	Signal 9
Task B fails; A to C signal 7, state 4/20	Task B fails; A to D signal 8, state 5/20	Task B Fails; A to E signal 9, state 8/20	Task B Fails; A to C signal 9, state 4/20
80%	10%	10%	N/A
RV 2	RV 2	RV 2	N/A
=IF(rand_num2<sig_7_percent,4,20)	=IF(AND(rand_num2>=sig_7_percent,rand_num2<(sig_7_percent+sig_8_percent)),5,20)	=IF(AND(rand_num2>=(sig_7_percent+sig_8_percent),rand_num2<(sig_7_percent+sig_8_percent+sig_19_percent)),8,20)	=IF(AND(signal_6=20,signal_7=4),4,20)
4	20	20	4
4	20	20	4
4	20	20	4
4	20	20	4
20	5	20	20
4	20	20	4

Table 36. Complex Model Simulation Description for Signals 7, 8, 9 and 19.

Signal 10	Signal 11	Signal 21	Signal 12
Or Gate to Task C signal 20, state 1/3/4/20	Task C signal 11, state 1/3/4/20	Task C signal 21, state 1/3/4/20	Task B fails; A to D signal 12, state 5/20
N/A	60%	10%	N/A
N/A	RV 3	RV 3	N/A
=MIN(signal_2,signal_6,signal_9)	=IF(AND(rand_num3<V16),MIN(signal_2,signal_6,signal_9),20)	=IF(rand_num3>(1-sig_21_percent),MIN(signal_2,signal_6,signal_9),20)	=IF(AND(signal_8=5,signal_4=20),5,20)
4	4	20	20
4	4	20	20
4	4	20	20
4	4	20	20
20	20	20	20
1	20	1	20

Table 37. Complex Model Simulation Description for Signals 10, 11, 12 and 21.

Signal 20	Signal 13	Signal 14	Signal 22
Task B fails; A to E signal 20, state 9/21	Task C fails; A to D signal 13, state 6/20	Task C fails; A to D signal 14, state 9/20	Task C fails; A to E signal 22, state 9/20
N/A	80%	N/A	20%
N/A	RV 4	N/A	RV 4
=IF(AND(signal_19=8,signal_18=20),8,20)	=IF(rand_num4<0.8,6,20)	=IF(AND(signal_13=6,signal_11=20),6,20)	=IF(rand_num4>sig_13_percent,9,20)
20	20	20	9
20	20	20	9
20	6	20	20
20	6	20	20
20	6	6	20
20	6	6	20

Table 38. Complex Model Simulation Description for Signals 13, 14, 20 and 22.

Signal 23	Signal 15	Signal 16	Signal 24
Task C fails; A to E signal 23, state 9/20	Or Gate to Task D signal 15 state 0/1/2/3/4/5/6/20	Task D signal 16, state 0/1/2/3/4/5/6/20	Task D fails; A to E signal 24, state 10/20
N/A	N/A	80%	80%
N/A	N/A	RV 5	RV 6
=IF(AND(signal_22=9,signal_21=20),9,20)	=MIN(signal_3,signal_4,signal_11,signal_12,signal_14)	=IF(rand_num5<0.8,MIN(signal_3,signal_4,signal_11,signal_12,signal_14),20)	=IF(rand_num6<AF16,10,20)
9	0	0	10
9	0	0	10
20	2	2	10
20	4	4	10
20	0	0	20
20	6	20	10

Table 39. Complex Model Simulation Description for Signals 15, 16, 23 and 24.

Signal 25	Signal 26	Signal 27
Task C fails; A to E signal 25, state 10/20	Or Gate to Task E signal 26 state 0-10/20	Task E signal 27, state 0-10/20
N/A	N/A	80%
N/A	N/A	RV 7
=IF(AND(signal_24=10, signal_16=20),10,20)	=MIN(signal_16,signal_17,signal_18,signal_20,signal_21,signal_23,signal_25)	=IF(rand_num7<AI16,MIN(signal_16,signal_17,signal_18,signal_20,signal_21,signal_23,signal_25),20)
20	0	0
20	0	0
20	2	2
20	4	4
20	0	0
10	1	1

Table 40. Complex Model Simulation Description for Signals 25, 26 and 27.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX C. GO PROGRAM CODE

```
GO1 DATA FOR SINGLE FEEDBACK LOOP
,,,,,10/
4 1 2 1 2 $ FIRST LVL A
1 5 1 5 $ SECOND LVL B
15 8 5 6 $ VALUE CHANGER
5 2 7 $ SECOND LVL B
9 6 6 7 9 $ TYPE 9 MODEL1
2 0 3 2 6 9 10 $ OR GATE
1 5 10 11 $ THIRD LVL C
0 11$ FINAL SIGNALS
EOR
□
```

Table 41. GO1 Data For Single Feedback Loop.

```
GO2 DATA FOR SINGLE FEEDBACK LOOP
0/
   1   4  2 3  0 10 0.8
              10  1 0.1
              10 10 0.1 $ FIRST LVL A
   8   15 3 10 0 0 0 1 $ VALUE CHANGER
   2   5  2 4 0.8 10 0.2 $ B FAIL INPUT
   6   9  1 -6 -6 $ TYPE 9 MODEL1
   5   1  0.8 0.2  $ SECOND LVL B/C
EOR
□
```

Table 42. GO2 Data For Single Feedback Loop.

```
GO3 DATA FOR SINGLE FEEDBACK LOOP
,,,1/
EOR
□
```

Table 43. GO3 Data For Single Feedback Loop.

GO1 DATA FOR STANDARD FEEDBACK LOOP
,,,,,10/
4 1 3 1 2 3 $ FIRST LVL A
12 4 1 2 4 5 $ PATH SPLITTER
15 8 5 6 $ VALUE CHANGER
4 2 2 7 8 $ SECOND LVL B
9 6 6 7 9 $ TYPE 9 MODEL1
9 9 4 8 12 $ TYPE 9 MODEL2
2 0 3 2 6 9 10 $ OR GATE
1 5 10 11 $ THIRD LVL C
5 3 13 $ DECISION C INPUT
9 7 11 13 14 $ TYPE 9 MODEL3
2 0 5 3 4 11 12 14 15 $ OR GATE
1 5 15 16 $ FOURTH LVL D
0 16 $ FINAL SIGNALS
EOR
□

Table 44. GO1 Data For Standard Feedback Loop.

GO2 DATA FOR STANDARD FEEDBACK LOOP
0/
1   4 3 3 0 10 10 0.8
10 1 10 0.1
10 10 2 0.1 $ FIRST LVL A
4   12 2 0.6 0.1 $ PATH SPLITTER
8   15 3 10 0 0 0 1 $ VALUE CHANGER
2   4 2 2 4 10 0.8
10 5 0.2 $ SECOND LVL B
6   9 1 -6 -6 $ TYPE 9 MODEL1
9   9 1 -5 -5 $ TYPE 9 MODEL2
5   1 0.8 0.2 $ THIRD LVL C/D
3   5 2 6 0.8 10 0.2 $ DECISION C INPUT
7   9 1 -4 -4 $ TYPE 9 MODEL3
EOR
□

Table 45. GO2 Data For Standard Feedback Loop.

GO3 DATA FOR STANDARD FEEDBACK LOOP
,,,1/
EOR
□

Table 46. GO3 Data For Standard Feedback Loop.

GO1 DATA FOR COMPLEX FEEDBACK LOOP
,,,,,20/
4 1 4 1 2 3 17 $ FIRST LVL A
12 4 1 3 4 5 18 $ B PATH SPLITTER
15 8 5 6 $ VALUE CHANGER
4 2 3 7 8 19 $ SECOND LVL B
9 6 6 7 9 $ TYPE 9 MODEL1
9 9 4 8 12 $ TYPE 9 MODEL2
9 10 18 19 20 $ TYPE 9 MODEL21
2 0 3 2 6 9 10 $ OR GATE
12 14 10 2 11 21 $ B PATH SPLITTER
4 3 2 13 22 $ DECISION C INPUT
9 7 11 13 14 $ TYPE 9 MODEL3
9 11 21 22 23 $ TYPE 9 MODEL32
2 0 5 3 4 11 12 14 15 $ OR GATE
1 13 15 16 $ FOURTH LVL D
5 5 24 $ DECISION D INPUT
9 12 16 24 25 $ TYPE 9 MODEL33
2 0 7 16 17 18 20 21 23 25 26 $ OR GATE
1 13 26 27 $ FIFTH LVL E
0 26 $ FINAL SIGNALS
EOR
□

Table 47. GO1 Data For Complex Feedback Loop.

```
GO2 DATA FOR COMPLEX FEEDBACK LOOP
0/
  1   4  4 4  0 20 20 20 0.7
            20  1 20 20 0.1
            20 20  2 20 0.1
            20 20 20  7 0.1 $ FIRST LVL A
  4   12 3 0.5 0.1 0.1 $ PATH SPLITTER
  8   15 3 20 0 0 0 1 $ VALUE CHANGER
  2   4  3 3 4  20 20 0.8
            20  5 20 0.1
            20 20  8 0.1 $ SECOND LVL B
  6   9  1 -16 -16 $ TYPE 9 MODEL1
  9   9  1 -15 -15 $ TYPE 9 MODEL2
 10   9  1 -12 -12 $ TYPE 9 MODEL22
 14  12  2 0.6 0.1 $ PATH SPLITTER
  3   4  2 2 9  20 0.8
            20 10 0.2 $ SECOND LVL B
  7   9  1 -11 -11  $ TYPE 9 MODEL3
 13   1  0.8 0.2  $ THIRD LVL C/D/E
 11   9  1 -10 -10  $ TYPE 9 MODEL31
  5   5  2 11 0.8 20 0.2 $ DECISION D INPUT
 12   9  1 -9 -9  $ TYPE 9 MODEL4
EOR
□
```

Table 48. GO2 Data For Complex Feedback Loop.

```
GO3 DATA FOR COMPLEX FEEDBACK LOOP
,,,1/
EOR
□
```

Table 49. GO3 Data For Complex Feedback Loop.

LIST OF REFERENCES

Blanchard, B. S., & Fabrycky, W. J. (2006). *Systems Engineering and Analysis*. Upper Saddle River, N.J.: Pearson Prentice Hall.

Incose.org. Retrieved Sep 6, 2008, from http://www.incose.org/practice/whatissystemseng.aspx

Institute of Electrical and Electronics Engineers., American National Standards Institute., IEEE Power Engineering Society. Nuclear Power Engineering Committee. Subcommittee 5--Reliability., & IEEE Power Engineering Society. Nuclear Power Engineering Committee. (1976). IEEE standard requirements for reliability analysis in the design and operation of safety systems for nuclear power generating stations. New York, N.Y.: Institute of Electrical and Electronics Engineers.

Kaman Sciences Corporation., & Electric Power Research Institute. (1983). KSC GO Feb 1983. Palo Alto, Calif.: Electric Power Research Institute.

MIL-STD-721c. Retrieved May 24, 2008 from http://www.usace.army.mil/publications/armytm/tm5-601/a-b.pdf

MIL-STD-882c: Retrieved May 24, 2008 from http://eic.ipo.noaa.gov/IPOarchive/MAN/doc124.pdf

THIS PAGE INTENTIONALLY LEFT BLANK

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California

3. Mark Rhoades
   Systems Engineering Department
   Naval Postgraduate School
   Monterey, California