SAND REPORT
SAND2005-0020
Unlimited Release
Printed January 2005

System of Systems Modeling and Analysis

James E. Campbell, Dennis E. Longsine, Donald Shirah, and Dennis J. Anderson

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

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

Issued by Sandia National Laboratories, operated for the United States Department of Energy by Sandia Corporation.

Available to DOE and DOE contractors from
U.S. Department of Energy
Office of Scientific and Technical Information
P.O. Box 62
Oak Ridge, TN 37831

Telephone: (865)576-8401
Facsimile: (865)576-5728
E-Mail: reports@adonis.osti.gov
Online ordering: http://www.osti.gov/bridge

Available to the public from
U.S. Department of Commerce
National Technical Information Service
5285 Port Royal Rd
Springfield, VA 22161

Telephone: (800)553-6847
Facsimile: (703)605-6900
E-Mail: orders@ntis.fedworld.gov
Online order: http://www.ntis.gov/help/ordermethods.asp?loc=7-4-0#online

SAND2005-0020
Unlimited Release
Printed January 2005

System of Systems Modeling and Analysis

Dr. James E. Campbell, DMTS
System Sustainment & Readiness Technologies Department
P.O. Box 5800
Albuquerque, NM 87185-1176

Dr. Dennis E. Longsine, Consultant
Intera, Inc.
9111-A Research Boulevard
Austin, TX 78758

Donald Shirah, MTS
System Sustainment & Readiness Technologies Department
P.O. Box 5800
Albuquerque, NM 87185-1176

Dennis J. Anderson, PMTS
System Sustainment & Readiness Technologies Department
P.O. Box 5800
Albuquerque, NM 87185-1176

Abstract

This report documents the results of an LDRD program entitled "System of Systems Modeling and Analysis" that was conducted during FY 2003 and FY 2004. Systems that themselves consist of multiple systems (referred to here as System of Systems or SoS) introduce a level of complexity to systems performance analysis and optimization that is not readily addressable by existing capabilities. The objective of the "System of Systems Modeling and Analysis" project was to develop an integrated modeling and simulation environment that addresses the complex SoS modeling and analysis needs. The approach to meeting this objective involved two key efforts. First, a static analysis approach, called state modeling, has been developed that is useful for analyzing the average performance of systems over defined use conditions. The state modeling capability supports analysis and optimization of multiple systems and multiple performance measures or measures of effectiveness. The second effort involves time simulation which represents every system in the simulation using an encapsulated state model (State Model Object or SMO). The time simulation can analyze any number of systems including cross-platform dependencies and a detailed treatment of the logistics required to support the systems in a defined mission.

Acknowledgments

The System of Systems Modeling and Analysis LDRD program team would like to acknowledge the significant support, time, and effort provided to the program by Robert Cranwell, LDRD Program Manager. The team also acknowledges the support of and guidance from the members of the Modeling and Simulation Thrust of the Emerging Threats Investment Area: Russell Skocypek, Alan Nanco, John Wagner, Robert Cranwell, and Ron Trellue. Finally, the team acknowledges and thanks Craig Lawton, Leon Chapman, and Chris Atcitty for their contributions to the program.

Table of Contents

List of Figures

List of Tables

Executive Summary

Evaluating design concepts for a complex system of systems (SoS) is an immediate need for military systems like Future Combat Systems (FCS). SoS analysis requires predicting performance at the SoS level in contrast to the traditional platform-by-platform approach. SoS analysis must examine a multitude of design and technology options in order to optimize mission effectiveness across wide parameter spaces. The U.S. Army is facing the need to establish SoS performance requirements and translate these SoS requirements down to optimal or near-optimal individual platform requirements for system design and development. This challenge is further extended by the complexity presented with new technology. Currently, about the only method to gain some performance knowledge at the SoS level is through traditional warfight simulation codes, which are costly and time-consuming.

The goal of the System of Systems Modeling and Analysis LDRD program was to develop an integrated modeling and simulation (M&S) environment that addresses these complex SoS modeling and analysis needs. The approach involves developing, enhancing and integrating state modeling methodologies, time simulation methodology, and agent-based simulation objects for detailed concept and scenario analysis. The methodology has been applied to a FCS UA concept to demonstrate the approach.

To achieve goals relating to the state modeling methodologies, a state modeling capability has been added to Sandia's SyOp software. At the core of SyOp are fault trees, which are typically used to model a single system. With the addition of a state modeling capability, SyOp will gain a more powerful way to model multiple systems and to incorporate non-system elements into models of system performance.

A major step toward analyzing complex SoS analysis has been the development of a multi-system time simulation capability. This multi-system simulation capability has centered on the development of a State Model Object (SMO) that enables a system, its elements, and its functionality to be encapsulated for use in the simulation. The concept behind the multi-system simulation is illustrated in Figure 1.1.

The state model object (SMO) is the central feature of the simulation with an SMO used to represent each system in the system of systems being simulated. The controlling simulation software provides needed information on environmental conditions, terrain, use conditions, supply network information, etc. There is a scenario model that describes the detailed scenarios that the systems will follow during the simulation. A combat damage model provides a mechanism to simulate the effects of combat damage including damage to individual system primary elements or completely disabling the system. Finally, a supplies and services model provides a means for spare parts and consumables to move from system to system in the simulation and makes maintenance services available to systems that require repairs.

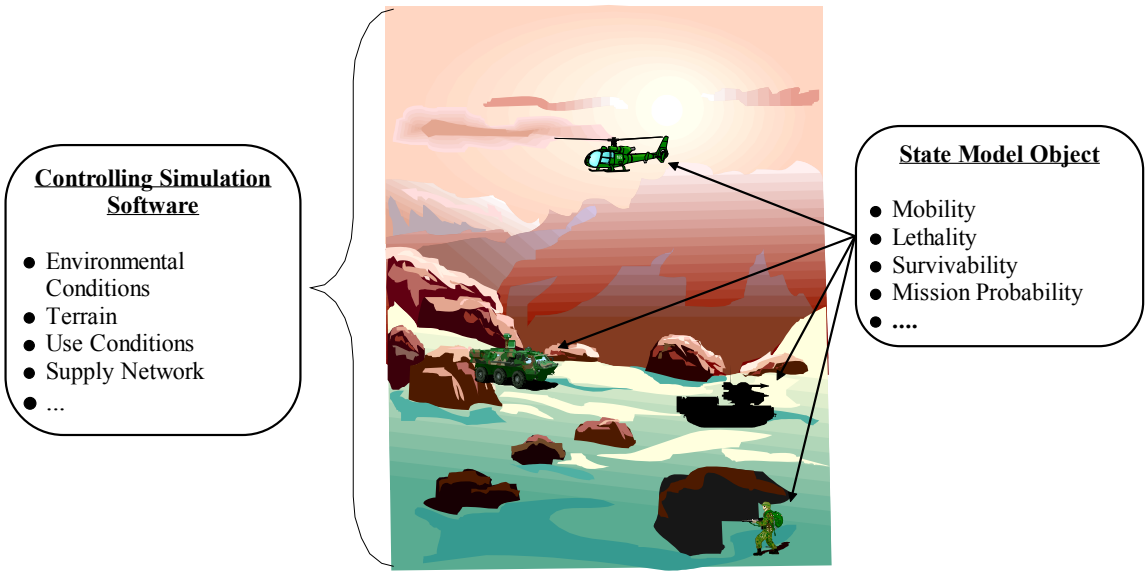

Figure 1.1 Multi-System Simulation Concept

The state modeling approach that has been added to SyOp and that forms the basis for the multi-system simulation capability has several benefits.

- A state model is quite flexible in the level of modeling detail. The approach readily adapts to high-level, overview models or to very detailed models that analyze systems in depth.
- A state model can have multiple goal states which means that multiple performance measures can be analyzed using a single model.
- A state model can have different sets of initial states. Typically results are desired for the case when every system is initially in its fully operational state. On the other hand if some systems are inoperable or are partially operable, the user can define the initial states that way.
- Goal states are not restricted to inoperable states. The state model can contain partially operable conditions.
- A state model can contain multiple systems.
- It is easy to incorporate dependencies between systems in a state model.
- External elements such as bad weather, rough terrain, or turbulence can be readily incorporated into a state model.

In summary, a SoS simulation capability has been designed, developed, and demonstrated under the SoS Modeling and Analysis LDRD that incorporates state model objects for detailed simulation of hundreds of interacting systems. This capability represents a significant accomplishment toward providing an ability to evaluate systems of systems.

This ability did not exist at the beginning of the program and, as far as is known, does not currently exist elsewhere.

As indication of the success of this LDRD, the U.S. Army, based on initial LDRD accomplishments, funded a large program with Sandia for SoS evaluation of the Future Combat Systems program, with $1.4M in FY04 non-LDRD funding. Funding for FY05-FY06 is projected to be $2.6M per year. Further, the SoS evaluation methodology has been defined as core to the Program Manager, UA Logistics Integration Directorate logistics assessment needs and the Army Evaluation Center's approach to developing test plans based on SoS performance evaluation. For application to the FCS, a comprehensive SoS logistics treatment approach was conceived and designed under this LDRD. The actual implementation in the SoS simulation was accomplished under the program with the U.S. Army.

1 Introduction

1.1 Problem Background

Evaluating design concepts for a complex system of systems (SoS) is an immediate need for military systems like Future Combat Systems (FCS). This evaluation involves predicting SoS performance and identifying critical operational parameters across a broad trade space, presenting a multidimensional research challenge. Even for a single system, performance is characterized by several measures of effectiveness (MOEs). For FCS, the SoS level is defined to be a 1,000-platform Unit of Action (UA). Analyzing the performance of several design options of a complex SoS across external parameters and multiple MOEs generates a massive number of trade space combinations, producing extreme computational issues.

The ability to evaluate performance at the SoS level is critical for FCS to achieve high performance objectives. SoS analysis requires predicting performance at the SoS level in contrast to the traditional platform-by-platform approach. SoS analysis must examine a multitude of design and technology options in order to optimize mission effectiveness across wide parameter spaces. The U.S. Army is facing the need to establish SoS performance requirements and translate these SoS requirements down to optimal or near-optimal individual platform requirements for system design and development. This challenge is further extended by the complexity presented with new technology. Currently, about the only method to gain some performance knowledge at the SoS level is through traditional warfight simulation codes, which are costly and time-consuming.

1.2 Goals and Objectives of the Project

The goal of the System of Systems Modeling and Analysis LDRD program was to develop an integrated modeling and simulation (M&S) environment that addresses the complex SoS modeling and analysis needs. The approach involves developing, enhancing and integrating state modeling methodologies, time simulation methodology, and agent-based simulation objects for detailed concept and scenario analysis. The methodology has been applied to a FCS UA concept to demonstrate the approach.

2 State Modeling

A state modeling capability has been added to Sandia's SyOp software. At the core of SyOp are fault trees, which are typically used to model a single system. State modeling provides an alternative approach to fault trees. While state modeling will not replace fault trees, it will provide a more convenient way to model multiple system functions and a system of systems.

The new State Modeling Software (SMS) component of SyOp is comprised of a user interface and a state model interpreter. They act in concert to implement the concepts described in section 3.1. An overview of the solution process is given in sections 3.2 and 3.3. Section 3.2 describes how the SMS fits into the SyOp framework and section 3.3 focuses on the specific tasks of the SMS. Instructions on how to use the interface can be found in Appendix A.

The sample problem in section 3.4 demonstrates many of the features of the SMS. The problem models an NLOS cannon, an unmanned aerial vehicle, and a forward spotter. The cannon is at full targeting capability if the UAV is operable or if there is a forward spotter available that has a functioning laser target marker. If the UAV is inoperable, there is a forward spotter, but his target laser is not operable, the spotter can relay estimated coordinates. In that case the targeting capability of the cannon is not lost but may be severely reduced.

The benefits of using the SMS are summarized in section 3.5.

2.1 Introduction and Background

The State Modeling Software (SMS) component is based on traditional dynamic state modeling found in state charts (Harel and Naamad, 1996), also called activity charts. State charts are used to define a hierarchy of states and a means of moving from state to state. The status of system(s) and their functions can be determined based on which states are occupied. The states of the system can be designed such that if a command and control vehicle loses an axle, its mobility status moves from the mobile state to the immobile state, for example. So if this latter state is occupied we know that the vehicle currently has no mobility.

The SMS offers considerable flexibility for the design of state models. The user can provide as much or as little detail as desired for a system. Generally the greater the number of detailed states the greater the potential flexibility. The mobility state for the command and control vehicle above can be broken down into states for axles, wheels, and engine parts. The finest level of detail is associated with failure events. When they occur, the failure events can trigger the move from one state to another. As required by SyOp, events must have numerical properties such as failure probability or failure rate.

The basics of state charts that are adopted by the SMS include:

- A state model contains a hierarchical system of states. Each state is either a parent state or a leaf state, that is, one with no children. Each parent state is

decomposed into its children either as an AND configuration or an OR configuration. If the system occupies an AND parent state, it must occupy each child state. If the system occupies an OR parent state, it must occupy exactly one of the children. So the OR in this case is an exclusive OR. If the system does not occupy a parent state, it cannot occupy any of its child states and vice versa.

- There is one state, called the root state, which is a parent of every state. It is the only state in the model that does not have a parent.
- The user must define a subset of the states as initial states. The system initially occupies each of these states and they cannot be conflicting. For example, two children of an OR parent state cannot both be initial states as the children would be conflicting.
- The system transitions from one set of states to the next set one step at a time. It does so by taking user-defined transitions. Figure 3.1 shows a simple example. Transition X is characterized by a source state S, a destination state D, a guard state G, and a trigger T. If at some step, the system occupies state S, the trigger T is true, and the guard state G is true then the system will transition from state S to state D. In general terms a trigger activates a transition and a guard state allows the transition. Both have to be true for the transition to fire. A transition can have a trigger, a guard state, or both.

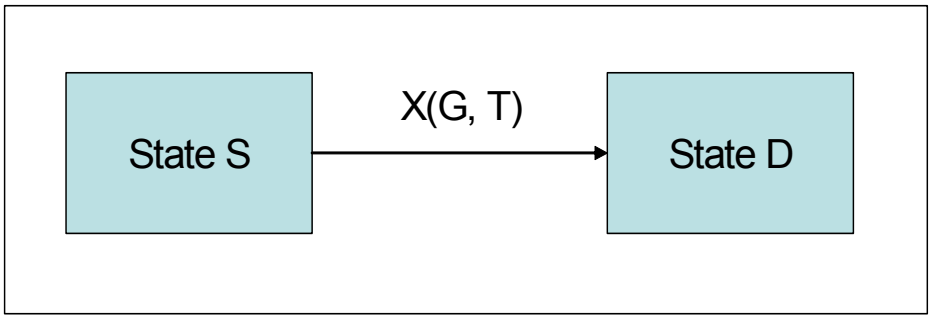

Figure 3.1 A Transition from State S to State D

- The source state for a transition is the state the transition emanates from and it can be a parent state or a leaf state.
- There can be multiple destination states for a transition, all of which must be leaf states. In traditional state charts if a destination state is a parent state, then the transition enters the destination state at a predefined default entry state. In the SMS the default entry state is included as a destination state for the transition.
- A trigger is a Boolean expression of events. The SMS determines which combinations of event occurrences cause the trigger to be true.
- Events are at the level in the SMS that can be quantified. Currently each event referenced in the trigger expressions must point to a failure mode in a SyOp data library. The failure mode must have either a failure probability or a failure rate and downtime.

- A guard is a Boolean expression of states and external elements. If the system occupies a state in a guard expression then effectively that state variable is set to true. The SMS determines which combinations of states will cause the guard to be true.
- External elements are variables that can appear in guard expressions. They are assigned a Boolean value prior to the model run. An example might be a sandstorm. For one run it could be assigned to true and for another it could be assigned to false.
- A goal state is a special state. If the system reaches this state there is a particular meaning or consequence. The primary function of the SMS is to determine what combinations of events must occur for the system to reach a goal state.

The SMS adds the concept of functions to traditional state charts. Functions are linked to goal states so that each goal state indicates some degree of functionality of the systems in the state model. Each function is evaluated for a standard set of performance measures. It is the responsibility of the user to provide the interpretation of these performance measures in the context of the function.

Figure 3.2 shows a simple system for a light. The parent states are blue and the leaf states are green. The decomposition of each parent is noted by the AND or OR to its immediate right. Transitions are labeled X1 through X4. Each transition either has a trigger T or a guard G.

The Light state has two child states: it is providing illumination or it is not. Just one of these can be true. Illumination depends on the switch, the bulb, and the electrical power. All of these have a status at all times, so the Illumination state has an AND decomposition. Each of the switch, bulb, and power are either operable or inoperable.

The three transitions on the right {X1, X2, X3} each have a trigger {T1, T2, T3}. Recall that a trigger can be any Boolean expression of events. A bulb could become inoperable if the filament fails, its contact point with the socket becomes corroded, the glass breaks, etc. If this is the desired level of detail, each of these events must be represented by a failure mode in a SyOp data library. If the event IDs are FILAMENT-FAILS, CONTACT-CORRODED, and GLASS-BREAKS, then the trigger expression is

$$\text{FILAMENT-FAILS} \cup \text{CONTACT-CORRODED} \cup \text{GLASS-BREAKS}$$

Here the union symbol indicates the Boolean OR operator. It is the inclusive OR operator so if any of these events occur, the bulb becomes inoperable.

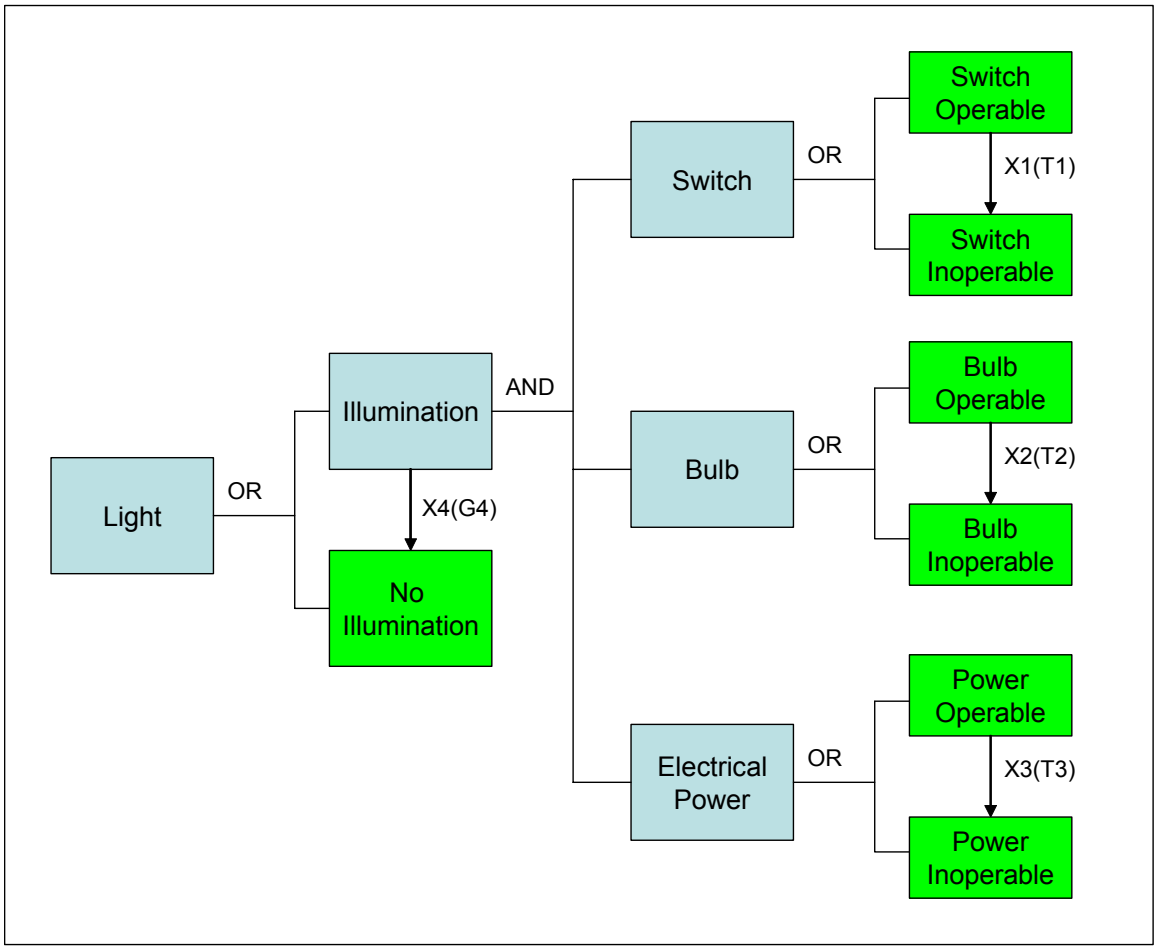

Figure 3.2 A State Model for a Light

Alternatively there could be a single event named BULB-FAILS. It is associated with a failure mode in the data library and the trigger expression is simply BULB-FAILS. The level of detail is a user decision.

Transition X4 has a guard, G4. Recall that a guard is a Boolean expression of states. It can also contain external element variables, but there are none in this simple example. The expression for the guard is

Switch Inoperable ∪ Bulb Inoperable ∪ Power Inoperable

The effect of this guard is that the light will pass from its Illumination state to its No Illumination state if the light reaches any of its states Switch Inoperable, Bulb Inoperable, or Power Inoperable.

There are four candidate goal states for this example: Switch Inoperable, Bulb Inoperable, Power Inoperable, and No Illumination. In the SMS the user can declare any subset of these as goal states. The SMS will evaluate each goal state specified in a single run, i.e., for a single solution-build command. Results will include the probability of

reaching each of the specified goal states (non-repairable model) or the frequency of reaching each of the specified goal states (repairable model).

The complete set of results depends on the functions that are tied to the goal states. For a nonrepairable analysis the user specifies what the probability of reaching the goal state means for its function. The No Illumination goal state is naturally linked to the operability function for the light. The probability of reaching this goal state is the probability that the light becomes inoperable.

For a repairable system for this example the standard performance measures would be interpreted as:

- MTBF = mean time between those occurrences when the light becomes inoperable
- Downtime = the average downtime when the light becomes inoperable
- Availability = the fraction of time that the light is operable

This interpretation can become more challenging for those cases when the function has intermediate levels. An example will be given in section 3.4.

2.2 Solution Steps

In traditional state charts the user can define an initial set of events in addition to the initial set of states. Also, each transition can cause other events to occur. The initial states and initial events together become the initial state model status. The process then begins taking steps. The initial step uses the initial status to determine which guards and triggers are true and thence to determine which transitions can fire. At the end of the step the process finds the new set of occupied states and collects the events that occur in response to the transitions that fired. Thus, a new system status is defined and the process is repeated. The typical questions posed are: which states are reachable given the initial status and can the system reach a specified state (goal state) given the initial status.

The SMS differs from traditional state charts at this point. The question posed by the SMS is: what events must occur for the model to transition from the initial states to a specified goal state. *To this end SMS assumes that any event can occur at any time*. Thus, there is no need for initial events to be specified and no need for transitions to cause other events to occur. Neither of these is included in the SMS model input.

To answer the question, the SMS finds each possible path from the initial states to the goal state. A path consists of a sequence of states that must be passed through. In parallel is the set of transitions that must fire along the path to move from state to state. The events that cause the triggers to be true for these transitions are the events of interest. These events become variables in the Boolean expression that describes which events had to occur in order to reach the goal state. Ultimately the Boolean expression is converted to an algebraic expression in order to quantify performance measures for the state model. For example, what is the probability of reaching the goal state?

It is more efficient to solve the backward problem. That is, the SMS finds the paths by starting at the goal state and working back. When an initial state is encountered, the path terminates. Alternative paths can arise from two sources: there can be more than one transition that points to a state that must be passed through or the guard expression for a transition can contain states configured with a Boolean OR.

There is a variety of approaches to finding and storing this path information. The SMS approach is motivated by the existing SyOp technology. For this reason the paths are found and stored in the form of a tree. The tree is passed to SyOp's Boolean reduction scheme that generates a Boolean expression of events in disjunctive form.

The disjunctive form is a union of intersections. In a SyOp fault tree application the events in each intersection are said to form a cutset. Thus fault trees are transformed to a union of cutsets. If any cutset occurs, the top event in the fault tree occurs. Also, the cutsets are minimal. If {E1, E2} is a cutset, there will be no larger cutsets that contain both E1 and E2.

Given the failure properties of the events, SyOp has technology to quantify each cutset and thence to quantify system performance measures. The SMS utilizes this capability from SyOp to quantify the performance measures for each function. Because the calculations are the same, the difference is the interpretation of the calculations. The user can label the results according to their meaning for the state model in general and in particular for the function/goal state.

In summary the SMS finds all paths from the goal state(s) back to initial states by traversing the transitions backwards. The SMS stores these paths in the form of a tree. Existing SyOp technology is then used to complete the solution. SyOp converts the tree into a Boolean expression of events in disjunctive form. SyOp converts this form into an algebraic expression and evaluates the expression, given input from a data library, to quantify various performance measures. It is the users' responsibility to interpret these measures in the context of their state model. The tasks performed by SyOp are discussed in SyOp documentation. The primary task of the SMS is discussed in the next section.

2.3 Finding Paths

Before initiating the path detection process, the SMS applies two state chart solution tools to the state model. First, the SMS encodes the states (Helbig and Kelb, 1994). It determines a set of Boolean state-representation variables whose values indicate whether a state is occupied or not. The variables are defined to honor the rules of AND and OR states. For example, suppose state A and state B are children of parent state C which is an OR state. Variables are configured so that if the state model occupies state A, it also occupies state C. Similarly for state B and state C. However, for the state model to occupy both A and B simultaneously at least one of the representation variables must be both true and false. Thus, the encoding scheme disallows state conflicts.

The second tool is the use of binary decision diagrams (BDDs) first introduced in 1978 (Akers, 1978). A BDD is a rooted finite directed acyclic graphical representation of a

Boolean expression. Two BDDs can be combined under Boolean operators to form a third BDD. If the BDDs are ordered (the variables always appear in order of increasing index for example) and reduced, they are unique. Some authors refer to these as ROBDDs, but most still use the simpler BDD terminology where reduced and ordered are understood.

Consider the Boolean expression **[(E1 ∪ E2) ∩ E3]**. Letting the variables increase in index from the root outward, the BDD for this expression is shown in Figure 3.3. There are five nodes numbered 0 through 4 shown in red. Terminal node 0 is reserved for the false node and terminal node 1 is reserved for the true node. All non-terminal nodes refer to a variable as shown in blue italics. Although not the case for this simple example, a variable can appear at more than one node. There is a true branch (solid line) and a false branch (dashed line) emanating from each node.

A satisfying set is a set of variable assignments that leads to the true node. From Figure 3.3, the BDD is true if both E3 and E1 are true or if both E3 and E2 are true and E1 is false. If this is a Boolean expression for a trigger in the SMS, we only are concerned with event occurrences. The nonoccurrence of an event is of no concern. Thus, the fact that E1 is false in the second set is ignored. If the Boolean expression is for a guard, the variables that must be false are included in the satisfying set. Such variables are important when interpreting which states must be occupied for the guard to be true.

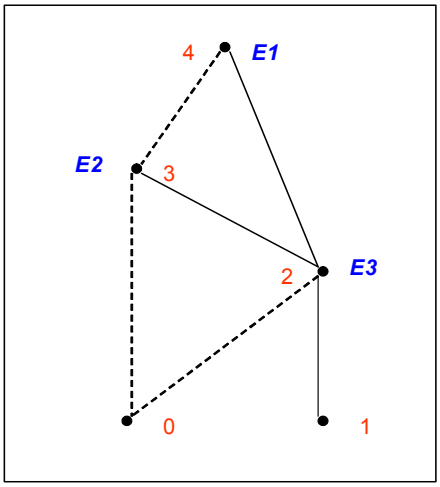

Figure 3.3 A Simple BDD Example

The process of finding the satisfying sets is equivalent to transforming the expression represented by the BDD into disjunctive form. Thus for a trigger application, the Boolean expression takes the equivalent form: **(E1 ∩ E3) ∪ (E2 ∩ E3)**.

For a guard application, states whose encoding agrees with the variable assignments are identified. Similar to a trigger the final result consists of alternative sets of states such that if the model occupies all states in the set, the guard expression is true.

Traditional state charts can use BBDs extensively. After state encoding and event encoding, BDDs are created for each state and each event. Using these, the Boolean expressions found for each guard and each trigger are cast as BDDs. For a transition to fire, its source state must be occupied, its guard must be true, and its trigger must be true. So the BDDs for each are combined with the Boolean AND operator yielding a BDD to represent the transition. Because each transition can fire or not, the BDDs for all transitions in the state model are combined using the Boolean OR operator. The resulting BDD is the state-transition BDD and it can be used at each step in the forward stepping process. The BDD is found that represents the current system status and it is combined with the state-transition BDD using the AND operator. The satisfying sets for the resulting BDD are interpreted to find the new set of occupied states and events that occur.

The SMS uses the state encoding scheme to formulate a BDD for each state. If state encoding requires N variables, the SMS starts event encoding at variable number N+1. Given these basic BDDs, the SMS finds BDDs for the Boolean expressions provided by the user for both guards and triggers. The satisfying sets are found thereby transforming the expressions into disjunctive form. This transformation step is the primary use of BDDs in the SMS. Given the backward tracing procedure used by the SMS, there is no need to find the BDD for each transition, nor for the combined transitions.

The backward path tracing starts at the goal state. The SMS finds all paths that lead to this state in the form of a tree. The tree is represented by a collection of nodes. The goal state node and intermediate state nodes have branches emanating from them. Ending nodes have no such branches. As the SMS traces backwards from state to state each new state encountered is treated with the same procedure as the starting (goal) state. So for state A in the path

1. Make a node for state A and find all transitions that point to state A. These include all transitions that have state A as a destination state. Because the SMS traces paths backwards, these transitions represent alternative upstream outlets from state A. Because these are alternatives, the node for state A is labeled as an OR node.

2. Make a node for each transition found in step 1 and determine what makes it fire. In general firing of the transition requires that the model occupies the transition's source state, the guard is true, and the trigger is true. So, the node is labeled as an AND node.

3. Make a node for the source state for the transition. If the source state is not an initial state, mark it for further investigation. At some point in the process the SMS will return to this state to continue tracing the path from this state starting at step 1.

4. Make a node for the guard for the transition, if there is one. As discussed above, the guard expression is in disjunctive form. In the general case of multiple satisfying sets which identify multiple states per set, the guard node is labeled as an OR node. Make a node for each satisfying set. Because each state belonging to a satisfying set must be occupied for the set to be true, the node for each

satisfying set is an AND node. Make nodes for each state under a satisfying set node. If a state is not an initial state, mark it for further investigation. At some point in the process the SMS will return to this state to continue tracing the path starting at step 1.

5. Make a node for the trigger for the transition, if there is one. As discussed above, the trigger expression is in disjunctive form. In the general case of multiple satisfying sets which identify multiple events per set, the trigger node is labeled as an OR node. Make a node for each satisfying set. Because each event belonging to the set must occur for the set to be true, the node for each satisfying set is an AND node. Make nodes for each event under a satisfying set node.

When step 5 is completed the SMS checks to see if any states that were marked for further investigation remain. If so, steps 1 through 5 will be repeated starting at the next such state. When finished, all branches in the tree will terminate either with an initial state or an event.

The SMS recognizes that a state could be introduced into the tree on more than one path from the initial states to the goal state. When this occurs the SMS marks the node for such a state as a transfer. This effectively places the node for the state at the top of a separate tree. In this way the subsequent tracing backward from this state is only done once and any time this state is referenced in the main tree, control passes to this separate tree. This feature saves computer time and memory.

Once the tree is constructed it undergoes three reduction steps.

1. If a path ends in a dead-end, the path is removed from the tree.
2. If a path ends with a node that does not represent an event, the node is removed from the tree.
3. For all but the goal state node, if a node has only one branch under it, the node at the end of that branch is moved up and replaces the existing node.

The SMS can encounter a dead-end path if it reaches a state that has no transitions pointing to it and the state is not an initial state. This is detected in step 1 above. At that point in the process the state is marked as a dead-end and steps 2 through 5 are skipped. All dead-ends are removed after the tree is constructed.

The removal of a dead-end state can cause a chain reaction in the tree. If the state was introduced because it is the source state for a transition, the transition can never fire. Thus, the transition and all its branches are eliminated from the tree. If the transition has only one destination state and that state has no other transitions that point to it, elimination of this transition creates a new dead-end state and the process is repeated.

If a dead-end state was introduced through a guard expression, then every satisfying set that contains this state is now false. The node for each such satisfying set is removed. If every satisfying set node is removed for a guard, the guard can never be true. This

implies that the transition that introduced the guard can never be true and the steps in the preceding paragraph are applied to the transition.

Once all dead-ends have been removed the SMS generates its first output. It creates a collection of states that are represented by nodes in the surviving tree. These are the states that appear on the possible paths from the initial states to the goal states. When the user requests path validation, the interface will display a list of the path states. The interpretation is that these states affect the functionality associated with the goal state.

The elimination of non-event nodes that terminate branches is an iterative process. The elimination of each such node can create additional nodes that have no branches. After this task, every branch in the tree must terminate with an event.

The elimination of non-event nodes that have only one branch is more of a convenience than a necessity. It is done to minimize the size of the tree.

2.4 Example Problem

2.4.1 Introduction

The example problem models three systems: a Non-Line-Of-Sight Cannon (NLOS-C), an unmanned aerial vehicle (UAV), and a forward spotter. They are linked by the fact that both the UAV and forward spotter can provide targeting information to the cannon. The potential targets for the cannon are on the order of 20km distant.

This example NLOS-C has the capability to fire smart bullets, similar to the 155mm Copperhead projectile, but smaller. The bullets have a guidance system and fins and they are capable of honing in on a target that has been painted by a laser. Thus if the target is marked by either the UAV or the forward spotter, it is assumed here that the target will be hit within 1m. This assumption implies, in part, that the bullet performs perfectly. Hence in this example we will not model the functionality of the bullet itself.

The NLOS-C has five major functions that potentially affect its operability: Mobility, Sensing, Electrical, Lethality, and Communications. In this example, loss of mobility, electrical, lethality, or communications causes the NLOS-C to become inoperable. The sensing equipment is assumed to be passive (such as a glint detector) and is used for short-range targets. So even though the sensing function is included in the model for the NLOS-C, it does not affect its operability given the long range targets anticipated for this exercise. Also given the parameters of this exercise, the M240 machine gun that the NLOS-C carries does not affect the lethality of the NLOS-C.

The UAV has four major functions that affect its operability: Mobility, Sensing, Electrical, and Communications. In this example, loss of any of these four functions makes the UAV useless to the NLOS-C. Hence in this model, loss of any of the four functions causes the UAV to become inoperable. The UAV Inoperable state can then be incorporated into guard expressions for targeting transitions for the NLOS-C.

Because of the high casualty risk involved, the army would prefer unmanned target detectors such as the UAV over the use of human spotters. The example problem will have states for spotter available and not available. If there is one available, the spotter will use a laser marker to pinpoint the target. If the laser marker is inoperable, the spotter will use landmarks and a gridded map to estimate target coordinates. With precise targeting unavailable the NLOS-C will fire dumb bullets. The circular error probable (CEP) for this case is estimated to be 150m.

By incorporating all functions for each system into the model, the assumptions of the previous three paragraphs can be easily modified. In this case the required modifications are typically confined to guard expressions.

2.4.2 State Model Construction

The steps for constructing the state model for the NLOS-C, UAV, and spotter example are given in this section. The one preliminary task is to create a SyOp data library for the events that will appear in trigger expressions for transitions in the model. The procedure for creating a data library in SyOp can be found in SyOp documentation. The failure events shown in Table 3.1 comprise the failure modes for the data library.

In a SyOp fault tree the failure events can be mapped to failure modes in a SyOp data library. This feature is planned for the SMS but has not yet been implemented. With that implementation for the failure of an axle on the NLOS-C, for example, the four failure events will point to a single failure mode in the data library. That is, all four axles are presumably of the same design and manufacture for the NLOS-C. In this problem there would be similar mapping for the wheels of the NLOS-C and the FBCB2 network and Sincgars radio that appear in both the UAV and the NLOS-C systems.

The first 32 events in Table 3.1, ending at WHEEL-4R, apply to the NLOS-C. As stated above the sensing function of the NLOS-C is not germane to this example. Also, the M240 machine gun has no use in the long-range fire model. However, states relative to these events will be included and the events will be placed into appropriate triggers. In that way the model is available for analysis under other scenarios.

The next 11 events in Table 3.1 will affect the operability of the UAV. As for the last two events we do not attempt to break down the operability of the spotter into separate functions. The spotter's laser marker will possibly fail due to the occurrence of the event LASER-MARKER. The SPOTTER-LOST event includes that the spotter is not in position, has vacated the area, or is otherwise disabled. There will be a separate state for the case of no spotter in the first place.

Although not shown in Table 3.1, each failure mode has probability distributions assigned to describe the uncertainty in failure rates, downtimes, and failure probabilities. Generally these are triangular distributions with the best-estimate being the nominal value shown and the minimum and maximum being the nominal value ±20%.

Table 3.1 Failure Events and Their Properties for the Example Problem

Function	ID	Nominal Failure Rate	Nominal Failure Probability	Nominal Downtime
Lethality	105MM-CANNON	0.00180	0.0926	4.0
Lethality	FIRE-CONTROL	0.00090	0.0474	3.0
Lethality	M240-MACHINE-GUN	0.00003	0.0014	2.0
Sensing	FLASH-DETECTOR	0.00009	0.0048	2.0
Sensing	FLIR-IMAGING	0.00036	0.0193	1.0
Sensing	FUEL-SYSTEM	0.00018	0.0097	8.0
Sensing	GLINT-DETECTOR	0.00036	0.0193	2.0
Sensing	NBC-SENSOR	0.00018	0.0097	2.0
Sensing	VISIBLE-IMAGING	0.00009	0.0048	1.0
Electric	MGV-BATTERIES	0.00031	0.0165	2.0
Electric	MGV-ELEC-SYS	0.00031	0.0165	10.0
Comm	NLOS-FBCB2-NETWORK	0.00045	0.0161	0.5
Comm	NLOS-SINCGARS-RADIO	0.00018	0.0065	0.5
Mobility	ALTERNATOR	0.00032	0.0170	6.0
Mobility	DIESEL-ENGINE	0.00054	0.0287	12.0
Mobility	INSTRUMENTATION	0.00011	0.0058	4.0
Mobility	STEERING-SYSTEM	0.00031	0.0165	8.0
Mobility	SUSPENSION	0.00013	0.0073	21.0
Mobility	TRANSFER-CASE	0.00014	0.0077	10.0
Mobility	TRANSMISSION	0.00011	0.0058	12.0
Mobility	AXLE-1	0.00013	0.0073	10.0
Mobility	AXLE-2	0.00013	0.0073	10.0
Mobility	AXLE-3	0.00013	0.0073	10.0
Mobility	AXLE-4	0.00013	0.0073	10.0
Mobility	WHEEL-1L	0.00036	0.0193	1.0
Mobility	WHEEL-1R	0.00036	0.0193	1.0
Mobility	WHEEL-2L	0.00036	0.0193	1.0
Mobility	WHEEL-2R	0.00036	0.0193	1.0
Mobility	WHEEL-3L	0.00036	0.0193	1.0
Mobility	WHEEL-3R	0.00036	0.0193	1.0
Mobility	WHEEL-4L	0.00036	0.0193	1.0
Mobility	WHEEL-4R	0.00036	0.0193	1.0
Sensing	ADVANCED-EO-IR	0.00009	0.0016	2.0
Sensing	HYPER-SPECTRAL	0.00009	0.0016	2.0
Sensing	LASER-RANGE-FINDER	0.00009	0.0016	2.0
Sensing	SAR	0.00225	0.0397	2.0
Sensing	TARGET-MARKER	0.00009	0.0016	2.0
Mobility	AIR-FRAME	0.00450	0.0778	1.0
Mobility	CONTROL-SYSTEM	0.00450	0.0778	1.0
Mobility	PROPULSION	0.00360	0.0627	1.0
Electric	UAV-BATTERIES	0.00180	0.0319	1.0
Comm	UAV-FBCB2-NETWORK	0.00045	0.0161	0.5
Comm	UAV-SINCGARS-RADIO	0.00018	0.0065	0.5
	LASER-MARKER	0.00090	0.0162	2.0
	SPOTTER-LOST	0.00963	0.5000	4.0

We use the data in Table 3.1 to create a SyOp data library named NLOS_UAV.RDL. The other required property for each record in the data library is failure mode name. These were generally the same as the failure mode ID except without the dashes and using mixed case. It is not a requirement that the data library exist prior to building a state model, but there are two advantages. Selecting failure events for trigger expressions from a list is easier than typing them in and this also helps minimize typing errors.

Given that the data library has been constructed, run the SMS and select New from the file menu. The new file wizard prompts for run parameters. Select the Non-Repairable

radio button as shown in Figure 3.4. This means that the results will be expressed in terms of probabilities. Change the number of trials to 200, the mission time to 72 hours, and the seed for the random number generator to 8312004, as shown in Figure 3.4.

On the next step specify the data library name. Browse to the appropriate directory and select NLOS_UAV.RDL.

On the functions page define four functions:

1. Targeting CEP 150m @ 20km. If the UAV and the spotter's laser marker become inoperable, targeting accuracy drops to this level.
2. No Targeting. If both the UAV and the spotter are inoperable, there is no long range targeting capability.
3. No Lethality. If the NLOS-C loses its targeting capability, the 105mm cannon, or the fire control, it loses its lethality.
4. NLOS Inoperable. If the NLOS-C loses its lethality, mobility, electrical system, or communications, it becomes inoperable.

We will define a goal state for each one of these as part of the state model input. The next step is to interpret the performance measures for each function.

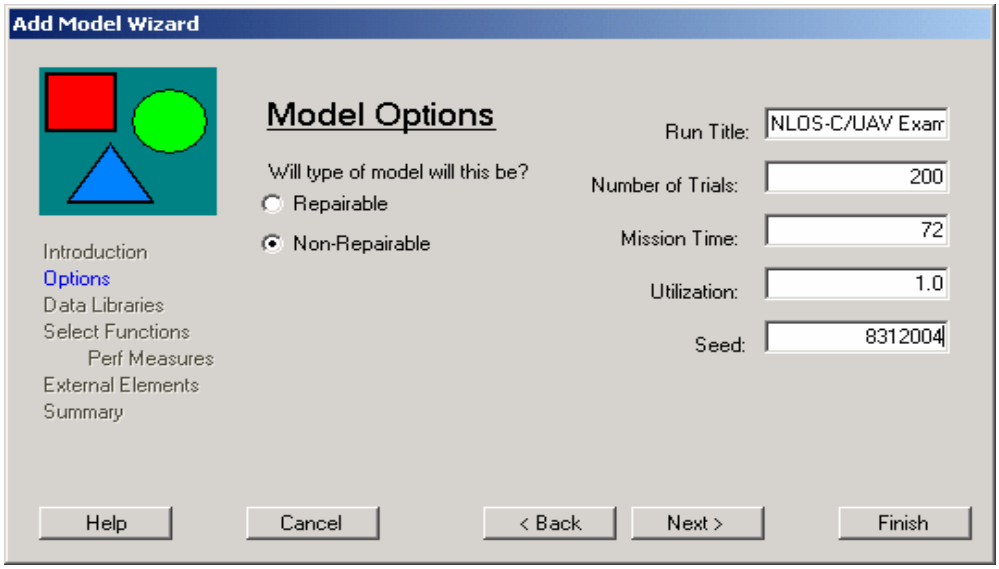

Figure 3.4 Entering Model Options for the Example Problem

Generally each function has three performance measures that must be interpreted for a non-repairable model. We ignore the Cost measure for this example and interpret only reliability and unreliability (failure probability). Because SyOp is based on a failure equation and data libraries contain failure data, *unreliability is the probability that the model reaches the goal state associated with the function*. On the other hand reliability is the probability that the model does not reach the goal state.

For each performance measure for each function we must define a caption, a label, and units, all for use by SyOp's Results Viewer. The captions are used as menu selections. The labels are placed on all plots and tabular output. Units are used as column headers and axis labels. Table 3.2 lists the text as defined for this example.

The last input form for the new file wizard prompts for external elements. This example problem does not include any, so the form is skipped. The SMS exits the wizard and displays a state model building form that has one state already in place. All state models are anchored to a root state. It can be renamed but not eliminated. The first step in the state model building process is to add children to the root state.

Table 3.2 Captions and Labels for Metrics for Nonrepairable Problem

Function	SyOp Metric	Menu Caption	Label
Targeting CEP 150m @ 20km	Reliability	Prob Not at CEP = 150m	Probability of Not Operating at CEP = 150m @ 20km
	Unreliability	Prob CEP = 150m	Probability of Operating at CEP = 150m @ 20km
No Targeting	Reliability	Prob Targeting	Probability of Some Targeting Capability
	Unreliability	Prob No Targeting	Probability of No Targeting Capability
NLOS Lethality	Reliability	Prob Lethality	Probability of Lethality
	Unreliability	Prob No Lethality	Probability That Lethality Fails
NLOS Inoperable	Reliability	Prob NLOS Success	Probability of NLOS Cannon Success
	Unreliability	Prob NLOS Failure	Probability of NLOS Cannon Failure

Figure 3.5 shows the state model after we have taken the following steps.

- Add two child states to the root state and define the root state decomposition as an OR. Name the two child states as Mission Operable and Mission Failed.
- Add three child states to Mission Operable and define its decomposition as an AND. Name the three child states as UAV, NLOS Cannon, and Forward Spotter.
- Each of these three child states will be developed into their functions and components. We will show the development for part of the UAV as an example.

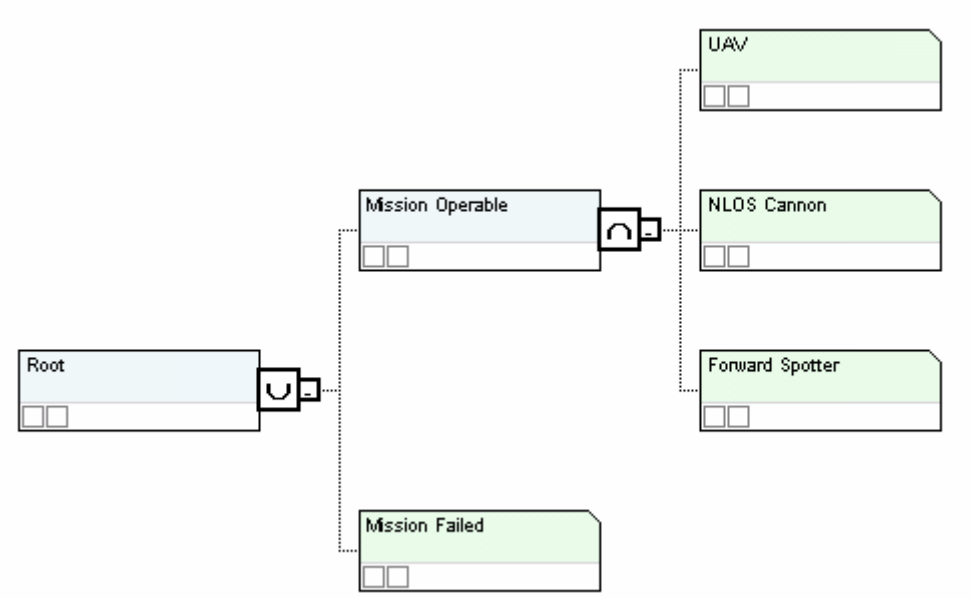

Figure 3.5 First Six States for the Example Problem

The state model for the UAV is fairly straightforward.

- Add two child states UAV Operable and UAV Inoperable and ensure that UAV has an OR decomposition.
- Add four child states to UAV Operable named UAV Mobility, UAV Sensing, UAV Comm, and UAV Elec Power and change UAV Operable to an AND decomposition.
- Add two child states to UAV Comm named UAV Comm Operable and UAV Comm Inoperable and ensure that UAV Comm has an OR decomposition
- Add two child states to UAV Comm Operable named UAV FBCB2 Network and UAV Sincgars Radio and change UAV Comm Operable to an AND decomposition.
- To each of UAV FBCB2 Network and UAV Sincgars Radio add two states and ensure that their decomposition is OR. The child state names are UAV FBCB2 Operable, UAV FBCB2 Inoperable, UAV Sincgars Operable, and UAV Sincgars Inoperable.

Figure 3.6 shows the states added to the UAV Operable state with the communications function shown in detail.

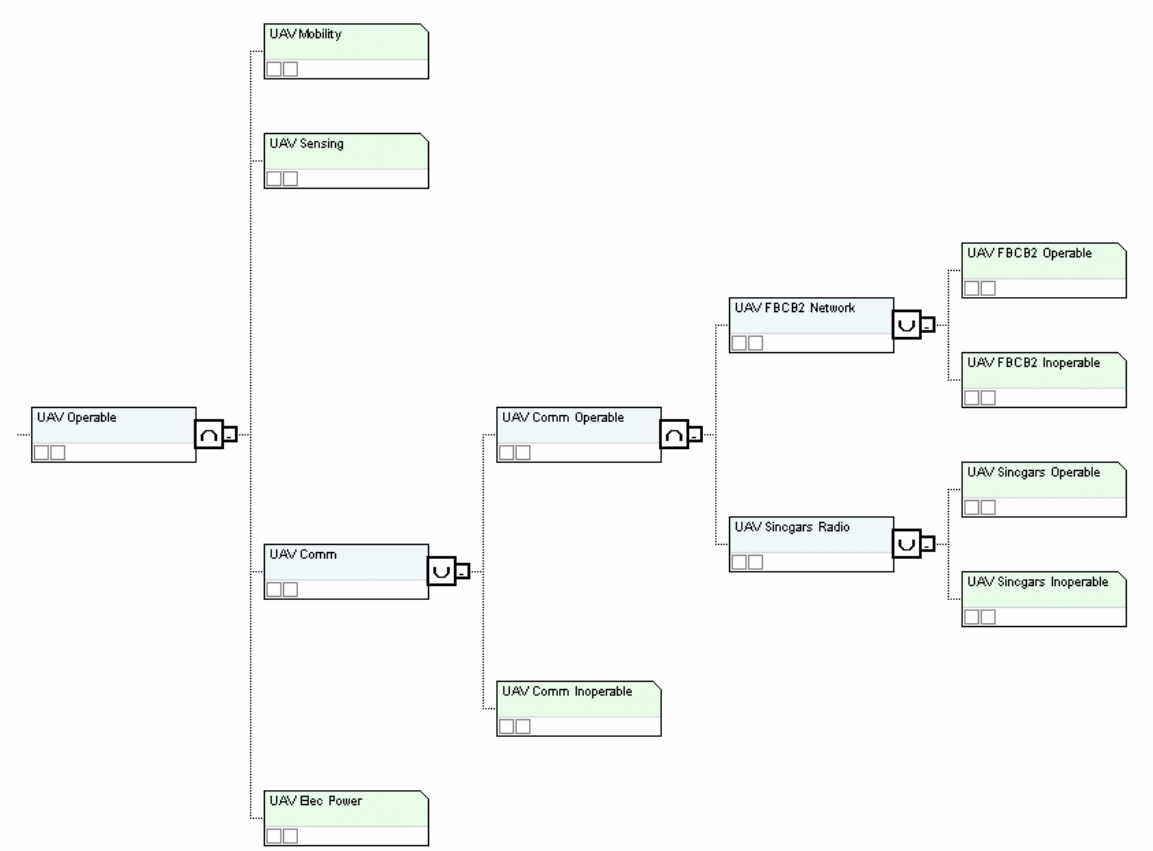

Figure 3.6 UAV Operable States with Communication States Completed

For this particular section of the state model there will be three transitions added. It is good practice to add transitions only after the entire state structure has been defined. For now we note that there will be a transition from UAV FBCB2 Operable to UAV FBCB2 Inoperable and another from UAV Sincgars Operable to UAV Sincgars Inoperable. Each will have a trigger whose expression consists of a single event (Table 3.1): UAV-FBCB2-NETWORK or UAV-SINCGARS-RADIO. The third transition will be from UAV Comm Operable to UAV Comm Inoperable. It will have a guard whose expression is:

UAV FBCB2 Inoperable ∩ UAV Sincgars Inoperable

The intersection symbol indicates the Boolean AND operator. It means that both the network and the radio have to fail for the communications function for the UAV to fail.

The other three functions for the UAV (Mobility, Sensing, and Electrical) are expanded in a similar manner. Each function is operable or not. The operable state has a number of child states based on the components selected to model that state. The child states each have two children which are operable and inoperable. The trigger expressions for the transitions between these states each contain an event from Table 3.1. The guard expressions for the operability of the function are based on:

1. Mobility. If any of the air frame, the propulsion, or the control system fail, the UAV loses its mobility.

2. Electrical. If the batteries fail, the electrical system fails.

3. Sensing. If the advanced EO-IR, hyper-spectral, and SAR all fail or if either of the laser range finder or target marker fails, the UAV loses its sensing capability.

There are five functions for the NLOS-C: Mobility, Sensing, Electrical, Lethality, and Communications. Each function state has an OR decomposition with operable and inoperable child states. The operable states are expanded as follows.

1. Mobility. The NLOS Mobile state has two intermediate children: NLOS Axles/Wheels and NLOS Engine/Drivetrain. If either the wheels/axles or the engine/drive train fail, the mobility becomes inoperable. The NLOS Axles/Wheels state has children NLOS Axles and NLOS Wheels. The wheels fail if any 2 of the 8 wheels fail. The axles fail if any of the four axles fail. The engine/drive train fails if any of the following fail: alternator, diesel engine, fuel system, instrumentation, steering system, suspension, transfer case, or transmission.

2. Electrical. If the batteries or electrical system fail, the electrical fails.

3. Communications. If both the Sincgars radio and the FBCB2 network fail, communications fail.

4. Sensing. If the glint detector, the flash detector, the NBC sensor, FLIR imaging, or visible imaging fail, sensing fails.

5. Lethality. If the targeting capability, the 105mm cannon, or the fire control fail, the NLOS-C loses its lethality.

The transition from NLOS Wheels Operable to NLOS Wheels Inoperable has a guard with a tedious expression. Each of the eight wheels is numbered according to its position on the vehicle and each has a separate state assigned, which itself has operable and inoperable child states. The transition from NLOS Wheels Operable to NLOS Wheels Inoperable occurs when any two of the eight individual wheels reach their inoperable state. The 28 possible combinations must be explicitly included in the guard expression. Using the ampersand for the Boolean AND operator and the plus sign for the Boolean OR operator, the guard expression is shown in Figure 3.7.

In future versions of the software the SMS will accommodate load-sharing lists for states in guard expressions. At that point the user will be required to supply the names of the eight wheel-inoperable states and the minimum number that allow the transition to fire, in this case two. This shortcut will be allowed as long as no event that causes wheel failure appears outside the triggers for the wheel states.

```
(NLOS Wheel 1L Inoperable&NLOS Wheel 1R Inoperable)+(NLOS Wheel 1L Inoperable&NLOS Wheel 2L Inoperable)+
(NLOS Wheel 1L Inoperable&NLOS Wheel 2R Inoperable)+(NLOS Wheel 1L Inoperable&NLOS Wheel 3L Inoperable)+
(NLOS Wheel 1L Inoperable&NLOS Wheel 3R Inoperable)+(NLOS Wheel 1L Inoperable&NLOS Wheel 4L Inoperable)+
(NLOS Wheel 1L Inoperable&NLOS Wheel 4R Inoperable)+(NLOS Wheel 1R Inoperable&NLOS Wheel 2L Inoperable)+
(NLOS Wheel 1R Inoperable&NLOS Wheel 2R Inoperable)+(NLOS Wheel 1R Inoperable&NLOS Wheel 3L Inoperable)+
(NLOS Wheel 1R Inoperable&NLOS Wheel 3R Inoperable)+(NLOS Wheel 1R Inoperable&NLOS Wheel 4L Inoperable)+
(NLOS Wheel 1R Inoperable&NLOS Wheel 4R Inoperable)+(NLOS Wheel 2L Inoperable&NLOS Wheel 2R Inoperable)+
(NLOS Wheel 2L Inoperable&NLOS Wheel 3L Inoperable)+(NLOS Wheel 2L Inoperable&NLOS Wheel 3R Inoperable)+
(NLOS Wheel 2L Inoperable&NLOS Wheel 4L Inoperable)+(NLOS Wheel 2L Inoperable&NLOS Wheel 4R Inoperable)+
(NLOS Wheel 2R Inoperable&NLOS Wheel 3L Inoperable)+(NLOS Wheel 2R Inoperable&NLOS Wheel 3R Inoperable)+
(NLOS Wheel 2R Inoperable&NLOS Wheel 4L Inoperable)+(NLOS Wheel 2R Inoperable&NLOS Wheel 4R Inoperable)+
(NLOS Wheel 3L Inoperable&NLOS Wheel 3R Inoperable)+(NLOS Wheel 3L Inoperable&NLOS Wheel 4L Inoperable)+
(NLOS Wheel 3L Inoperable&NLOS Wheel 4R Inoperable)+(NLOS Wheel 3R Inoperable&NLOS Wheel 4L Inoperable)+
(NLOS Wheel 3R Inoperable&NLOS Wheel 4R Inoperable)+(NLOS Wheel 4L Inoperable&NLOS Wheel 4R Inoperable)
```

Figure 3.7 Guard Expression for Failure of NLOS-C Wheels

The targeting function that falls under the lethality function for the NLOS-C has a special treatment in this example. Figure 3.8 shows the hierarchy of the states.

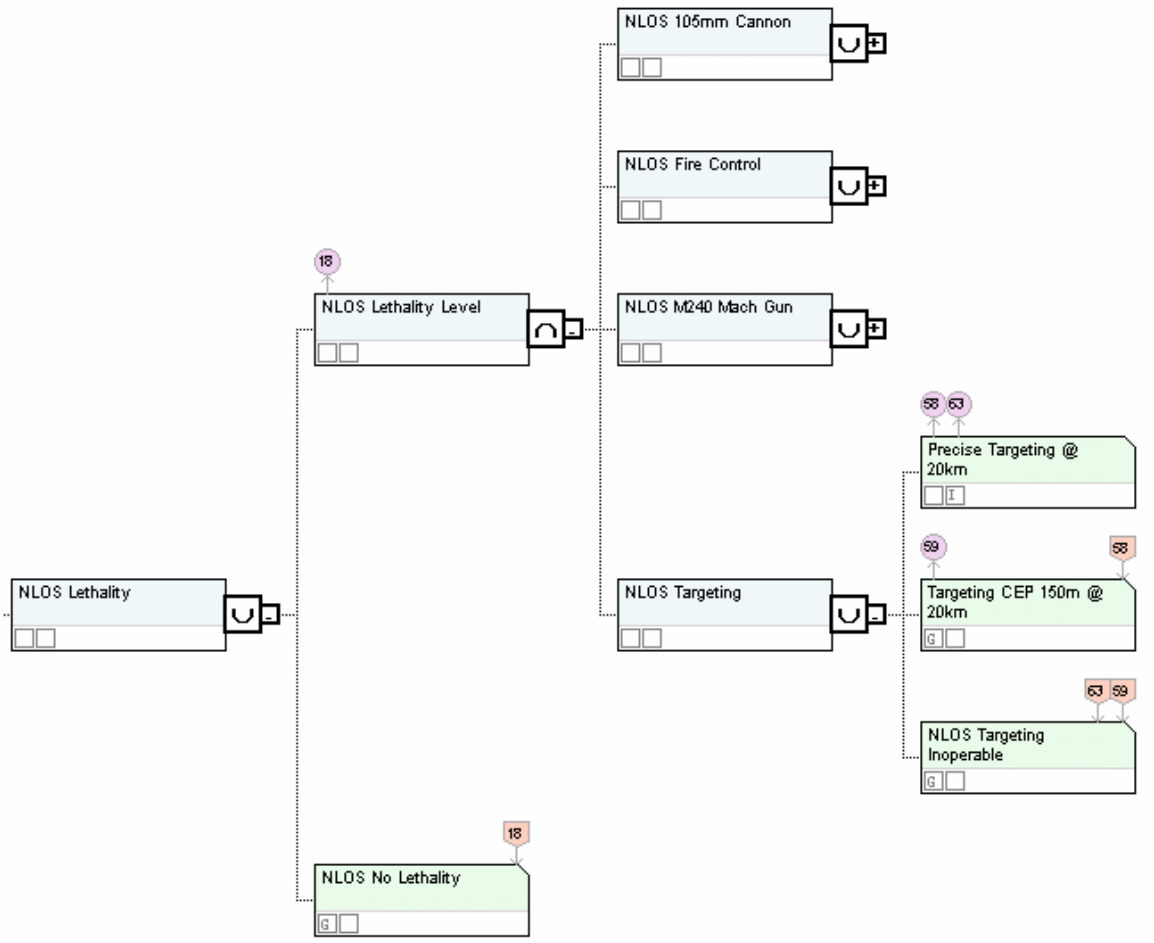

Figure 3.8 Targeting Function for the NLOS-C

There are three transitions shown in the block of targeting states. Transition 58 originates in the Precise Targeting @ 20km state and points to the Targeting CEP 150m @ 20km state. The guard expression for the transition is UAV Inoperable & Spotter Operable &

Laser Marker Inoperable. This implies that there is a spotter in place but his laser marker is inoperable and the UAV is inoperable so it cannot mark the target. Only in this circumstance will the model resort to the old-fashioned way of targeting. This reduces targeting accuracy to a CEP of 150m at 20km.

Transition 59 originates in the Targeting CEP 150m @ 20km state and points to the NLOS Targeting Inoperable state. The guard expression for the transition is Spotter Inoperable. So if the NLOS-C is already at reduced precision by relying on a spotter, it loses targeting totally if the spotter leaves the area or becomes disabled.

Transition 63 originates in the Precise Targeting @ 20km state and points to the NLOS Targeting Inoperable state. The guard expression for the transition is UAV Inoperable & (No Spotter in Area OR Spotter Inoperable). This transition bypasses the reduced targeting state and goes straight to the no targeting state if the UAV becomes inoperable and either there was not a spotter in the area or the spotter was there and is now inoperable.

In Figure 3.8 note that the Precise Targeting @ 20km state is marked as an initial state, whereas both the Targeting CEP 150m @ 20km state and the NLOS Targeting Inoperable state are marked as goal states. These two goal states are associated with the functions Targeting CEP 150m @ 20km and No Targeting.

The third system in this example is the forward spotter. The states and transitions are shown in Figure 3.9.

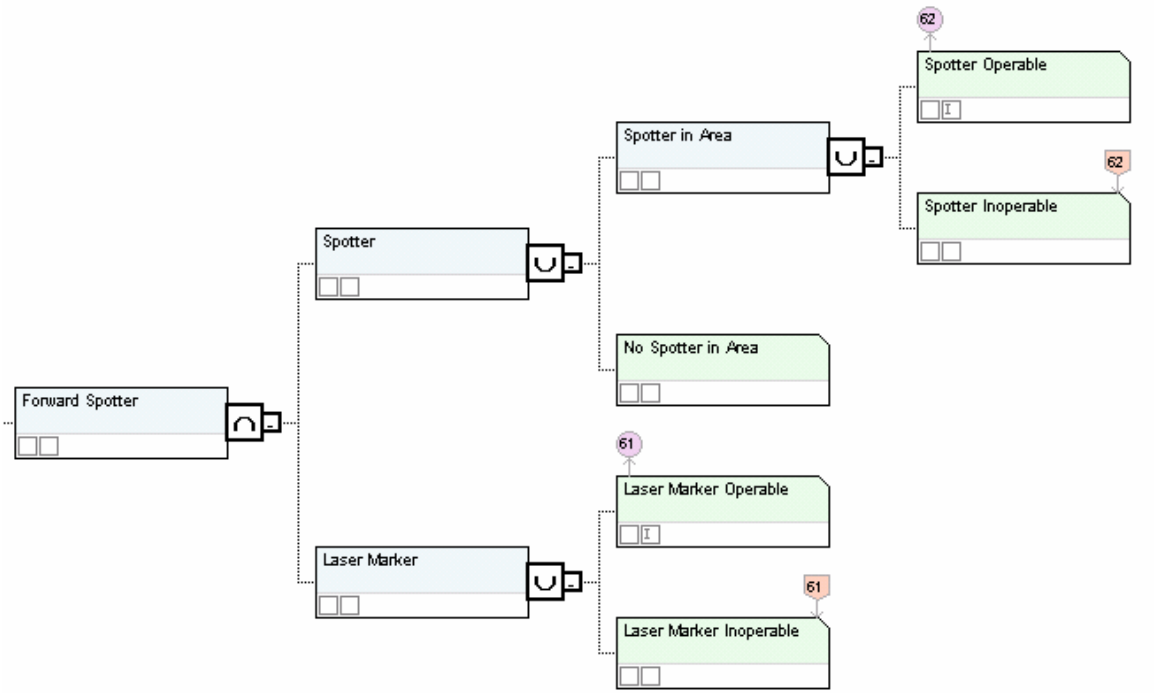

Figure 3.9 State Model for the Forward Spotter

For this example we assume that there is a spotter in the area and that the spotter is initially capable of providing targeting information. So the Spotter Operable state is an initial state. If none was available in the area the No Spotter in Area state would be an initial state. Transitions number 61 and 62 have triggers whose expressions are the single events LASER-MARKER and SPOTTER-LOST, respectively.

For additional information for the analysis, the states named NLOS No Lethality and NLOS Inoperable are also marked as goal states. They are associated with the No Lethality and NLOS Inoperable functions, respectively. In addition to the initial states already discussed, for all leaf state pairs of Operable/Inoperable states the Operable state is marked as an initial state.

The SMS generates results for each goal state that has an associated function. This example has four such (goal state, function) pairs. Currently the SMS is required to use the same set of initial states for all four model evaluations. Future versions will allow the user to assign ({initial states}, goal state, function) triplets.

2.4.3 State Model Results

When the state model for the example problem is run the SyOp Results Viewer is automatically accessed. Figure 3.10 shows histograms for two of the functions. The probability of operating at the CEP of 150m is much smaller than the probability of no targeting capability. We estimate the probability of operating at full targeting accuracy by adding these probabilities and subtracting from 1.0.

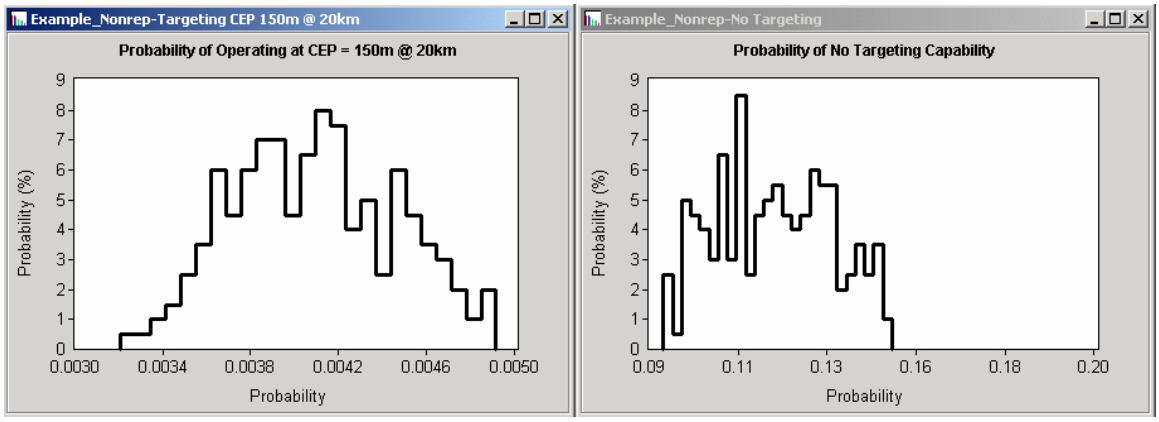

Figure 3.10 Histograms of Probability for the Targeting Functions

The SMS does not currently have he capability to do the subtraction on a trial by trial basis, but such capability is planned for future versions. So even though results have uncertainty, we are required to focus on the nominal results. Nominal results are available under the Summary Statistics menu in the SyOp Results Viewer. They are:

- Probability of operating at CEP 150m @ 20km ≈ 0.0041
- Probability of no targeting = 0.1210

Thus, the estimate for the probability of operating at full accuracy is 0.8749.

The first approximation is good to first order. Although the estimates for the probabilities are quite good for this example, future versions of the SMS will likely provide more accurate values for general cases. In addition the user will have the opportunity to specify how the performance measures should be combined, if at all. In this way customized results can be found for each trial and statistics will be available.

Figure 3.11 shows the events that are the important contributors to the magnitude of the probabilities of reaching the two goal states. The contribution of the loss of the spotter and the spotter's laser marker are greater than the contributions of the individual components of the UAV. However, adding the individual UAV component contributions shows that the loss of the UAV has about the same contribution as the loss of the spotter or the spotter's laser marker.

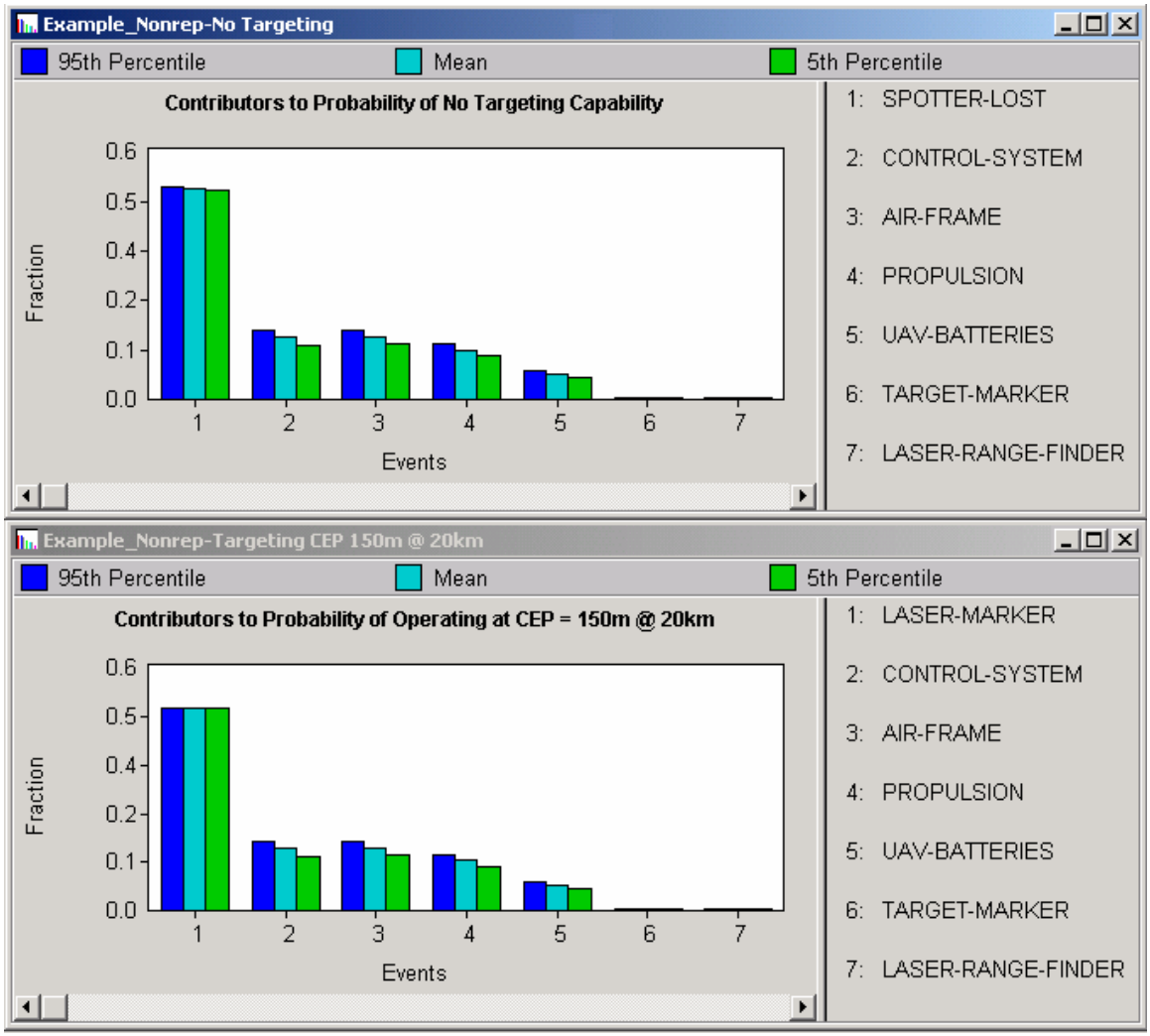

Figure 3.11 Contributors to the Probability of Reaching Goal States

Figure 3.12 shows histograms for the other two functions that were included in the run. The SMS does not have a limit on the number of function/goal state pairs. Each is evaluated separately but all can be viewed simultaneously using the Results Viewer.

The analyst may be more interested in the frequency of failure rather than the probability of failure. If so, the state model should be run as a repairable system. On the form shown in Figure 3.4, the Repairable button should be selected and the utilization should be addressed. In this example we assume that the utilization is 1.0.

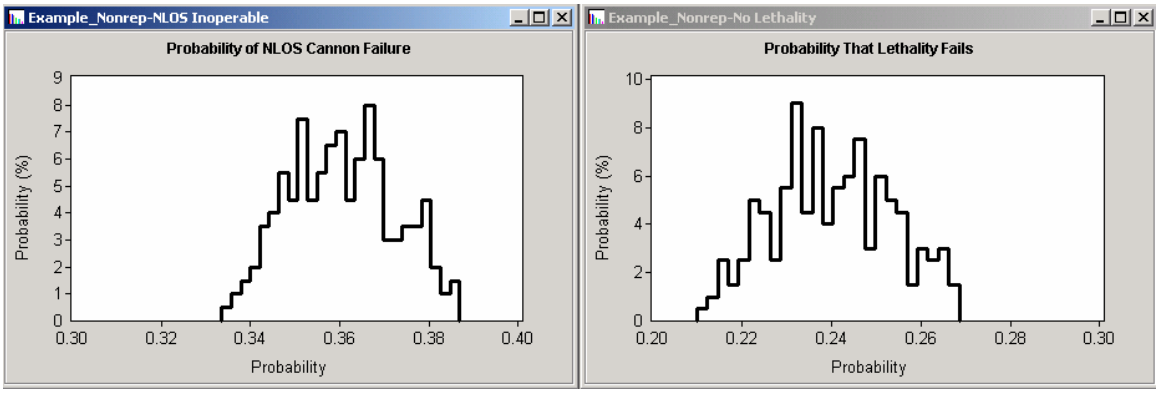

Figure 3.12 Histograms of Probability for NLOS-C Operability and Lethality

For a repairable system there are four performance measures per function. Again we will ignore cost for this example and address the standard SyOp measures of mean time between failures (MTBF), downtime, and availability. The interpretation of these metrics is fairly straightforward for the three functions No Targeting, NLOS Lethality, and NLOS Inoperable. The interpretations are given in Table 3.3.

The standard measures for the function Targeting CEP 150m @ 20km are less clear. MTBF in this case is the mean time between occurrences of dropping from precise targeting to Targeting CEP 150m @ 20km. This is difficult to squeeze into a short menu caption so the caption MTB Resorting to 150m CEP is defined (Table 3.4). A second interpretation of this MTBF is the time not spent in the Targeting CEP 150m @ 20km state.

The downtime for this function is the time spent in this state while making repairs to enable the return to precise targeting. Hence, this is a reasonable approximation to the average time spent in the Targeting CEP 150m @ 20km state.

SyOp calculates the availability metric as MTBF / (MTBF + Downtime). Unavailability is the complement of this or Downtime / (MTBF + Downtime). Using the second interpretation for MTBF above, this latter definition is a measure of the availability for the Targeting CEP 150m @ 20km function. Because SyOp reports the complement, the availability metric should be interpreted as the unavailability of the Targeting CEP 150m @ 20km function.

Table 3.3 Captions and Labels for Metrics for Repairable Problem

Function	Metric	Menu Caption	Label
NLOS Inoperable	MTBF	MTB NLOS-C Failures	Mean Time Between NLOS Cannon Failures
	Availability	Availability NLOS-C	Availability of the NLOS Cannon
	Downtime	NLOS-C Downtime	Downtime for NLOS Cannon Failures
NLOS Lethality	MTBF	MTB Lethality Failures	Mean Time Between Lethality Failures
	Availability	Lethality Availability	Lethality Availability
	Downtime	Lethality Downtime	Downtime when Lethality Fails
No Targeting	MTBF	MTB NLOS-C Targeting Failures	Mean Time Between NLOS Cannon Targeting Failures
	Availability	Availability NLOS-C Targeting	Availability of NLOS Cannon Targeting
	Downtime	NLOS-C Targeting Downtime	Downtime for NLOS Cannon Targeting Failures
Targeting CEP 150m @ 20km	MTBF	MTB Resorting to CEP 150m	Mean Time Between Resorting to Targeting CEP 150m @ 20km
	Availability	Unavailability	Unavailability of Targeting CEP 150m @ 20km
	Downtime	Time at Targeting CEP 150m	Mean Time Spent Per Occurrence at Targeting CEP 150m @ 20km

Figure 3.13 shows histograms for four of the performance measures obtained from the SyOp Results Viewer. These can be used to determine the mean time spent in each of the states Precise Targeting @ 20km, Targeting CEP 150m @ 20km, and NLOS Targeting Inoperable.

The histogram on the upper left is the mean time spent in the NLOS Targeting Inoperable state. The one on the upper right approximates the mean time spent per occurrence in the Targeting CEP 150m @ 20km state. The two histograms on the bottom both show a measure of the mean time between failures of precise targeting. Restated, they show the average time spent in the Precise Targeting @ 20km state. The one on the left measures the length of time between total failures and the one on the right measures the length of time between failures that drop the targeting precision to a CEP of 150m @ 20km. Because the times on the left histogram are much smaller than those on the right, the histogram on the left is more indicative of the mean time spent in the Precise Targeting @ 20km state.

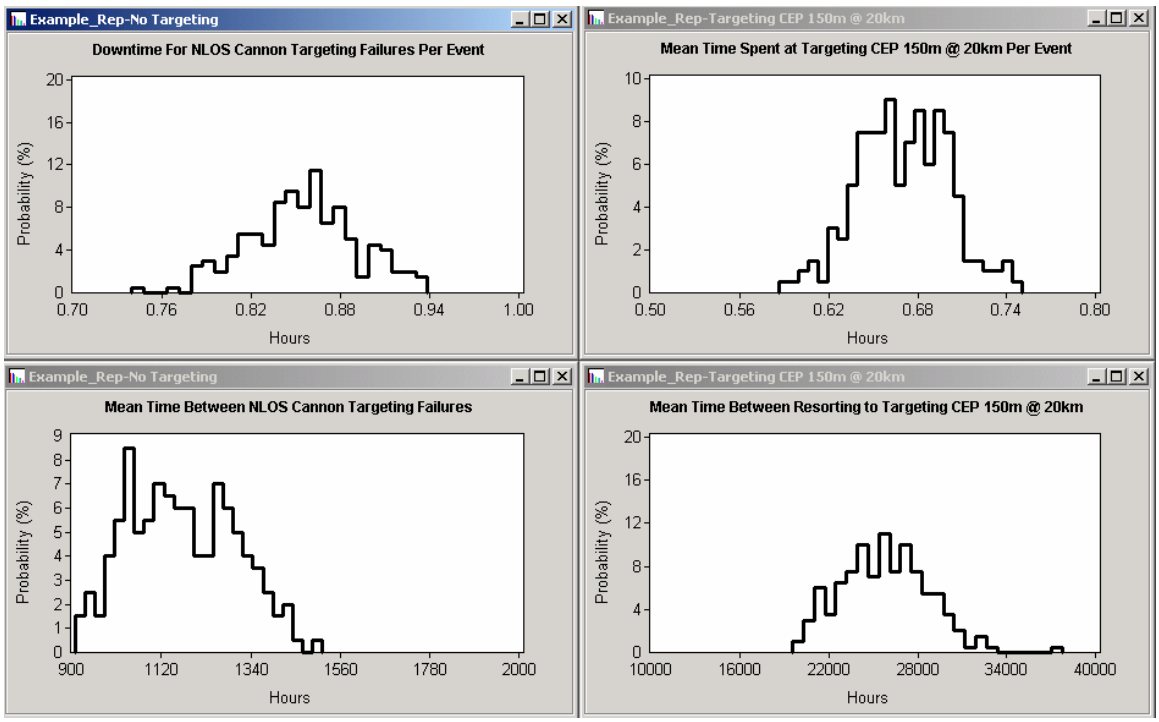

Figure 3.13 Example Histograms for Repairable Results

The mean values for the appropriate measures are found to be:

- Downtime for NLOS Cannon Targeting Failures Per Event = 0.85 hrs
- Mean Time Spent at Targeting CEP 150m @ 20km Per Event = 0.67 hrs
- Mean Time Between NLOS Cannon Targeting Failures = 1171.37 hrs

Changing these to normalized percentages the percentage of time spent in each targeting state is:

- Percent of time spent in the NLOS Targeting Inoperable state = 0.07%
- Percent of time spent in the Targeting CEP 150m @ 20km state = 0.06%
- Percent of time spent in the Precise Targeting @ 20km state = 99.87%

The example problem has demonstrated some of the flexibility of the SMS. Although the focus of the discussion was on targeting capability, inoperability results for the NLOS-C and the lethality of NLOS-C were also generated and can be viewed. There could be more detail added or some of the detail could be consolidated. The interdependence of systems was shown to be easy to incorporate. This problem does not include external elements, but it is simple to define such elements and add them to guard expressions.

There are four areas where future versions of the SMS will make the software more user-friendly.

1. The failure events will be mapped to failure modes in the supporting data library.

2. Special load sharing lists will be incorporated to simplify guard expressions.
3. The process of interpreting results will be simplified.
4. The user will be able to customize the results.

2.5 Benefits of State Modeling

State modeling offers several benefits.

- A state model is quite flexible concerning the definition of states and transitions, in particular the trigger expressions for the transitions. In general the more states that are included the simpler the trigger expressions. On the other hand, the number of states can be reduced by defining more complex trigger expressions. If the detailed states are required because it is anticipated that they could be a goal state, it is important to include them in the state model detail. Otherwise, their inclusion becomes optional. For example, a function of a system may go from its operable state to its inoperable state for a variety of reasons. The light fails because the switch fails, the bulb fails, or the electrical power fails. As presented in section 3.1.1, each of these components was assigned a state with operable and inoperable child states. When any one of these reached their inoperable state, transition X4 fired which moved the light to its No Illumination state. We could have eliminated all of the children of the Illumination state and replaced the guard in transition X4 with a trigger that described the failure of the events.
- A state model can have multiple goal states. When the SMS builds a solution, the solution contains results for every goal state in the model. Results for all goal states can be displayed simultaneously using SyOp's Results Viewer.
- A state model can have different sets of initial states. Typically results are desired for the case when every system is initially in its fully operational state. On the other hand if some systems are inoperable or are partially operable, the user can define the initial states that way.
- Goal states are not restricted to inoperable states. The state model can contain partially operable conditions. A military system that has two weapons has full lethality when both are functioning, has partial lethality when one is down and the other is up, and has no lethality when both are down. The state model can define the conditions that reduce the lethality function from full to partial and the partial lethality state can be defined as a goal state. It can be informative to run the model with both partial and no lethality as goal states. This arrangement can be used to estimate the fraction of time spent in each state for example.
- A state model can contain multiple systems. Suppose a state model has 10 assault guns defined. What is the probability that 7 are operable at the end of a specified mission time? This question can be incorporated into a guard expression.
- It is easy to incorporate dependencies between systems in a state model. Suppose that the targeting for an NLOS Cannon depends on coordinates fed to it from an unmanned aerial vehicle. If the UAV loses its mobility, its sensing capability, or

its ability to communicate with the NLOS-C, it is no longer useful as a targeting mechanism. By incorporating both systems into a single state model, it becomes easy to include the functions of the UAV into a guard expression for the NLOS-C.

- It is simple to incorporate external elements into a state model. The occurrence of bad weather, rough terrain, or turbulence, for example can be defined as an external element and incorporated into guard expressions. At the beginning of a run the user defines each of these as true or false. This can disable or enable different parts of the guard expression causing the SMS to examine different paths.

3 System of Systems Simulation

3.1 Overview

A major step toward analyzing complex SoS analysis has been the development of a multi-system time simulation capability. Key to the multi-system simulation capability has been the development of a State Model Object (SMO) that enables a system, its elements, and its functionality to be encapsulated for use in the simulation. The concept behind the multi-system simulation is illustrated in Figure 4.1.

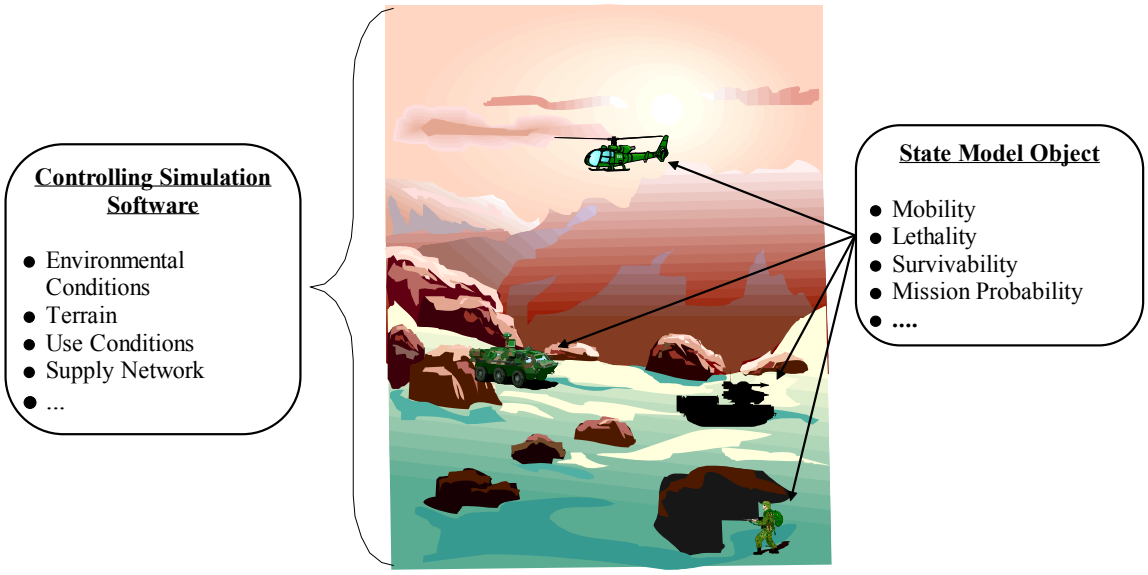

Figure 4.1 Multi-System Simulation Concept

Every system in the simulation is represented by an SMO which models the system's functionality while the controlling simulation software provides needed information on environmental conditions, terrain, use conditions, supply network information, etc.

A simplified view of the SMO Simulation architecture is shown in Figure 4.2. The state model object is the central feature of the simulation with an SMO used to represent each system in the system of systems being simulated. A scenario model describes the detailed scenarios that the systems will follow during the simulation. A combat damage model provides a mechanism to simulate the effects of combat damage including damage to individual system primary elements or damage that completely disables the system. Finally, a supplies and services model provides a means for spare parts and consumables to move from system to system in the simulation and makes maintenance services available to systems requiring repairs. The following sections discuss these components of the SMO Simulation.

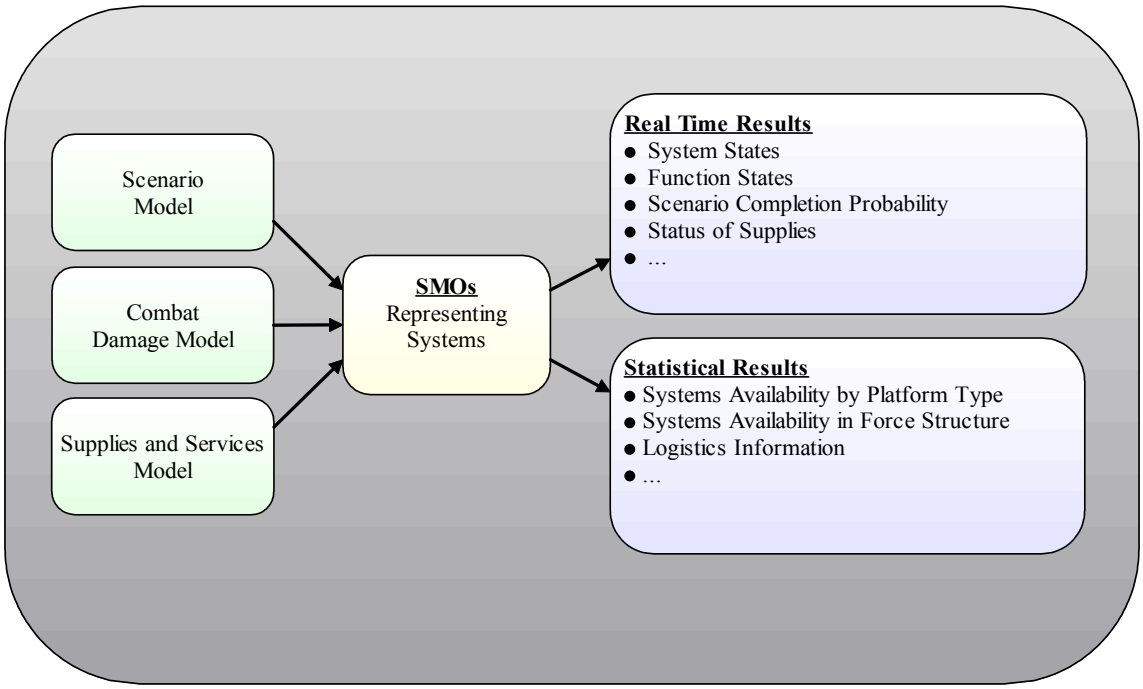

Figure 4.2 SMO Simulation Architecture

3.2 The State Model Object

The SMO can be configured to represent a wide variety of systems. Examples of the types of systems that might be represented by the SMO include air vehicles, ground vehicles, manufacturing equipment, a soldier and the equipment he carries, etc. For modeling and simulation purposes, the SMO can be used to represent almost any system whose functionality can be described by the states of the system's elements.

The SMO is made up of a collection of elements that may be subsystems, components, failure modes, external conditions, or functional elements of other systems. The SMO can have multiple functions such as mobility, communications, sensing, lethality, etc. Furthermore, any function can itself have multiple states and is not restricted to success or failure. The state of any function is determined by the states of the elements that contribute to that function. Elements and functions of the SMO are described in the next two sections.

3.2.1 Elements of the SMO

Elements of an SMO can be any one of the following four types:

- Primary Elements: These should be considered the elements that are subject to normal reliability processes such as failures and repairs. Primary elements might be components, field replaceable units, failure modes, etc.
- Consumables: A consumable can be any item that is used by a system during its operation. Examples might be fuel, ammunition, and water. Once a system is

assigned a consumable, the consumable becomes an element whose state becomes false when the consumable is used up.

- External Elements: These are elements outside a system that can affect a system's functionality. Examples of external elements might be a sandstorm or heavy forestation. External elements are defined as part of the scenario model and may then be identified as applicable to any system.
- Reference Elements: These are references to functions of another system that the current system may require for its functionality.

3.2.1.1 Primary Elements

Primary elements are elements that change through normal reliability processes as characterized by time-to-failure (TTF) or time-to-repair (TTR) distributions. Primary elements can identify spare parts and maintenance services required for their repair and are the means by which systems request and use parts and maintenance services.

Currently available time-to-failure distributions in the SMO Simulation are as follows:

1 Exponential: The only parameter needed for the exponential distribution is a failure rate which is assumed to be constant.

2 Weibull: The Weibull distribution is often used as a time-to-failure distribution. The version used in SMO Simulation requires three parameters;

- Shape: This parameter defines the shape of the distribution (dimensionless),
- Scale: This parameter determines the scale of the distribution (hours), and
- Location: This parameter locates the distribution on the time scale (hours).

3 Wearout. This distribution is a three-part distribution developed for use as a time-to-failure distribution. Its three parts are burn-in, normal life, and wear out. During the burn-in period, the failure rate is assumed to be linearly decreasing. During the normal life, a constant failure rate is assumed. In the wear out portion of the distribution, the time-to-failure distribution is assumed to be normal. The wear-out distribution requires five parameters as follows:

- Burn-In Fraction: This parameter determines the fraction of failures that occur during the burn-in period,
- Burn-In Duration: This parameter sets the duration of the burn-in period. Its units are hours,
- Random Fraction: This parameter sets the fraction of failures that are assumed to occur during the component's normal life,
- Mean: This is the mean of the normally-distributed end-of-life portion of the distribution. Its units are hours, and
- Standard Deviation: This is the standard deviation in hours of the normally-distributed end-of-life portion of the distribution.

Currently available time-to-repair distributions are as follows:

1 Fixed: This option simply specifies a fixed time-to-repair.

2 Normal: The normal distribution is defined by two parameters which are;

- Mean: This is the average value in hours for the time-to-repair and
- Standard Deviation: The standard deviation is a measure of the spread of the distribution (hours).

3 Lognormal: The lognormal distribution is defined by two parameters which are;

- Mean: This is the average value in hours for the time-to-repair and
- Standard Deviation: The standard deviation is a measure of the spread of the distribution (hours).

4 Uniform: The uniform time-to-repair distribution requires two parameters;

- Minimum: This is the lower bound of the range of time-to-repair values to be sampled and
- Maximum: This is the upper bound of the range of time-to-repair values to be sampled.

5 Triangular: The triangular time-to-repair distribution requires three parameters;

- Minimum: This is the lower bound of the range of time-to-repair values to be sampled,
- Most Likely: The most likely value must fall between the minimum and maximum values, and
- Maximum: This is the upper bound of the range of time-to-repair values to be sampled.

The time-to-repair distribution is intended to represent just the time to repair a failed element after any required parts and maintenance services become available. Delay times in acquiring a need part or maintenance service are treated through the supply network.

3.2.1.2 Consumable Elements

Any system may be assigned one or more consumables such as fuel, ammunition, etc. A consumable is defined by the following key properties:

- Capacity: This is the maximum amount the system can carry.
- Initial Quantity: This is the amount that the system carries at the start of the simulation.
- Reorder Level: This is the amount below which the system begins to request that the consumable be replenished.

- Usage Rate: This is the amount of the consumable used per hour of operation. This rate can be varied throughout the simulation scenario.
- Quantity: This is the current amount of the consumable as the simulation proceeds.

The state of the system element that represents a consumable is *True* so long as the quantity of the consumable is greater than zero. If the consumable is all used up before being replenished, the state of the corresponding element becomes *False*. Because the element that represents the consumable can be included in the success or failure equations for any system function, running out of a consumable can cause a system function to fail or degrade.

3.2.1.3 External Elements

External elements are defined for the simulation and may be assigned to as many systems as desired. Examples of external elements might be a sandstorm, rough terrain, heavy forestation, etc. Each external element has a desired state and an actual state. When the actual state is the same as the desired state, the element returns *True*. When the actual state is not the same as the desired state, the element returns *False*. As an example, suppose an external element called Sandstorm has been defined with *False* as its desired state. Suppose further that the Sandstorm element has been assigned to a group of systems and included in the failure equation for their visible imaging function. Then if the state of the Sandstorm external element becomes *True* (i.e., a sandstorm occurs), then all those systems with the Sandstorm element in their visible imaging functions would lose visual imaging or have their visual imaging function degraded.

3.2.1.4 Reference Elements

Reference elements are references to functions of another system that the current system may require for its functionality. For example, suppose a weapon system's targeting accuracy relies on laser target marking provided by an air vehicle. The weapon system can be given a reference to the air vehicle's target marking function. The reference element can then be included in the weapon's failure or success equations for targeting accuracy or lethality. If the air vehicle becomes inoperable, the targeting accuracy of the weapon system could be reduced.

3.2.2 SMO Functions

An SMO may have as many functions or measures of effectiveness (MOEs) as needed for the scenario to be simulated. The relationship between any SMO function and its contributing elements is characterized by a combination of failure and success equations. The failure equation is used to determine whether a function is available at all whereas success equations determine whether a function is partially available and at what level. The SMO characterizes a system's overall operability and as many functions of the system as desired. For system operability and each system function, the SMO provides the following information:

- Real-time status of any function (available, partially available, unavailable),

- The probability of maintaining the function for a specified future time interval,
- The most likely problem areas for the function (elements most likely to cause loss of function), and
- Detailed statistics such as the cumulative time in each state, number of state changes, etc.

Failure and success equations for a system can be derived from a state model as discussed in Section 2 or from one or more fault trees. In its current form, the user must directly input the desired equations when setting up a multi-system simulation. As development of the tools reported here becomes more mature, failure and success equations will be imported directly into the SMO Simulation from the state model or fault tree solvers.

To illustrate the development and use of failure and success equations, a simple weapons system is examined that consists of an M240 Machine Gun, M242 Chain Gun, and an Electrical System that provides needed electrical power. Three goal states are of intereest: 1) no lethality, 2) partial lethality with only the M240 Machine Gun operable, and 3) partial lethality with only the M242 Chain Gun operable. The state model is shown in Figure 4.1. Leaf states are shown in green and transitions are labeled as X.

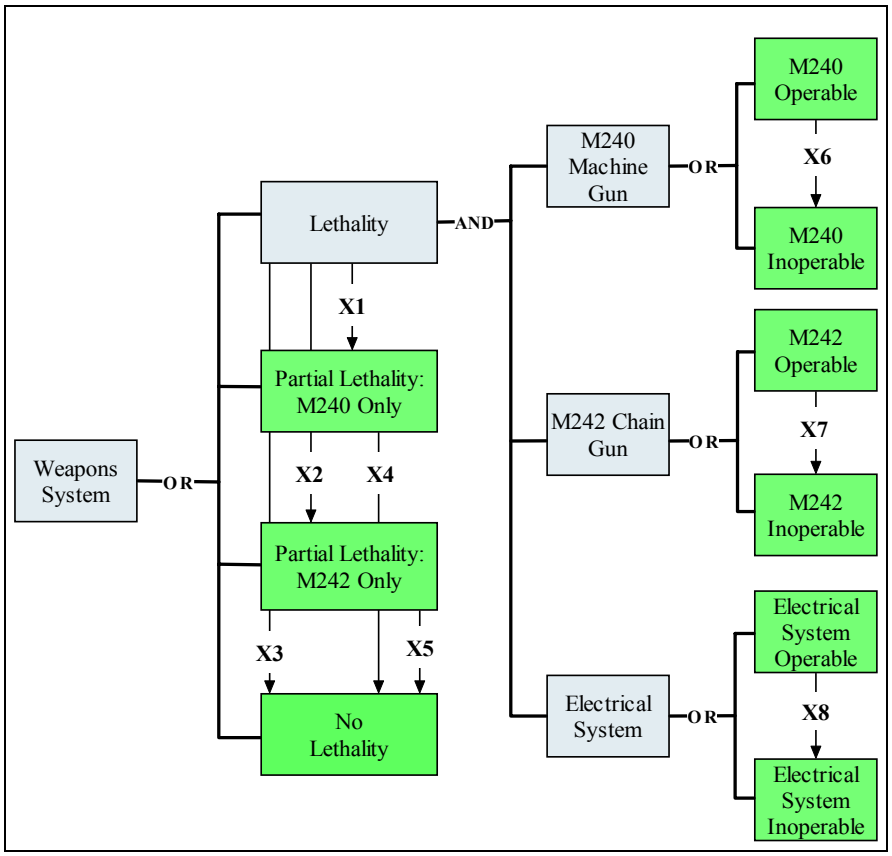

Figure 4.1 State Model for Simple Weapons System

The transitions shown in Figure 4.1 are defined as follows:

X1 = X1(G1): Transition from ***Lethality*** to ***Partial Lethality: M240 Only***.

X2 = X2(G2): Transition from ***Lethality*** to ***Partial Lethality: M242 Only***.

X3 = X3(G3): Transition from ***Lethality*** to ***No Lethality***.

X4 = X4(G4): Transition from ***Partial Lethality: M240 Only*** to ***No Lethality***.

X5 = X5(G5): Transition from ***Partial Lethality: M242 Only*** to ***No Lethality***.

X6 = X6(T6): Transition from ***M240 Operable*** to ***M240 Inoperable***.

X7 = X7(T7): Transition from ***M242 Operable*** to ***M242 Inoperable***.

X8 = X8(T8): Transition from ***Electrical System Operable*** to ***Electrical System Inoperable***.

The guard expressions G1 to G5 are defined as follows:

G1 = (***Electrical System Operable***) $\cap$ (***M240 Operable***) $\cap$ (***M242 Inoperable***)

G2 = (***Electrical System Operable***) $\cap$ (***M240 Inoperable***) $\cap$ (***M242 Operable***)

G3 = (***Electrical System Inoperable***) $\cup$ ((***M240 Inoperable***) $\cap$ (***M242 Inoperable***))

G4 = (***Electrical System Inoperable***) $\cup$ (***M240 Inoperable***)

G5 = (***Electrical System Inoperable***) $\cup$ (***M242 Inoperable***)

The triggers T6 to T8 are defined as follows:

T6 = ***M240-Fail*** (M240 Machine Gun fails).

T7 = ***M242-Fail*** (M242 Chain Gun Fails).

T8 = ***Elec-Fail*** (Electrical System Fails).

To address how the state model arrives at the No Lethality state, a Boolean equation for arriving at this state is built based on the initial conditions. It is assumed that each component is initially in its operable state. Thus, M240 Operable, M242 Operable, and Electrical System Operable are initial states. By extension, this implies that the parent states Lethality and Weapons System are also initial states.

In the context of the state model Boolean equation, if a state is an initial state it is assigned a value of true because it can be reached unconditionally. In other words, nothing has to occur to reach the state because the system starts out in the state.

The construction of the Boolean equation for the No Lethality state follows the procedures outlined in section 3.3. The starting point is the No Lethality state and the first step is to find all transitions that lead to the state. Hence,

No Lethality = X3 ∪ X4 ∪ X5.

A transition is true only if its source state is occupied and its guard and trigger are true. None of these three transitions has a trigger so this reduces to source state is occupied and its guard is true. Expanding these conditions for transition X3:

X3 = ***Lethality*** ∩ G3

X3 = True ∩ G3

X3 = G3

X3 = ***Electrical System Inoperable*** ∪ (***M240 Inoperable*** ∩ ***M242 Inoperable***)

Expanding for transitions X4 and X5:

X4 = ***Partial Lethality: M240 Only*** ∩ G4

X4 = ***Partial Lethality: M240 Only*** ∩ (***Electrical System Inoperable*** ∪ ***M240 Inoperable***)

X5 = ***Partial Lethality: M242 Only*** ∩ G5

X5 = ***Partial Lethality: M242 Only*** ∩ (***Electrical System Inoperable*** ∪ ***M242 Inoperable***)

To further expand transitions X4 and X5 we need to determine how the state model reached their source states. So for both of these two states, find the transitions that point to them as destination states. In each case there is only one such transition. Hence the source state can be replaced with the appropriate transition. Rewriting then,

X4 = X1 ∩ (***Electrical System Inoperable*** ∪ ***M240 Inoperable***)

X5 = X2 ∩ (***Electrical System Inoperable*** ∪ ***M242 Inoperable***)

The next step is to expand transitions X1 and X2. Because each only has a guard and because the source state in each case (***Lethality***) is an initial state, the equations for the transitions reduce to those of the guards:

X1 = G1

X1 = ***Electrical System Operable*** ∩ ***M240 Operable*** ∩ ***M242 Inoperable***

X1 = ***M242 Inoperable***

X2 = G2

X2 = ***Electrical System Operable*** ∩ ***M240 Inoperable*** ∩ ***M242 Operable***

X2 = ***M240 Inoperable***

Here we have again replaced the initial (operable) states with a value of true. Substitute X1 back into the equation for X4 and X2 into X5. Use the distributive law to rewrite the equations in disjunctive form as shown.

X4 = ***M242 Inoperable*** ∩ (***Electrical System Inoperable*** ∪ ***M240 Inoperable***)

X4 = (***M242 Inoperable*** ∩ ***Electrical System Inoperable***) ∪

(***M242 Inoperable*** ∩ ***M240 Inoperable***)

X5 = ***M240 Inoperable*** ∩ (***Electrical System Inoperable*** ∪ ***M242 Inoperable***)

X5 = (***M240 Inoperable*** ∩ ***Electrical System Inoperable***) ∪

(***M240 Inoperable*** ∩ ***M242 Inoperable***)

Recall that X3, X4, and X5 are combined under the Boolean OR operator. The intersection term (***M240 Inoperable*** ∩ ***M242 Inoperable***) appears in the expressions for all three. Hence using the idempotent law, it only appears once when the union taken and is simplified. Also, the law of absorption is used twice. That is for example, by adsorption

Electrical System Inoperable ∪ (***M240 Inoperable*** ∩ ***Electrical System Inoperable***)

is equivalent to

Electrical System Inoperable

Accounting for the steps taken thus far we have:

No Lethality = ***Electrical System Inoperable*** ∪ (***M240 Inoperable*** ∩ ***M242 Inoperable***)

It remains to address how the state model reached the three states shown in the latest equation. Each one can be arrived at via a single transition. So again, replace the state with the appropriate transition.

No Lethality = X8 ∪ (X6 ∩ X7)

A transition is true only if its source state is occupied and its guard and trigger are true. None of these three transitions has a guard so this reduces to source state is occupied and its trigger is true. Also, for each of these three transitions the source state is an initial state, hence true, so each transition can be replaced by its trigger.

No Lethality = T8 ∪ (T6 ∩ T7)

Even though in general a trigger can be a Boolean expression of events, each of these triggers has but a single event. Making the final substitution:

No Lethality = ***Elec-Fail*** ∪ (***M240-Fail*** ∩ ***M242-Fail***) (4.1)

Thus, lethality is lost if the electricity fails or both of the guns fail. This expression is already in disjunctive form which is appropriate for defining the cutsets that constitute a failure equation for the lethality function in an SMO. Using the terminology of the user-interface for SoS input (Appendix B), there is a Simple (or Series) cutset ***Elec-Fail*** and an And (or Parallel) cutset {***M240-Fail***, ***M242-Fail***}. If either of these cutsets occurs, the lethality function fails.

Since there is redundancy in the failure equation, the question arises whether there might be intermediate levels of lethality – not just full lethality or no lethality. For example, suppose the M240 fails but the M242 remains operable as does the electrical system. To find all the ways the lethality function can be successful, we will apply negation to the failure equation. First, it is convenient to rewrite the failure equation (4.1) as follows:

$$\overline{\text{Lethality}} = \overline{\text{Elec}} \cup \left(\overline{\text{M240}} \cap \overline{\text{M242}}\right) \quad (4.2)$$

The rewritten failure equation is read as No Lethality = No Electrical System or (No M240 and No M242) where No Electrical System implies electrical system failure, etc. We will now apply negation to the above equation to get

$$\overline{\overline{\text{Lethality}}} = \overline{\overline{\text{Elec}} \cup \left(\overline{\text{M240}} \cap \overline{\text{M242}}\right)} \quad (4.3)$$

Now by De Morgan's theorem,

$$\overline{\text{A} \cup \text{B}} = \overline{\text{A}} \cap \overline{\text{B}}$$

and

$$\overline{\text{A} \cap \text{B}} = \overline{\text{A}} \cup \overline{\text{B}}$$

Applying De Morgan's theorem to the negated lethality equation we get

$$\text{Lethality} = \overline{\overline{\text{Elec}}} \cap \left(\overline{\overline{\text{M240}}} \cup \overline{\overline{\text{M242}}}\right)$$

Using the distributive law and noting the double negation,

$$\text{Lethality} = \left(\text{Elec} \cap \text{M240}\right) \cup \left(\text{Elec} \cap \text{M242}\right) \quad (4.4)$$

Or lethality succeeds if both the electrical system and the M240 succeed or both the electrical system and the M242 succeed. When the success equation is written in disjunctive form as above, one can examine the terms in parenthesis and consider whether system performance differs depending on which success term is true. In this example we can see that the two intermediate levels of lethality, partial lethality with the M242 operable and partial lethality with the M240 operable, are represented by the terms in parenthesis. The two success equations for these intermediate states are

Partial Lethality: M240 Only = Elec $\cap$ M240

Partial Lethality: M242 Only = Elec $\cap$ M242

Notice that the series term from the failure equation 4.1 (actually it's negative), appears in both success equations. In general one would expect that all series terms in the failure equation would appear in every success equation. In other words, if a system fails by any of its series elements, there is no need to evaluate success equations since they will all evaluate to *false*.

3.3 The Scenario Model

Scenarios allow the analyst to provide detailed specifications for the location of a system, its expected performance, and any external conditions that might affect its behavior. The analyst can define as many scenarios as desired. Further, multiple systems can follow the same scenario or, if desired, every system can follow a different scenario.

3.3.1 Scenario Segments

A scenario is made up of time intervals or segments and a scenario can have as many segments as desired. For each time segment, the following information is required:

- Duration: This is the length of the time segment in hours.
- Location: The location is an enumerated property with three values: 1) field, 2) repair facility, and 3) other. These locations help determine such things as where a failed system can be repaired. For example, some failures may be repairable in the field while others might only be repairable when the system is at a repair facility.
- Desired System State: This is the desired state of systems that follow this scenario. Desired system states would usually be either operating (the system is expected to operate) or operable (the system is not expected to operate but should be capable of operating) during the segment.
- External Condition: External conditions (discussed below) provide a mechanism for modifying system properties. For example, an external condition such as air turbulence might increase the probability of failure or aging rate of aircraft elements that would undergo additional stress in turbulent conditions.
- Condition Probability: This is the probability that a specified external condition will actually occur during the scenario segment.

3.3.2 External Conditions

External conditions are conditions within the simulation that are external to any of the systems but may affect the properties or performance of one or more systems. An external condition may affect a system directly or affect the elements of the system. Specifically, at the system level, external conditions can have the following effects:

- Combat Damage Change: A combat damage definition can be applied to a system to enable combat damage to occur or to change the type of combat damage that occurs.
- Combat Damate Rate: The rate or probability per unit time of combat damage can be modified by applying an external condition to a scenario segment.

For primary elements, external conditions can have the following effects:

- Failure Rate: The failure rate of any primary element having an exponential time-to-failure distribution to be increased or decreased from its baseline input value.
- Aging Rate: The aging rate of any primary element of any system can be increased or decreased from its baseline input value.
- Repair Time: The repair time of any primary element can be modified.
- Weibull Location Parameter: For primary elements having a Weibull time-to-failure distribution, the location parameter can be modified. This effectively reduces or increases the expected life of the element.
- Wearout Mean Life: For primary elements having a wearout distribution, the mean of the end-of-life portion of the distribution can be modified to increase or decrease the expected life of the element.
- Wearout Random Fraction: For primary elements having a wearout distribution, the probability of a random failure during the element's normal life can be modified.

Other possible effects of external conditions include a change in the rate of consumable usage and changing the state of an external element. Planned development in this area of the simulation include allowing external conditions to modify the delay times associated with delivering spares, consumables, and maintenance services.

3.4 The Combat Damage Model

Combat damage is represented by a model that is similar to a decision tree. A simple example is shown in Figure 4.2.

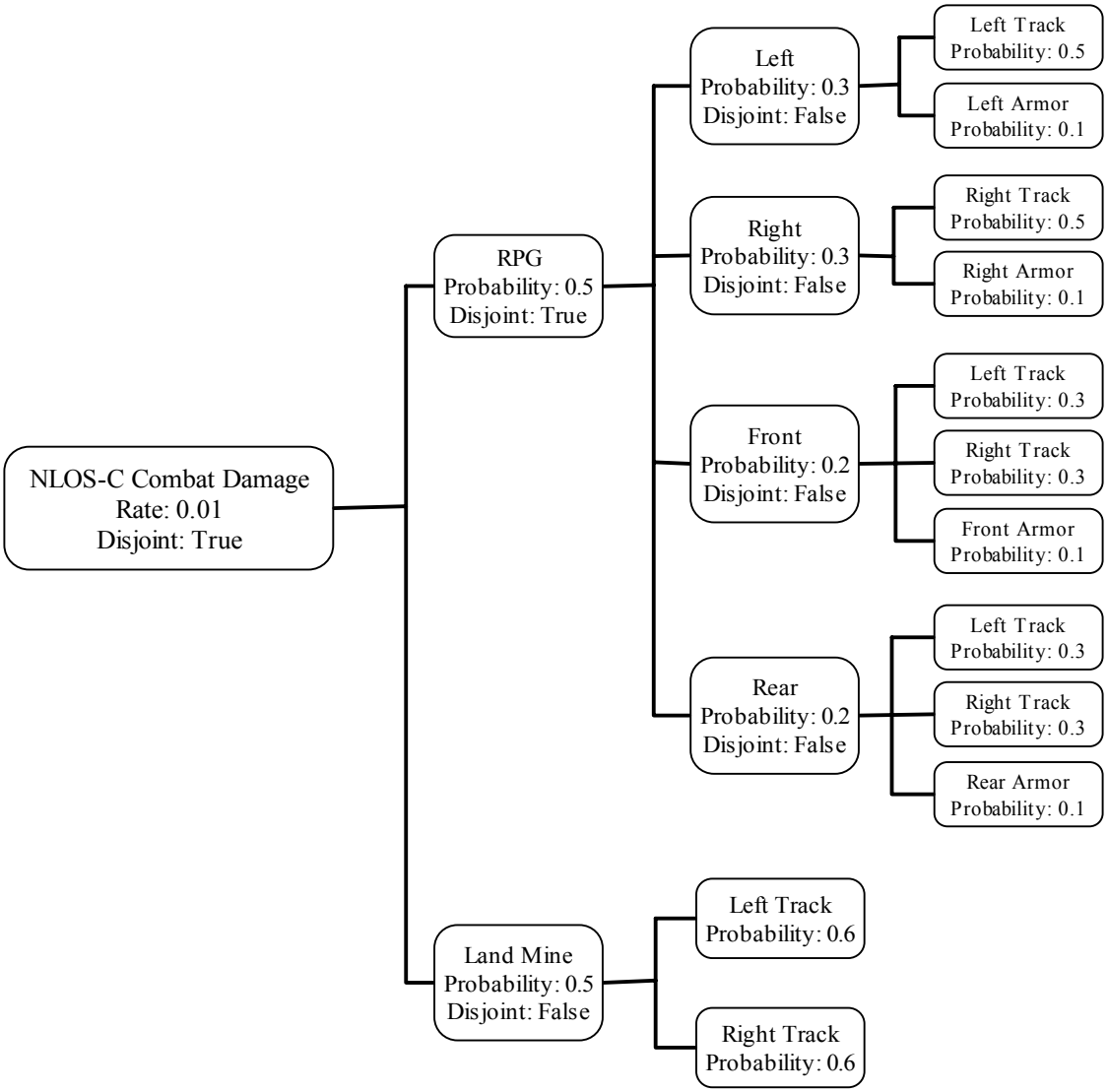

Figure 4.2 Example of a Combat Damage Tree

For the root node (NLOS-C Combat Damage in Figure 4.2), the required information is the combat damage rate (probability per hour of combat damage) and an indication of whether the child nodes are disjoint. If the child nodes are disjoint, only one child node can occur. In this case, the probabilities of the disjoint child nodes must add to 1. If the child nodes are not disjoint, the probabilities of the child nodes are not required to add to 1. For nodes other than the root node, the required information is the conditional probability of reaching that node (given that its parent has occurred), and whether its children are disjoint.

With the above explanation, we can interpret Figure 4.2 as follows:

- The probability per hour of combat damage for the NLOS cannon is 0.1.

- If the NLOS-C experiences combat damage, it will come from either an RPG (50% chance) or a land mine (50% chance).
- If the damage comes from an RPG, the damage will be to the left (30% chance), right side (30% chance), front (20% chance), or rear (20% chance) of the platform.
- If the RPG hits the left side of the NLOS-C, there is a 50% chance that the left track will be damaged and a 10% chance the left armor will be damaged.

The tree can go to as many levels as necessary or can be a single node. Not shown in Figure 4.2 is the kill probability which can be applied at any level of the tree.

3.5 The Supplies and Services Model

This section discusses the approach for modeling the movement and use of supplies and services in the SMO Simulation. The supplies and services model in SMO Simulation is designed to meet the following needs:

- Orders for spares needed for repairs,
- Orders for parts to replenish inventories,
- Orders for consumables by users,
- Orders for consumables to replenish supplies, and
- Orders for service

The process by which these needs are met during a simulation is illustrated in Figure 4.3. The need for a supply or service is identified when a part or service is required to repair a primary element, when a consumable is running low, or when a supplier of parts or consumables needs to replenish their inventory. When the need for a supply or service is identified, an order is created that identifies the customer (the SMO that needs the supply or service). If the order is for parts or consumables, they are identified along with the quantity needed. If the order is for a service, the required service is identified.

Any system in the simulation can be a user or supplier of parts or consumables. Similarly, any system can be a user or provider of maintenance services. To manage orders for supplies and services, every system maintains collections of active orders for parts, consumables, and services. When a system requires a supply or service, an order is created and placed in the customer system's collection of active orders. Then a supplier or provider is sought using the supply connections established for the customer system. When the best supplier or provider is found, the order is added to its active orders. The best supplier or provider is the one with the best priority who can provide the supply or service in the shortest time. There is a delay time associated with every delivery of parts, consumables, or services. The delay time is a random variable whose value depends on the supply connection that is used. The delay time count down for parts begins as soon as the order is accepted by a supplier. Orders for consumables and services are placed in a queue and delivered by the supplier or provider in the order that they are received. The

exception to this rule is when a system self-supplies a consumable or self-provides a service in which case the order is placed at the front of the queue.

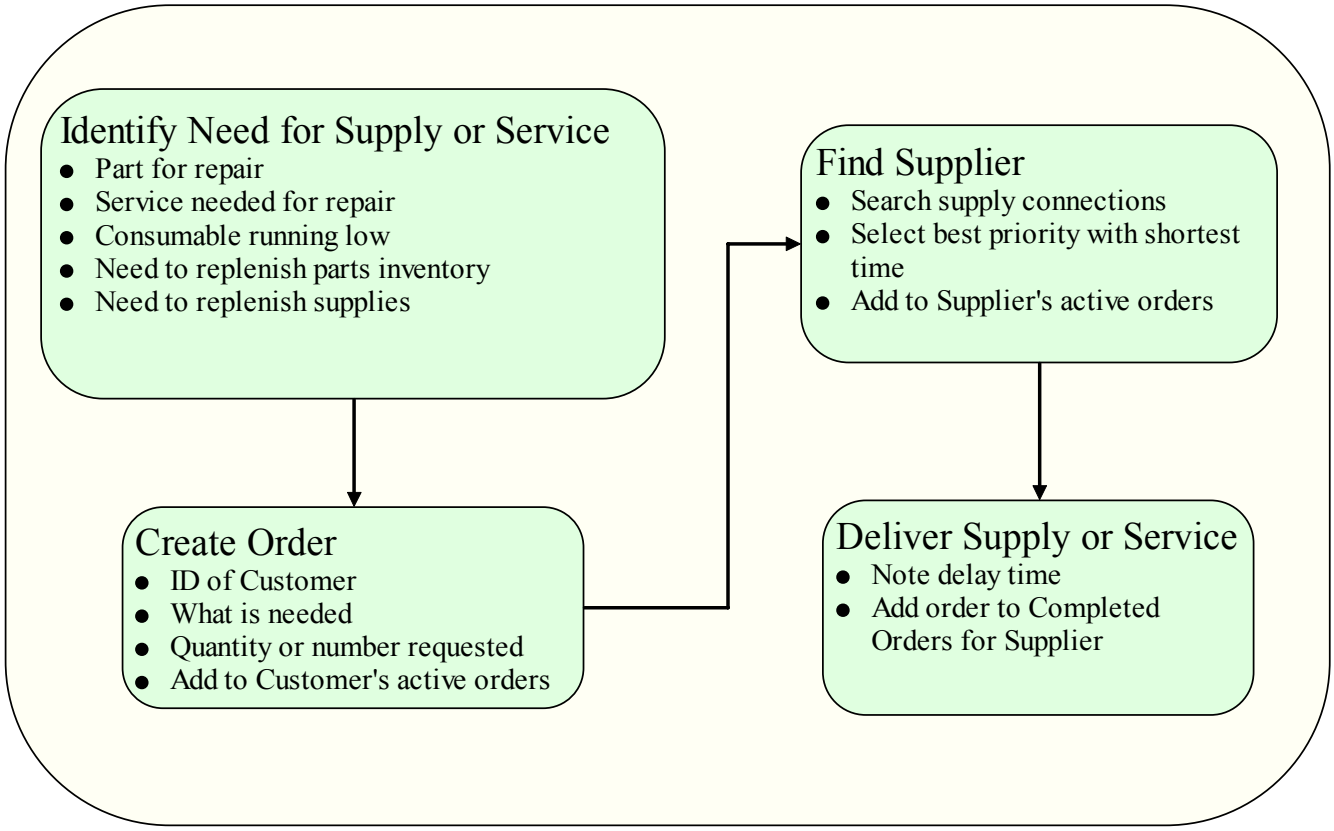

Figure 4.3 Process for Using the Supply Network

3.5.1 Spare Parts

The treatment of spare parts begins with a collection of all parts to be used in the simulation. Spares are characterized by the following properties:

- ID: This property identifies the part.
- Cost: The cost of the part.
- Weight: The weight of the part in consistent units.
- Volume: The volume of the part in consistent units.

Any carrier or storage area for spare parts will be a system as defined by the State Model Object (SMO). To assign spares to be carried by a system, we first create spares kits or inventories. For every part in the inventory, the following information is needed:

- Desired Number: This is the number of a particular part that should be in the inventory.

- Actual Number: This the actual number of the part in the inventory at any time.
- Reorder Level: This is the level at which an order is placed to take the inventory of a particular part back up to desired number.
- Lot Size: Orders for replacement parts are made in multiples of lot size.

The user may create as many different parts inventories as desired and assign them to the various systems in the simulation. If desired, evey system in the simulation can carry a different spares kit or inventory.

3.5.2 Consumables

Any supplier or storage area for consumables will be a system as defined by the State Model Object (SMO). Similarly any system (SMO) may be a user of one or more consumables.

Consumables are defined by the following properties:

- ID: This property uniquely identifies the consumable.
- Units: This property specifies the units in which the consumable will be used and replenished. Examples are gallons of fuel or rounds of ammunition.
- Weight per unit: The weight per unit will allow weights of consumables to be calculated. Weight values should be in internally consistent units.
- Volume per unit: The volume per unit will allow consumable volumes to be calculated. Volume values should be in consistent units.
- Cost per unit: The cost per unit allows consumable costs to be calculated.

Consumables as they are used by a system are defined by the following properties:

- ID: This string uniquely identifies the consumable and must match a consumable definition.
- Initial Quantity: This is the amount of the consumable at the beginning of the simulation.
- Quantity: This is the current quantity as the simulation proceeds.
- Capacity: This is the maximum amount of the consumable that the system can carry.
- Request Level: This is the level below which the system requests replenishment.
- Order Quantity: This is equivalent to lot size for spares. For example, if water comes in 5 gallon containers and water volume is measured in gallons, this value should be 5.
- Usage Rate: This quantity defines the baseline rate at which the consumable is used when the system uses it. The usage rate can be varied throughout the simulation if desired.

Supplies (consumables provided by supplier systems to end users) have the same properties as consumables except for the usage rate.

Consumables used by a system become elements of the system and as elements, they can be included in the failure or success equations for any system function. An example might be fuel which could be included in the failure equation for the mobility function. The state of the element represented by a consumable remains true so long as its quantity is greater than 0. So, continuing the fuel example, the *fuel* element would transition to the *false* state when the system runs out of fuel. If fuel were included as a series element in the failure equation for mobility, then mobility would be lost when the fuel was all used up.

3.5.3 Maintenance Resources

Maintenance resources are services that may be required to repair a failed primary element. Such services may be provided by the crew of the system being repaired or may be provided by another system that is equipped for the service. The service may be the actual element repair in which case the time to perform the service is the repair time. Or, a service may required in order to allow the repair to take place; e.g., towing. In this case, the service may require time to perform that is not part of the normal element repair time. In any case, there is a delay time required to access the service, even if it is provided by the crew. When a service provider takes multiple requests for service, each service order in the queue will have an access delay time. So, for the second user in the queue, the delay time will be the sum of the time required for the first user to access the service, the time required to perform the service for the first user, and the time required for the second user to access the service.

The simulation treats three types of services as follows:

1. Basic service: This is a basic service such as a track repair, an engine repair, a tow, etc. Basic services should be defined at a level of granularity that is appropriate to the details of the simulation.
2. User service: A user service consists of one or more basic services that a system may require in order to repair a failed primary element. An example of a user service might be just a basic service such as track repair or an ordered list of basic services such as towing followed by engine repair.
3. Provider services: Provider services are collections of basic services that are available from service providers. Provider services parallel spares inventories. If a particular system can provide more than one basic service, these basic services must be grouped together into a provider service so that the corresponding collection of basic services can be assigned to the provider system.

Provider services are assigned, much like spares inventories, to systems that will provide the various basic services to other systems. For example, all the basic services that the crew of NLOS cannons can perform should be grouped together into a provider service and assigned to NLOS cannons.

Primary elements of every system can identify a user service or basic service that is required to repair the element when it fails. For example, an element representing the transmission of a manned ground vehicle might identify a required user service that includes a tow followed by a transmission repair.

Services are made available to systems needing services through the SupplyConnections class. SupplyConnections are discussed below in section 4.5.4.

3.5.4 Supply Connections

A Supply connection establishes a link between customers or users of supplies or services and suppliers or providers of these supplies and services. For example, a self-supply connection might be established to allow a fuel truck to refill its own fuel tank when necessary. Similarly, a connection might be established with all manned ground vehicles in a platoon as customers of a single field repair team in the same platoon. A supply connection can tie any group of customers or users to any group of suppliers or providers and can be used for any or all of the following purposes:

- Provide a spare needed for a repair.
- Replenish the spares inventory for a supplier of spares.
- Supply a consumable for an end user of the consumable
- Replenish the consumables supply for a consumable supplier.
- Access a maintenance service needed to repair a system.

Key properties of the supply connection are as follows:

- ID: This string uniquely identifies the connection.
- Connection Uses: This is a bitwise summed value that indicates which of the above purposes the connection can be used for.
- Delay Time: This distribution characterizes the uncertainty in the time required to acquire a supply or service using the connection. There is a different distribution for each use of the connection.
- Users: This is a collection of all system names that can use the connection for the specified purposes.
- Suppliers: This is a collection of all systems that are suppliers/providers through this connection.

When a system identifies a need for a part, supply or service, it examines the supply connections that are available to it as a user to see if any of the connections can be used for the desired purpose. For example, if a primary element fails and a part is needed for its repair, the system looks through its supply connections for a connection that can be used to acquire a spare for a repair. If it finds such a connection, it then searches through all the suppliers specified by the connection to see if any of them have the required spare. If there are multiple potential suppliers, the one having the combination of best priority and shortest delivery time is chosen.

3.6 Results for 99 System Example

Because of its size and complexity, the example problem will not be presented in detail. The construction of an input file is described in Appendix B and, for this example problem, the input file contains thousands of lines of set-up data. Instead, the discussion will focus on the kinds of results that are currently available.

The example problem models a battalion with a total of 99 systems with the types and numbers shown in Table 4.1. The force structure setup consists of a battalion with two companies and each company has two platoons. The distribution of the 99 systems within the force structure is shown in Table 4.2.

Table 4.1 Systems in the Example Problem

System Type	Number
Command and Control (C2V)	7
Fuel Truck (FT)	8
Infantry Carrier Vehicle (ICV)	16
Non Line-of-Sight Cannon (NLOS-C)	16
Reconnaissance and Surveillance Vehicle (RSV)	16
Unmanned Air Vehicle (UAV)	32
Parts Truck (PT)	2
Parts Depot (PD)	1
Forward Spotter (FS)	1

Table 4.2 Distribution of Systems in the Force Structure

	C2V	FT	ICV	NLOS-C	RSV	UAV	PT	PD	FS
Battalion	1							1	
Company A	1	4					1		
Platoon 1	1		4	8	4				1
Platoon 2	1		4		4	16			
Company B	1	4					1		
Platoon 1	1		4	8	4				
Platoon 2	1		4		4	16			

The manned ground vehicles and the spotter all follow a scenario that consists of three segments – 72 hours of activity followed by a 24 hour replenishment interval followed by a final 72 hours of activity. The UAVs have 2 hour flights every eight hours for the two 72 hour intervals of activity with a 24 hour replenishment interval in the middle. The parts depot has a single interval of 168 hours of activity.

All the manned ground vehicles have the following 5 functions:

- Operability: This function includes all the capabilities considered necessary for the mission.
- Command and control: This function includes communications, computing, network and other control functions as appropriate to the platform.
- Sensing: This includes all sensing capabilities available on the platform.
- Mobility: This function includes every system element required for the platform to be able to move.
- Lethality: This function takes into account any weapons on the platform and their control systems.

The UAVs have all the functions except lethality since they are assumed to have no weapons.

The only consumable in the simulation is fuel which is used by all the manned ground vehicles. The next several figures illustrate the outputs that are currently available.

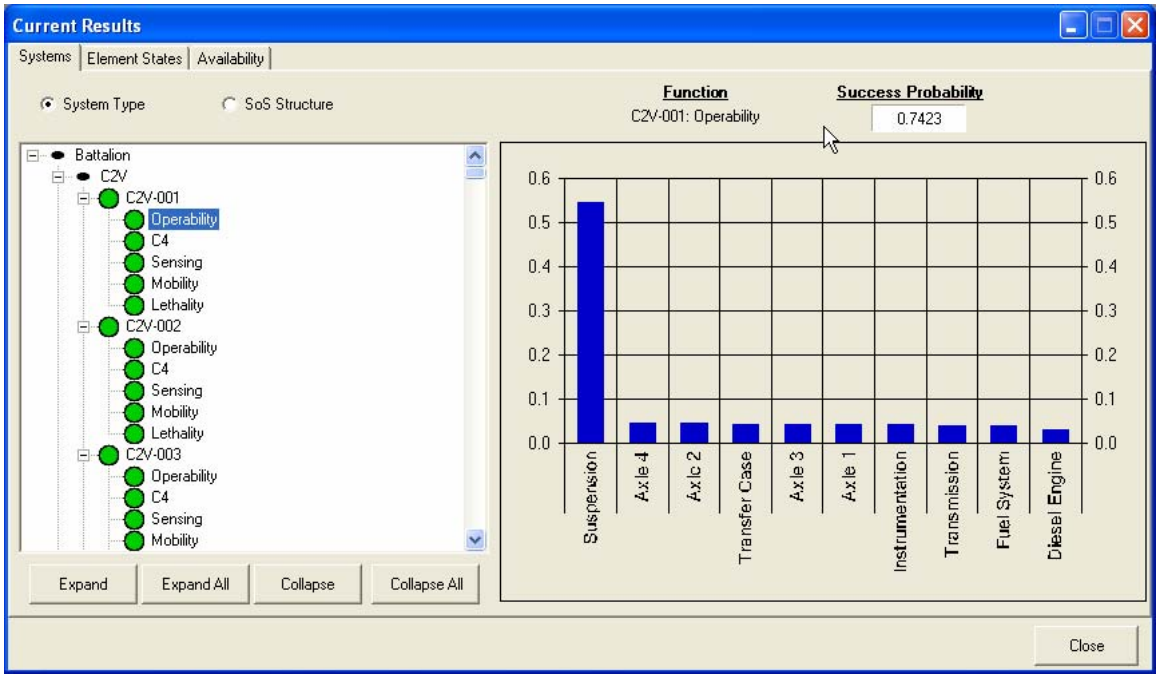

Figure 4.4 Functions by System

Figure 4.4 shows systems and their functions in real time as the simulation proceeds. The tree structure on the left side can be shown with the systems organized by system type as is the case in Figure 4.4 or the systems can be organized within the system-of-systems structure. All functions are shown for each system. When a system function fails, the green circle beside that function turns red. If the function is reduced to an intermediate state (partially operable), the circle turns yellow. The right side of the form shows the probability of successful operation of the selected function for the remainder of the mission. In Figure 4.4, we can see that the probability that C2V-001 will remain operable for the remainder of the mission is 0.74. Should C2V-001 lose operability, the most likely causes are shown in the Pareto chart.

Figure 4.5 shows the same output screen where NLOS-C-010 has become inoperable. The results on the right side of the form are for the C2 function of the NLOS-C-010.

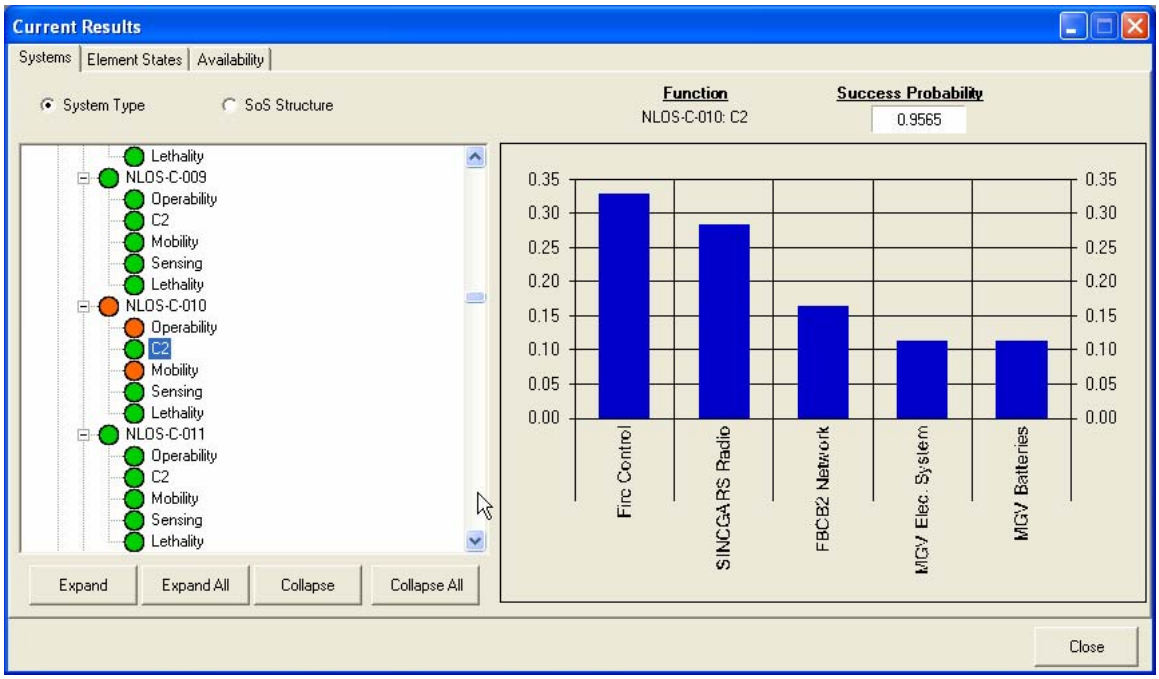

Figure 4.5 Loss of Operability for NLOS-C-010

The next form (Figure 4.6) shows the states of elements of a selected system. On the left side of the form is a list of all the systems in the simulation. The right side shows elements of the selected system. We can see that NLOS-C-010 has been selected. The grid on the right of the form shows that the alternator has failed (column 2) which has led to the loss of mobility and operability of the system. The time the element has spent in its current state is shown in the third column and the fourth column shows the length of time the element is expected to spend in that state. In the case of the alternator, we don't know how long it will remain failed because the failure cannot be repaired until the replenishment interval which has not been reached by the simulation.

The bottom right of the form shows consumables in use by the system. We can see that NLOS-C-010 had over 49 gallons of fuel left when the failure occurred. The projected time remaining is infinite since the system has failed and is not using fuel.

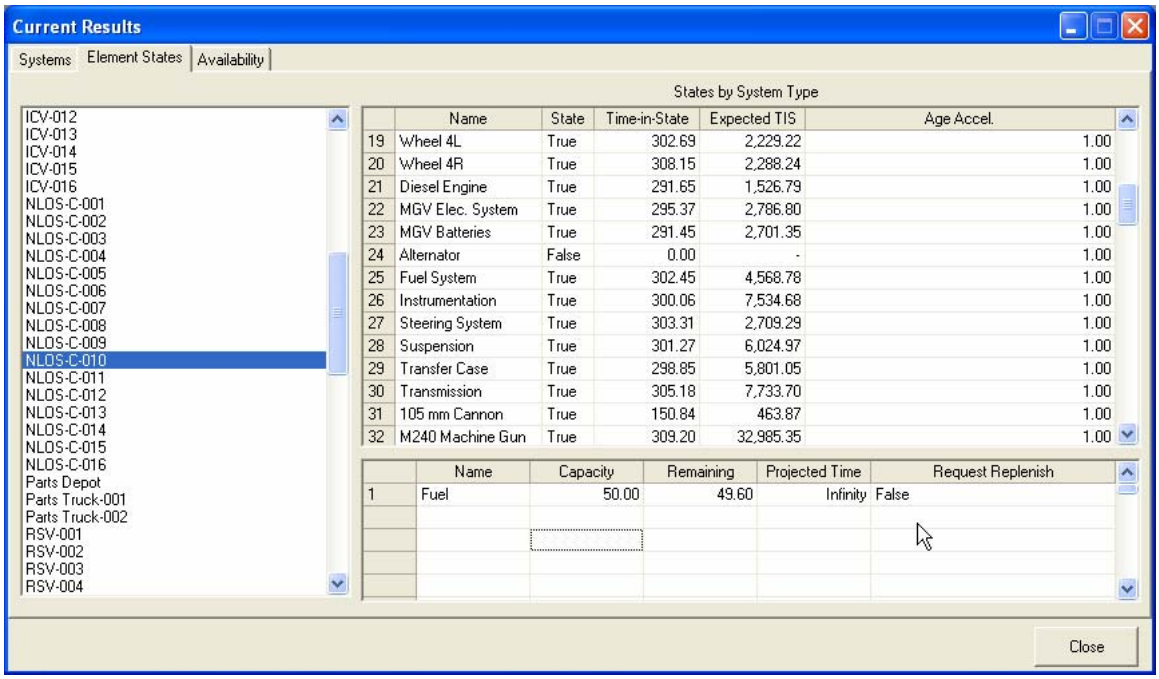

Figure 4.6 Elements of NLOS-C-010

The last of the real-time output forms is shown in Figure 4.7. In this case, the form shows instantaneous availability of the different platform types. The instantaneous availability is defined as the fraction of the systems of a particular type that are operating or operable at the time. The left side of the form lists the different system types plus a category called *All* at the top of the list. When *All* is selected, the grid on the right of the form shows the total number of systems of each type, the number operating or operable, the number inoperable, and the fraction of systems of that type that are operating or operable. When a particular type of system is selected in the list on the left of the form, the grid on the right side shows the number of systems for which each function is operable (green), partially operable (yellow) or inoperable (red) (Figure 4.8). It also shows the fraction of systems of the given type that are operable (green or yellow).

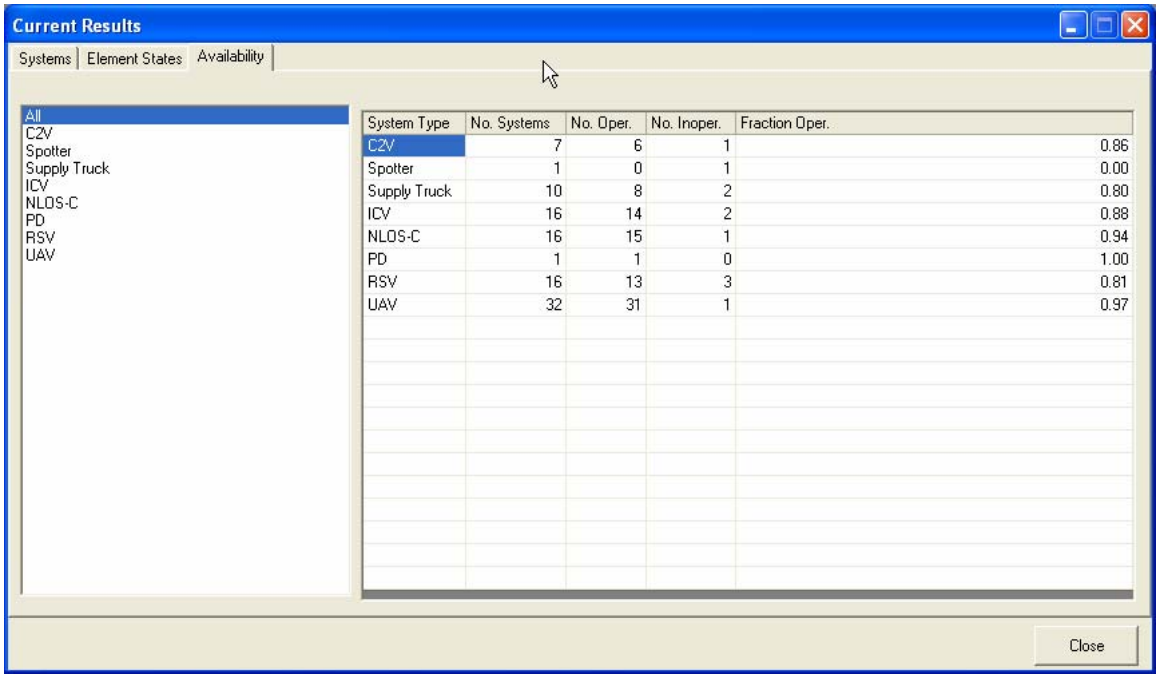

Figure 4.7 Instantaneous Availability by System Type

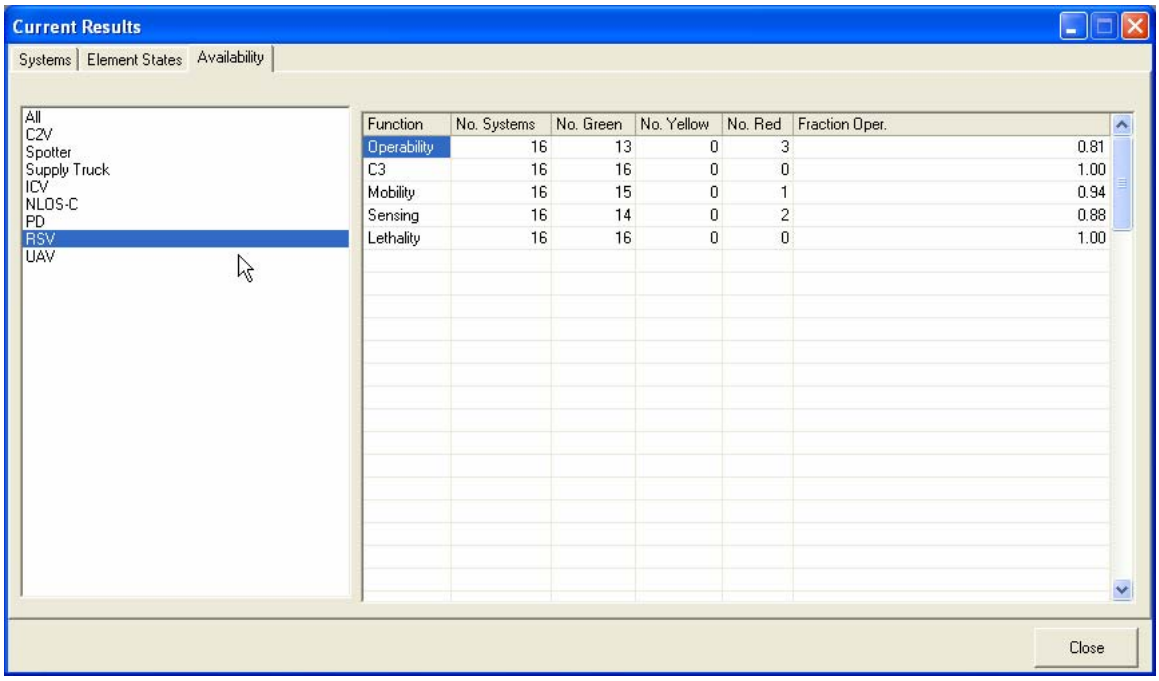

Figure 4.8 Instantaneous Function Availability for RSVs

The remaining figures in this section show statistical results obtained from running the simulation through several replications; in this case 25 replications were used. Figure 4.9 shows MTBF by system type. The three values in parenthesis represent the 5th percentile, mean, and 95th percentile. For example, for the 32 UAVs in the simulation, the mean MTBF was 14.9 hours with the 5th and 95th percentiles being 12.81 and 16.54 hours. Results for the NLOS-C show that the 5th percentile is >144. This means that the results, when sorted by simulation, show that at the 5th percentile no NLOS-C failed during the entire 144 hours it was expected to operate. The simple average option for the MTBF calculation finds the average MTBF for each system under a node then finds the average of those averages as the mean value for the node. If the global average option is selected, the mean MTBF for a node is determined by finding the total operating hours for all systems under a node and dividing by the total number of failures for all systems under the node.

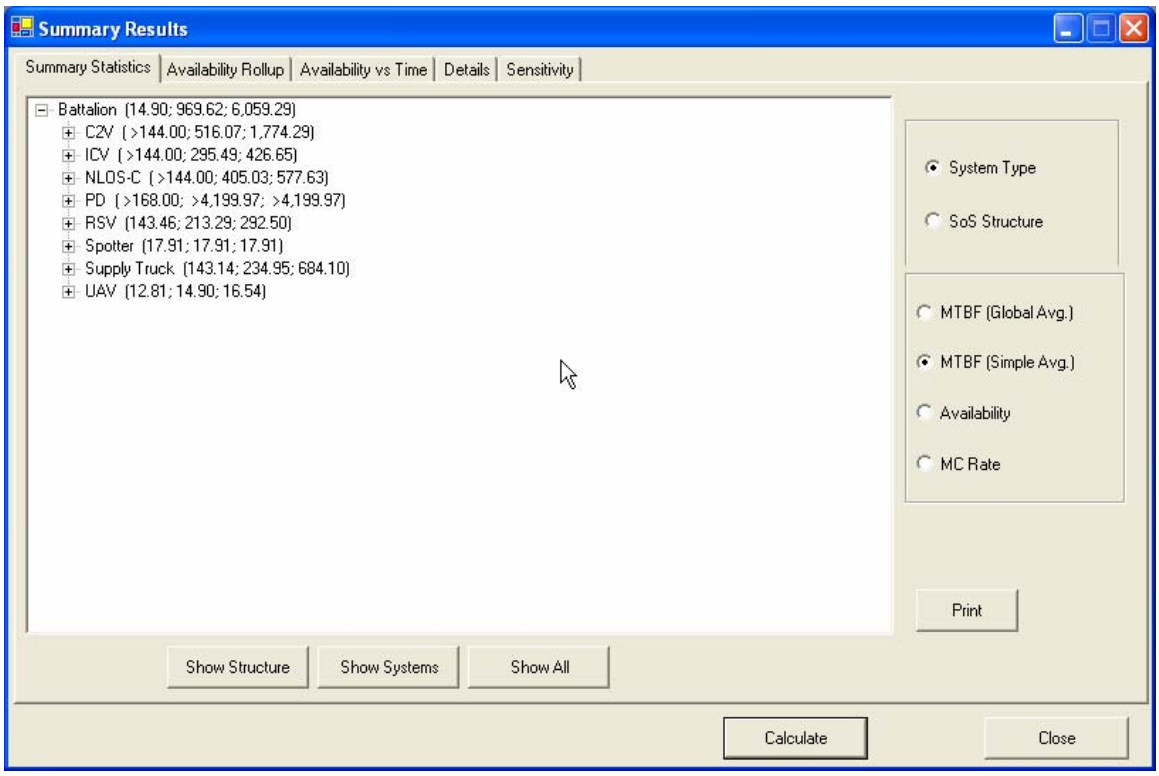

Figure 4.9 MTBF by System Type

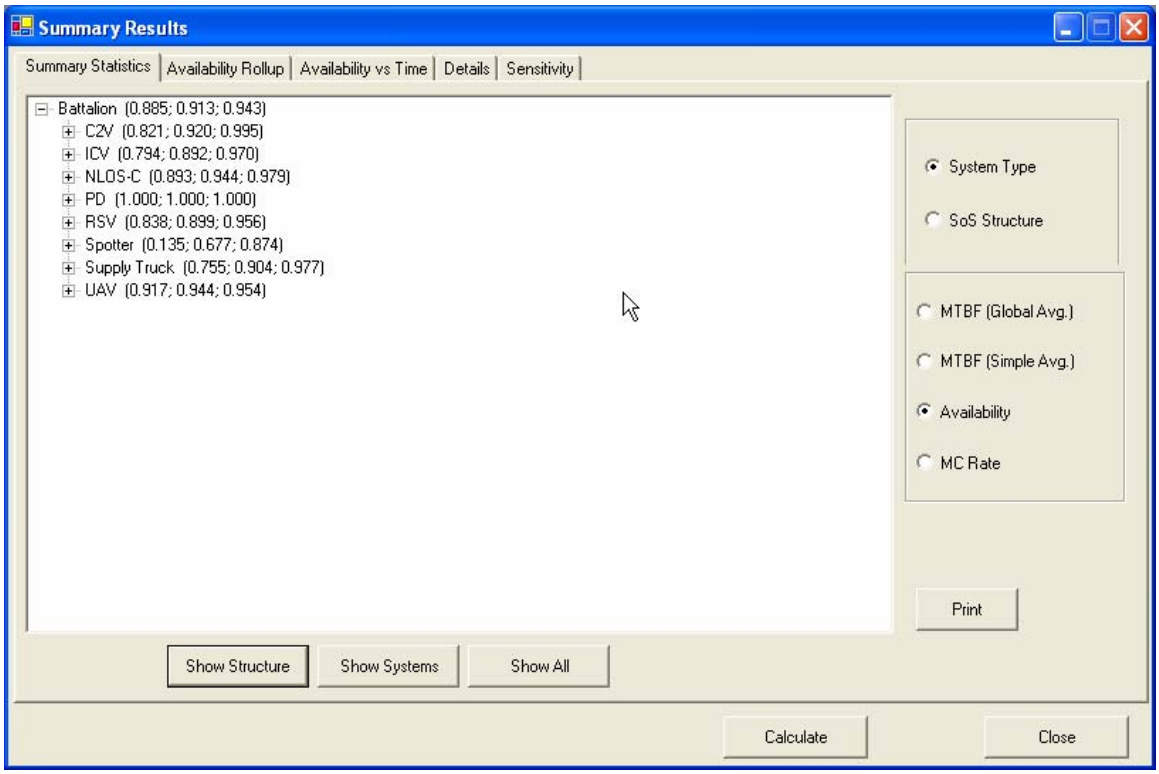

Figure 4.10 Availability by System Type

Figure 4.10 shows availability by system type. The values in parenthesis represent the 5th, mean, and 95th percentiles for availability for all systems under the node. Similar results can be found for mission capable rate (MC rate). The difference in the availability and MC Rate calculation is that availability accounts for operating and downtime hours whereas MC Rate includes operating, operable and downtime hours.

The next tab (Availability Rollup) in the summary results output form provides several options for evaluating the probability of mission success. Figure 4.11 shows a case where the success criterion is 70% availability throughout the mission for each system type. Figure 4.11 indicates that the probability of maintaining at least 70% availability is 1.0 for every system type except the spotter which has a 0.64 probability of maintaining at least 70% availability. Thus the probability that every system type in the battalion will maintain at least 70% availability is 0.64.

Similar calculations can be performed by specifying a minimum MC Rate by node or a minimum number of operable systems by node.

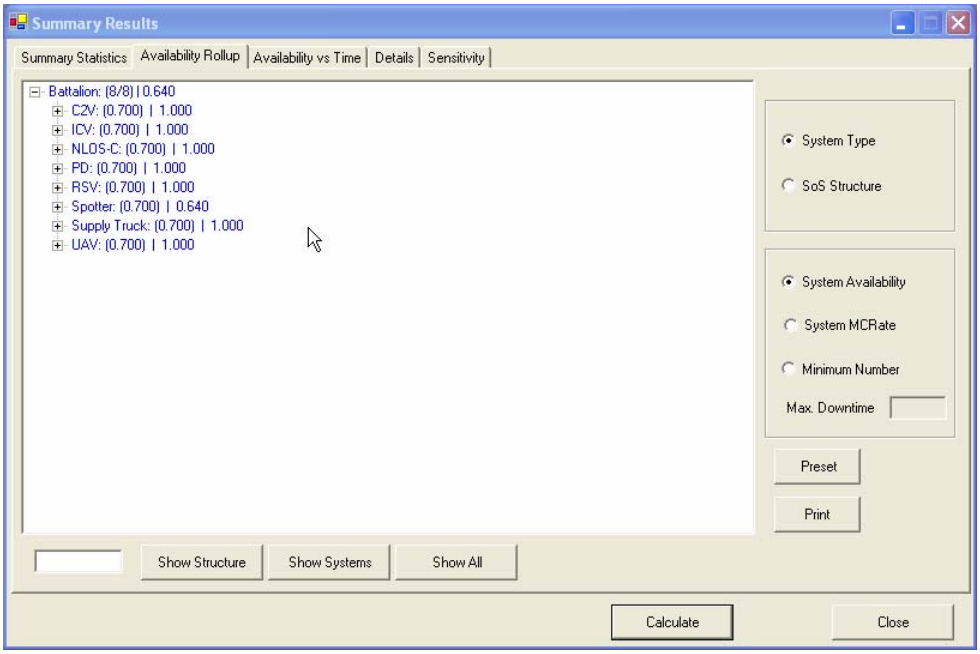

Figure 4.11 Probability of Mission Success for Availability Requirement

The Availability vs. Time tab displays the instantaneous availability as a function of time for any node in the force structure. Figure 4.12 shows results for the 16 RSVs. The four columns of results show the time, the 5th percentile, mean, and 95th percentile of instantaneous availability. Results can be copied and pasted into a spreadsheet for further analysis or display.

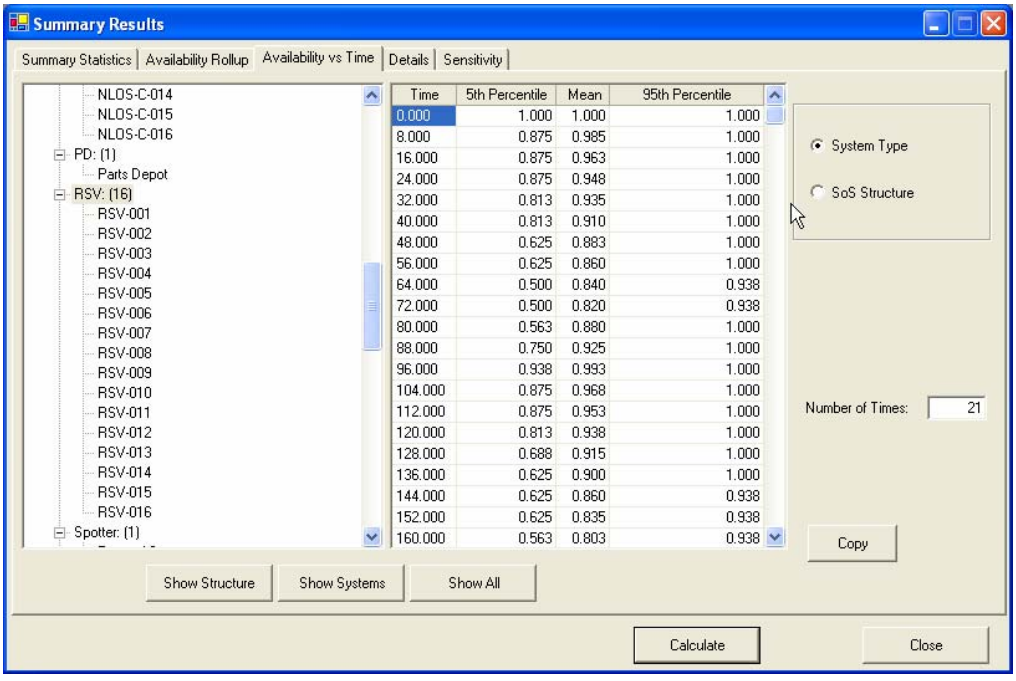

Figure 4.12 Instantaneous Availability versus Time for RSVs

The Details tab provides a means for examining detailed results for a system or group of systems. Figure 4.13 shows an example of the available results for the 32 UAVs. Results are shown by simulation or run number and include operating, operable, inoperable, and downtime hours as well as number of failures (not visible in Figure 4.13). Note that inoperable hours represent time when a system's desired state was operable but the system was inoperable. Downtime hours represent time when a system's desired state was operating but the system was not able to operate.

The final tab (Sensitivity) has not yet been implemented.

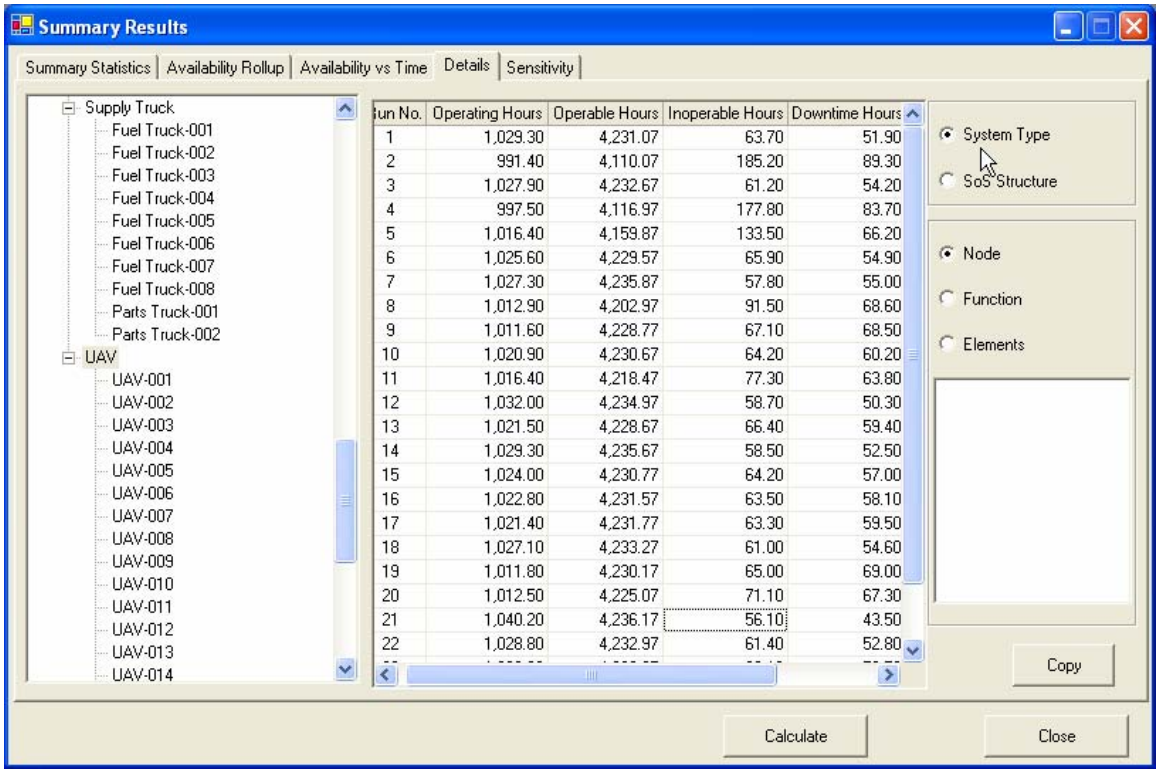

Figure 4.13 Detailed Results for the UAVs

4 Conclusions

A state modeling approach has been developed and implemented into both static (SyOp) and time-simulation (SMO Simulation) analysis capabilities. State modeling as developed in this LDRD has several benefits:

- A state model is quite flexible in the level of modeling detail. The approach readily adapts to high-level, overview models or to very detailed models that analyze systems in depth.
- A state model can have multiple goal states which means that multiple performance measures can be analyzed using a single model.
- A state model can have different sets of initial states. Typically results are desired for the case when every system is initially in its fully operational state. On the other hand if some systems are inoperable or are partially operable, the user can define the initial states that way.
- Goal states are not restricted to inoperable states. The state model can contain partially operable conditions.
- A state model can contain multiple systems.
- It is easy to incorporate dependencies between systems in a state model.
- External elements such as bad weather, rough terrain, or turbulence can be readily incorporated into a state model.

The time simulation capability, incorporating state model objects, has been used for detailed simulation of hundreds of interacting systems. This capability represents a significant accomplishment toward providing an ability to evaluate systems of systems. This ability did not exist at the beginning of the program and, as far as is known, does not currently exist elsewhere

As indication of the success of this LDRD, the U.S. Army, based on initial LDRD accomplishments, funded a large program with Sandia for SoS evaluation of the Future Combat Systems program, with $1.4M in FY04 non-LDRD funding. Funding for FY05-FY06 is projected to be $2.6M per year. Further, the SoS evaluation methodology has been defined as core to the Program Manager, UA Logistics Integration Directorate logistics assessment needs and the Army Evaluation Center's approach to developing test plans based on SoS performance evaluation. For application to the FCS, a comprehensive SoS logistics treatment approach was conceived and designed under this LDRD. The actual implementation in the SoS simulation was accomplished under the program with the U.S. Army.

Some of the specific accomplishments of this LDRD are:

1. A state model object has been designed, developed, and demonstrated to serve as the basis for representing individual systems in a SoS simulation.

2. A new state modeling capability has been developed that can analyze multiple measures of effectiveness for individual systems or SoS. This capability was implemented in software with an innovative, new interface.

3. An approach has been developed for handling and analyzing the large scale redundancy present in many SoS problems.

4. Techniques for handling potential sources of "singularities and nonlinearities" have been researched. These "singularities and nonlinearities" are phenomena that can cause a system-of-systems to exhibit behaviors that are substantially different from what might be expected by analyzing individual systems.

5. Several large-scale FCS UA supportability assessments have been performed.

6. A full supply network has been added to the SoS simulation for comprehensive, flexible treatment of spare parts, consumables, and constrained maintenance resources.

7. A beta version of the State Modeling Interface (SMI) is currently being tested.

5 References

Akers, S. B., Binary Decision Diagrams, IEEE Transactions on Computers, Vol. c-27, No. 6, June 1978.

Harel, D. and Naamad, A, The STATEMATE Semantics of StateCharts, ACM Trans. Soft. Eng. Method, 5:4, Oct. 1996.

Helbig, J. and Kelb, P., An OBDD-Representation of Statecharts, EDAC-ETC-EUROASIC 1994, pp 142-149, 1994.

Appendices

Appendix A: State Model Input

State models are created and edited using the State Model Interface (SMI). This appendix describes the steps required by the SMI to create and execute state models. State models in the SMS are comprised of the state hierarchy and transitions (section 3.1) and functions and other supporting information. The user provides the latter using the New Model Wizard and the former using the Editor Screen.

When creating a new state model the SMI automatically runs the New Model Wizard, so it is discussed first (section A.1). After all steps have been completed, the Editor Screen (section A.2) appears so that the state hierarchy can be drawn. It is good practice to completely define the state hierarchy before entering transitions. The Editor Overlays (section A.3) can be of assistance to the user during the state model building process. Section A.4 describes how to run a model.

A.1. New Model Wizard

Creating a new State Model is simplified by using the New Model Wizard. To start the wizard, select New from the File menu. The wizard will display 7 pages which prompt for information necessary to create the model.

Except for the introductory page and the finish page, the pages are there to enable you to provide input for the functions and other supporting information. All of this input can be later changed as necessary. To do so, select the Model Properties item from the File menu. From there the RunTime tab gives access the Model Options Page (section A.1.2), the Model tab gives access to the Data Libraries Page (section A.1.3), and the Functions tab gives access to the Functions Page (section A.1.4). The performance measures (section A.1.5) can be edited from the Functions tab also.

A.1.1 Wizard Introduction Page

The introduction page (Figure A.1) serves as a welcome and a preview of the upcoming pages. The upcoming pages are listed in this appendix in the order presented by the wizard. They are previewed on the left hand side of Figure A.1. Select Next to proceed to the Model Options page.

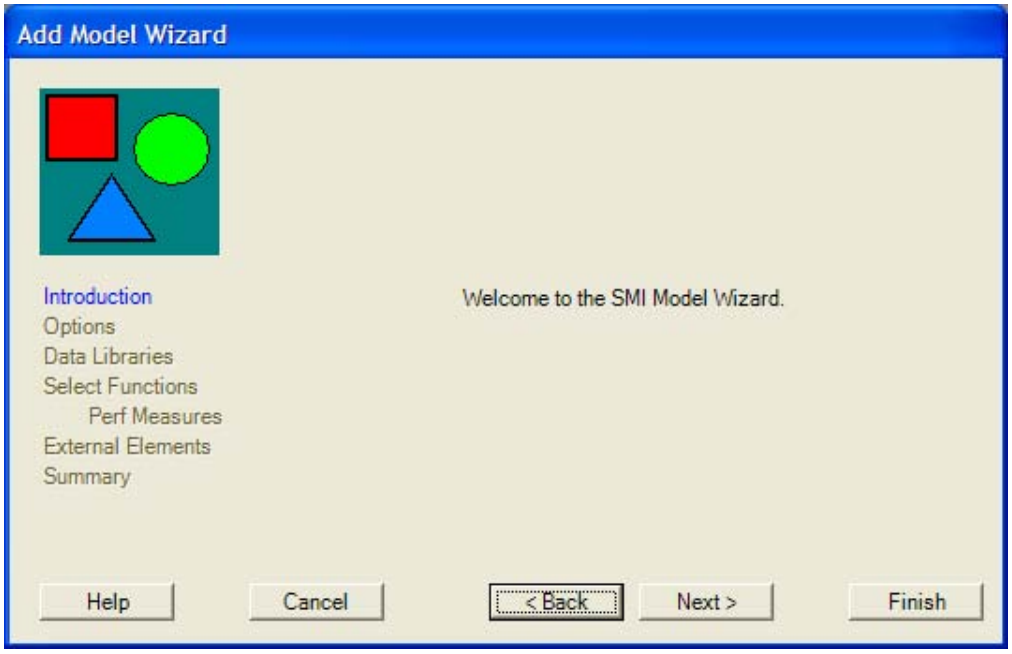

Figure A.1 New Model Wizard Introduction Page

A.1.2 Model Options Page

The Model Options Page has two radio buttons and 5 edit boxes, as shown in Figure A.2. This page can be reached for editing an existing model by selecting Model Properties from the File menu and selecting the RunTime tab.

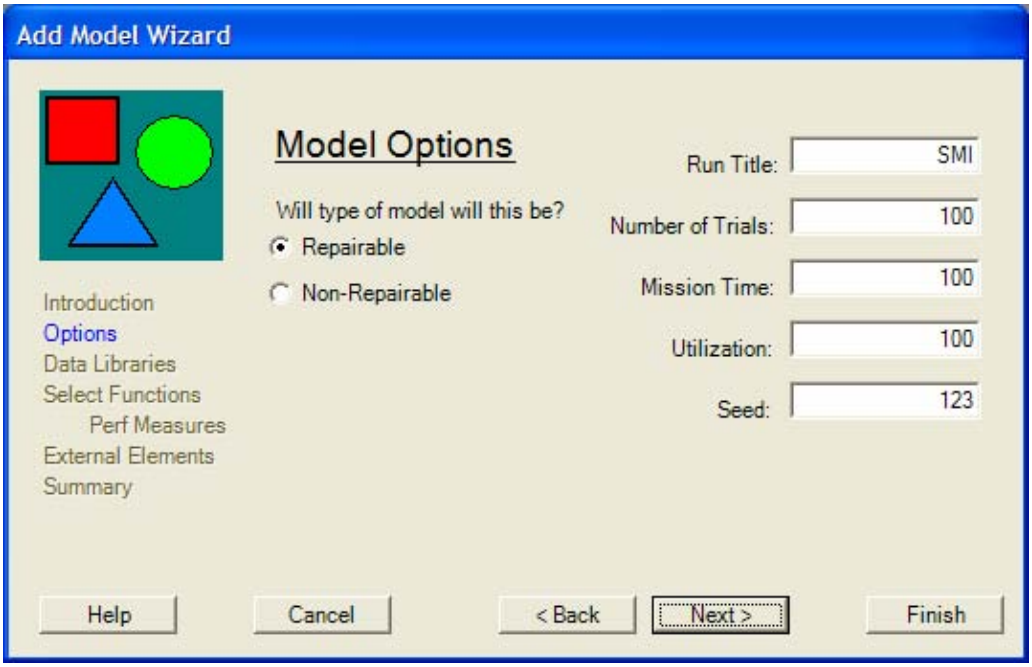

Figure A.2 New Model Wizard Model Options Page

One radio button must be selected. Select Repairable if the frequency of reaching a goal state function is of interest. If the probability of reaching the goal state function is of primary interest, select the Non-Repairable button.

Data entry for two of the edit boxes is specific to the radio buttons, as described next. After entering all information, click Next to proceed to the Data Libraries page.

- *Repairable:* Select whether the model is Repairable or Non-Repairable.
- *Run Title*: The run title is used to label outputs. The title should be short and descriptive.
- *Number of Trials*: The SMS uses statistical sampling and repeated trials to characterize the uncertainty in analysis results. Enter the number of trials here. The number should typically be between 200 and 500.
- *Mission Time*: The mission time is only needed for a non-repairable analysis. It is the time (hours) over which you want to determine the probability that the modeled system will reach one or more of its goal states.
- *Utilization*: Utilization is only needed for a repairable analysis. Utilization is the fraction of time that the system is expected to be in its operating state. For example, if a system operates 8 hours a day for 7 days a week, its utilization would be 0.33.
- *Seed*: – The random seed is used to initialize the random-number generator before statistical sampling is done. For a particular model, use the same seed and same number of trials if you want to duplicate a previous run.

A.1.3 Data Libraries Page

The Data Libraries page is used to attach one or more Data Libraries to the model (Figure A.3). Select Add Data Library to attach a Data Library. Select Remove Data Library to remove the currently selected Data Library. This page can be reached for editing an existing model by selecting Model Properties from the File menu and selecting the Model tab.

As described in section 2.1, the SMS develops a failure expression that describes the combinations of events that must occur in order for the state model to transition from the initial states to the goal state. To quantify this expression for obtaining useful metrics for the goal state, each event must have numerical properties. The events and their properties are maintained in a SyOp data library. Here you are providing the name of that data library.

It is anticipated that some models must utilize more than one data library in order to provide all pertinent events. At this point however, only one data library can be used. After attaching the data library, select Next to proceed to the Functions page.

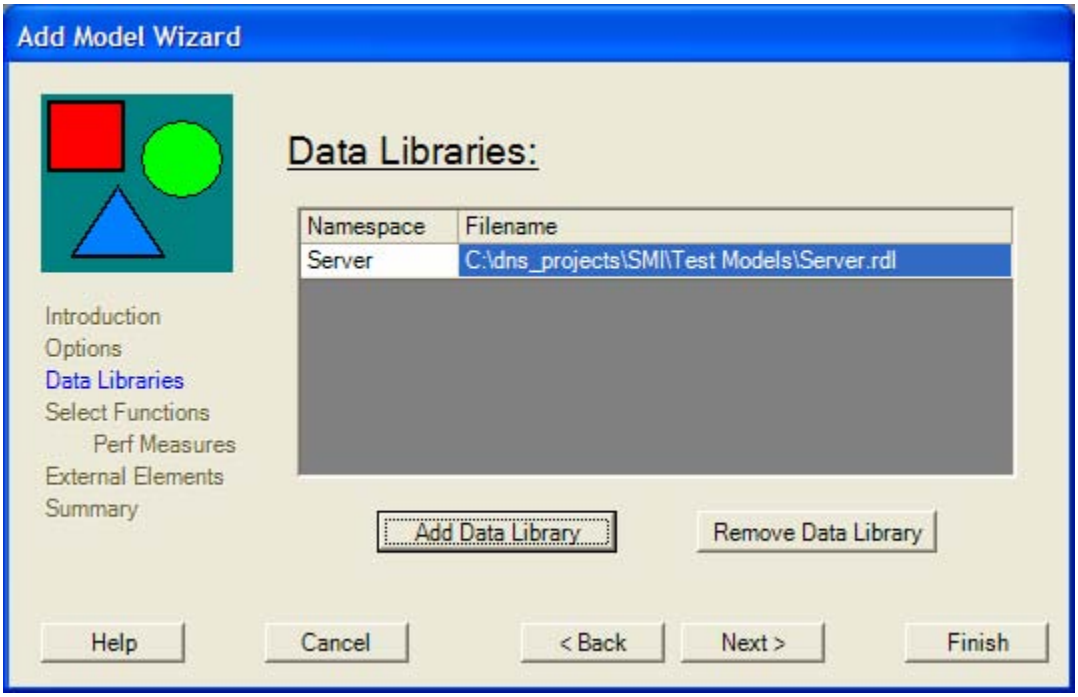

Figure A.3 New Model Wizard Data Libraries Page

A.1.4 Functions Page

The Functions Page (Figure A.4) is used to define names and symbols for functions that are important for the state model. Each function will also have a Performance Measure Page (section A.1.5, below) where additional information is required. On the Functions Page each function will be named and have a symbol and color associated with it. To delete a function, either type over the Function name to reuse the row or highlight the row and select Delete Function to remove the row. This page can be reached for editing an existing model by selecting Model Properties from the File menu and selecting the Functions tab.

Suppose your state model hierarchy describes the condition of a military system. Candidate functions can describe the lethality, the mobility, and the operability of the system, for example. Lethality need not necessarily be modeled as fully operable versus fully inoperable. A partially operable function could also be defined; for example, one of its two weapons may be down. Enter the functions here that you intend to model and examine.

Other features of this form include:

- *Populate* works in conjunction to the drop down list to its left. As the user builds models, previously defined functions will be collected into the drop down list. If a previously used function is the same, or is very nearly the same, as a new function to be specified here, the user can select it from the drop down list and click the Populate button. The previous function will be added as a new function.

- *Delete Function* deletes the highlighted function from the list.
- *Default* works in lieu of having a collection of previously defined functions. Default functions are available from the SMI using this button.

During the input of the state model hierarchy whenever you declare a state as a goal state (section A.2.5, below), you must associate that state with a predefined function. The SMI will display the list of candidate functions according to the information you enter here. When you execute the state model (section A.4, below), the SMS generates results for those functions that have assigned goal states.

The symbols entered here are a convenience. They will be used to identify which states are relevant to the function (in addition to the goal state). More specifically, each state that must be passed through to transition from the initial states to the goal state associated with this function will contain the symbol defined for the function.

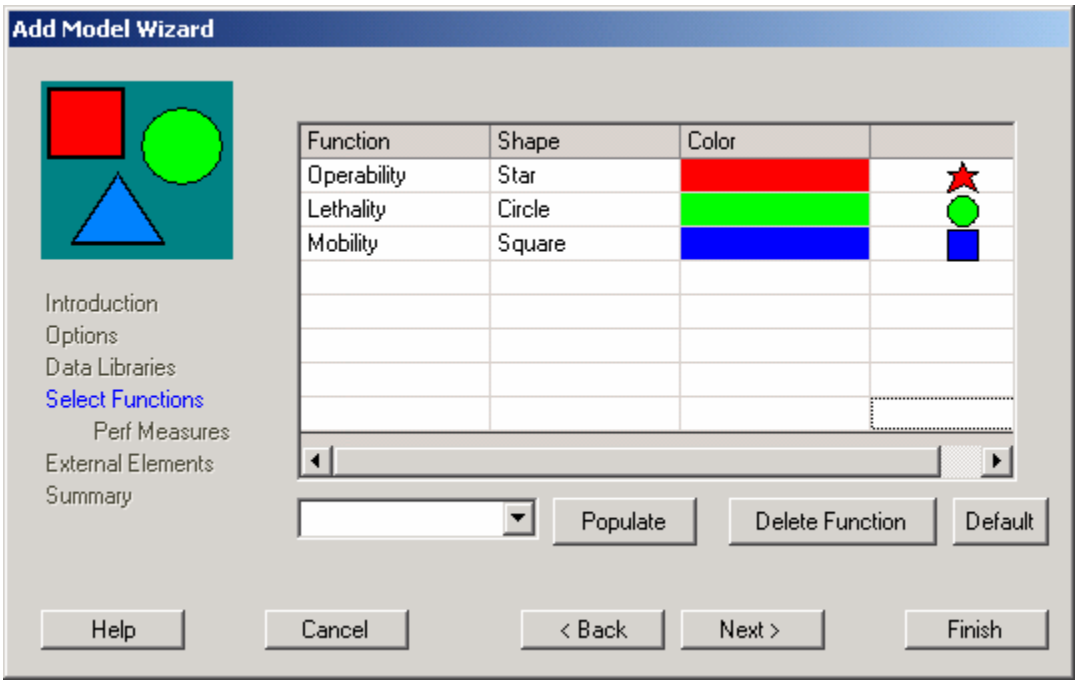

Figure A.4 New Model Wizard Functions Page

A.1.5 Performance Measures Page

The Performance Measures Pages are used to define a set of Performance Measures for each Function. So this page will be repeated for each function defined in section A.1.4. If you are editing performance measures for an existing model, they are reachable by selecting Model Properties from the File menu, selecting the Functions tab, and then clicking in the Performance Measure column.

The Performance Measures shown in Figure A.5 are relevant to a Repairable model (section A.1.2). If the state model was to be analyzed as a non-repairable model, there would be three rows, one each for reliability, unreliability, and cost.

The Performance Measures column is hard-wired. These are the performance metrics that are evaluated by SyOp. It is at this point where the user can tailor these metrics to the functions within the state model that are to be evaluated. The SyOp Results Viewer uses the text entered here. Uses for each column include:

- *Performance Measure*: These are the performance metrics that are evaluated by SyOp.
- *Caption*: This text will appear on menu items. Enter brief yet meaningful text for the performance measure for the relevant function.
- *Label*: This text will appear on the graphical and tabular displays. Brevity is not as important here, but be aware that this will be the default plot title.
- *Units*: Supply the units for the metric. The text is used for labeling columns and axes.

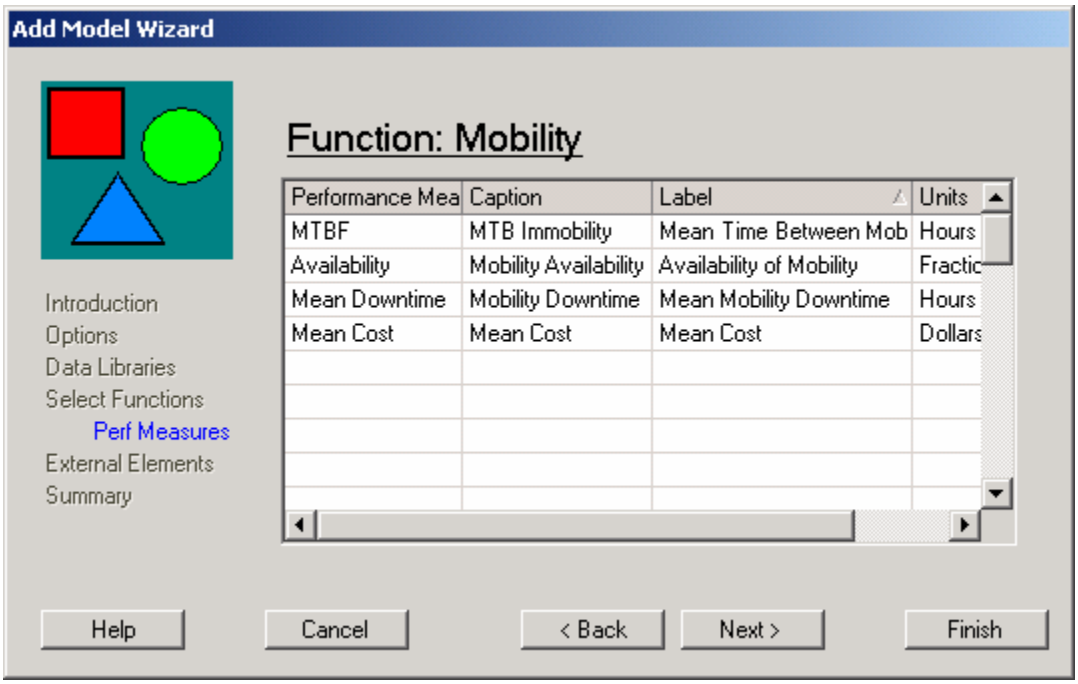

Figure A.5 New Model Wizard Performance Measures Page

For a repairable model the calculations assume that when an event occurs a component fails and that component can be repaired or replaced to be like new. The MTBF is the mean time between failures and the downtime is the time required to acquire replacement parts and to make the repair. For a nonrepairable model, there is a mission time on which the metrics depend. If an event occurs before the end of the mission time there is no repair.

Each performance measure is tailored to a function, which is associated with a goal state. The descriptions that follow may be helpful in naming the captions and labels.

- MTBF: The mean time between reaching the goal state.
- Downtime: The mean time spent in the goal state (or beyond).
- Availability: The approximate fraction of time *not* spent in the goal state.
- Cost (repairable): The cost incurred when the goal state is reached.
- Reliability: The probability of not reaching the goal state.
- Unreliability: The probability of reaching the goal state (or beyond).
- Cost (nonrepairable): The cost of reaching the goal state during the mission.

The parenthetical "(or beyond)" in the descriptions is relevant only if the goal state does not represent inoperability. If the goal state is for some intermediate level of partial functionality, then the "or beyond" includes the time spent when or probability of, exiting the goal state. An illustration of this is given for the example problem (section 2.4).

After entering all performance measure information for a function, click Next to proceed to the next function. After all functions are addressed, click Next to proceed to the External Elements page.

A.1.6 External Elements Page

The External Elements page is used to define the external elements that are relevant to the state model. Two examples are shown in Figure A.6. These elements can be included in guard expressions along with state names. For a given model execution each of these must be assigned a value, true or false. Currently if an external element is defined on this page and then used in a guard expression, the element is assumed to be true (occur). In the future there will be a second column on this page where Boolean values can be assigned to the element. After entering the desired External Elements, select Next to proceed to the Finish page.

A.1.7 Finish Page

Select Finish on this page (Figure A.7) to exit the New Model Wizard and proceed to the Editor Screen. It would be a good time to save your model input thus far.

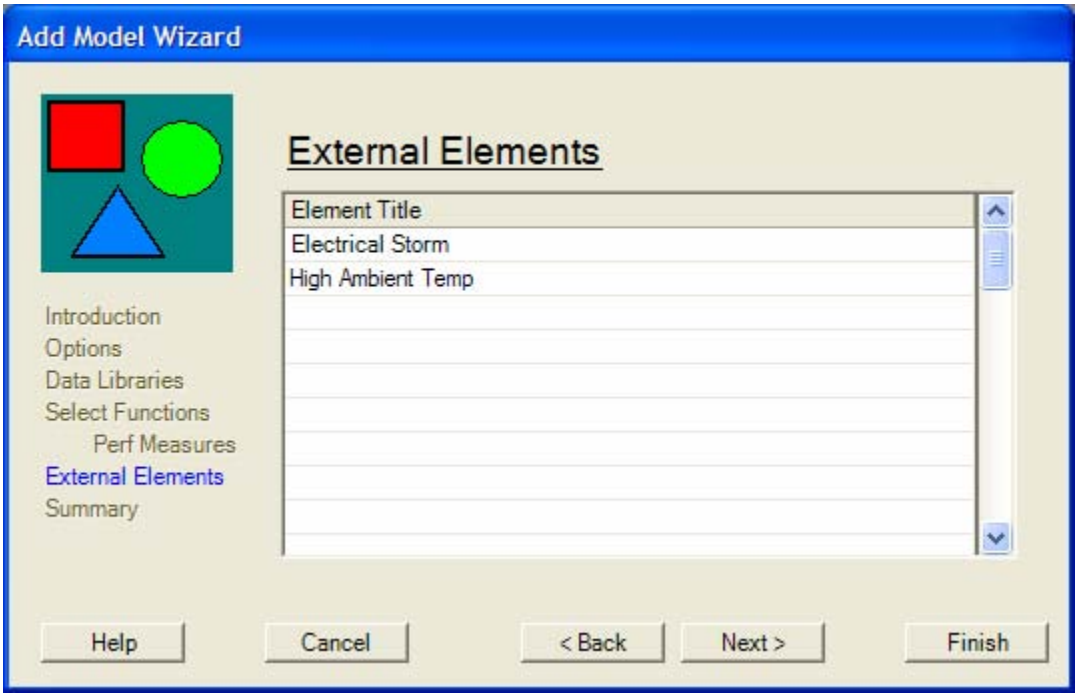

Figure A.6 New Model Wizard External Elements Page

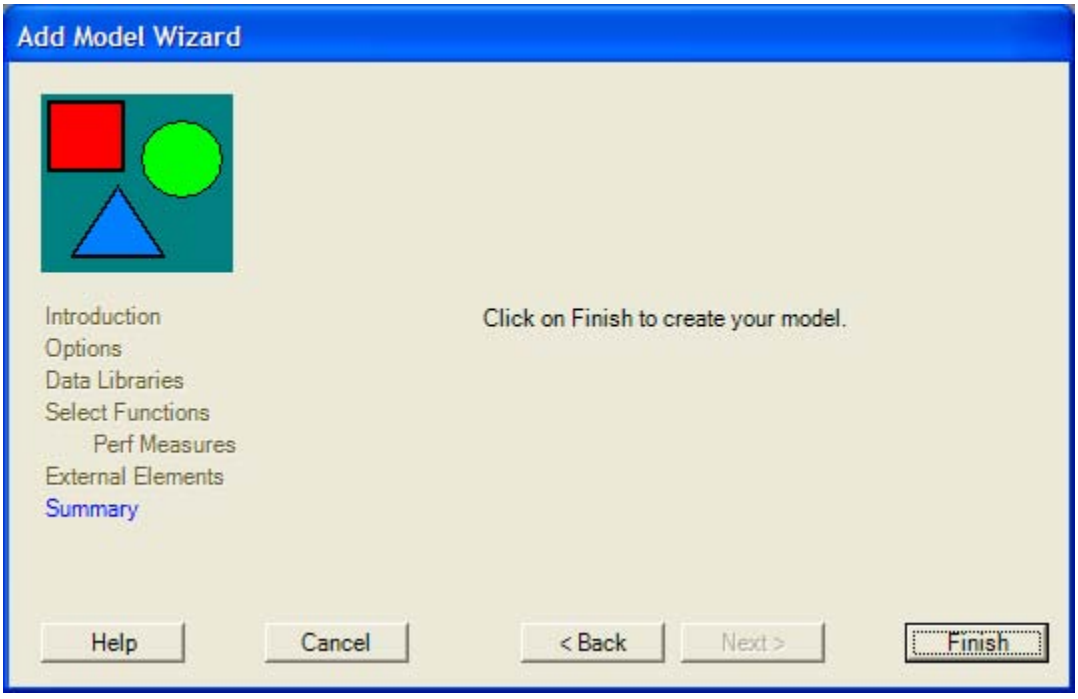

Figure A.7 New Model Wizard Finish Page

A.2 SMI Editor Screen

After completing the New Model Wizard, the SMI automatically displays the Editor Screen. The Editor Screen also appears each time an existing SMS file is opened. The menus and horizontal tool bar shown in Figure A.8 are present to perform standard operations. Only the Build menu is specialized for the SMI and therefore discussed in this document (section A.4). The nonstandard features of the initial Editor Screen are the vertical toolbar on the left, the Root state, the Legend, and the Map.

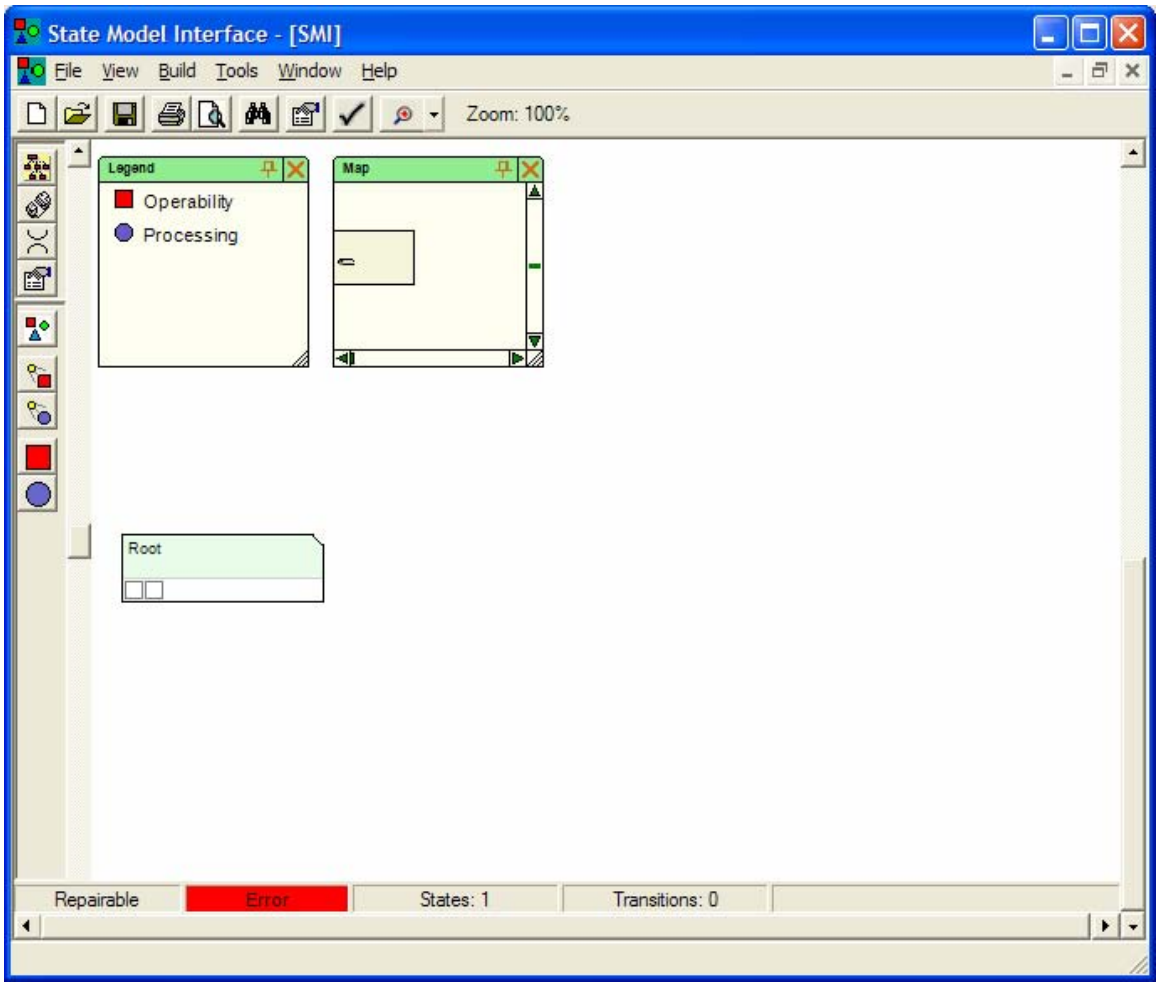

Figure A.8 SMI Editor Screen

The vertical toolbar provides several shortcuts that can be used while editing a state model. The meaning of each symbol is summarized in Table A.1. More detail can be found in the task descriptions throughout section A.2. Note that the number of function symbols shown on the toolbar is governed by the number of functions defined (section A.1.4).

Every state model hierarchy is anchored to a root state. The root state is the ultimate parent for every state in the model. The SMI provides this state as a starting point. It can be renamed but not deleted.

Table A.1 Description of Vertical Toolbar Symbols

Symbol	Meaning
	**Add a child state to the state that currently has focus** **Add a transition that will emanate from the state that currently has focus** **Change the decomposition for the state that currently has focus from OR to AND or vice versa** **Show states that are causing the state model to be unrunnable** **Show symbols for functions in all states for all functions** **Show/don't show symbols for functions in all states for the selected function** **Add the selected symbol to the state that currently has focus**

The Legend and the Map are discussed in section A.3. Briefly, the Legend shows the functions and their symbols. The Map shows a shrunken picture of the state hierarchy.

The Editor Screen also displays error conditions for your state model. The SMI attempts to determine whether your model is runnable or not on a real time basis. If there is an obvious input error or missing input, either a red Error flag is shown at the bottom of the window or a yellow Warning flag. To find the offending states use the finger pointer icon on the vertical toolbar (Table A.1). Following the completion of the New Model Wizard there is always an error. The reason is that there are no user-defined states in the model.

The remainder of this section discusses how to perform the basic tasks of state model construction. The recommended approach is to generally follow the three steps below. In practice however, steps 2 and 3 are interchangeable.

1. Build the state hierarchy. Add child states and declare the decomposition of the parent states.
2. Define initial states and goal states. Associate goal states with functions.
3. Define the transitions. Each must have a source state, at least one destination state, and either a guard or a trigger.

A.2.1 Adding New States

To add child states to a state, highlight the state by left clicking one time with the mouse and selecting Add State icon from the Toolbar on the left side of the screen. Alternatively, right click for the popup menu (Figure A.9) and select Add. The new state will be displayed on both the Editor screen as well as the Map overlay.

In Figure A.10 we have added four states. When a state is first added, it is displayed as a leaf state (light green). When children are added to a state, its color changes to that of a parent state (light blue) and its decomposition is displayed by the union symbol to the right of the state. To change the decomposition to AND, see section A.2.3, below.

Figure A.9 Popup Menu for States

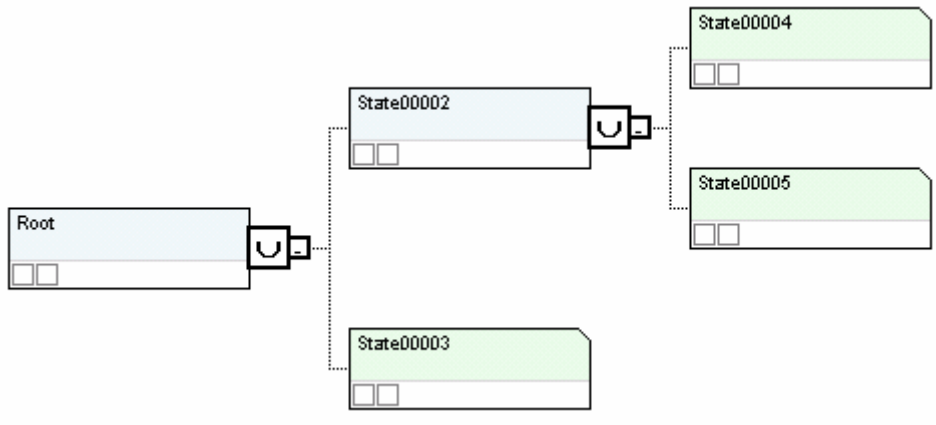

Figure A.10 SMI Editor Screen with New States

A.2.2 Renaming States

As shown on Figure A.10 the SMI initially numbers each new state, thereby assigning a unique default name to the state. To change a state name to a more meaningful name double click on the state using the left mouse button and enter the new name in the text box. Another method to change the name of a state is to right click the state, select Properties from the popup menu (Figure A.9), and enter the new name on the properties form. Example properties forms are shown in Figure A.11 below.

The SMS requires that all states have unique names. It is typical that a state model will have several states that represent inoperable states. It is prudent to prepend a system name, a function name, or an event name to the word Inoperable for such states. In that way UAV Inoperable is readily distinguished from NLOS Alternator Inoperable during model building and the state names will be unique.

A.2.3 Changing Parent State Decomposition

When children are added to a state, its color changes to that of a parent state (light blue) and its decomposition is displayed. The decomposition is defaulted to OR. Change the decomposition as necessary using the OR/AND icon (Table A.1) or the popup menu (Figure A.9).

A.2.4 Deleting States

To delete a state, right click the state and select Delete from the popup menu (Figure A.9). If the state is the source of a transition, the transition will be deleted as well. If the state is the destination of a transition with multiple destinations the state will be removed from the transition. If the state is the destination of a transition with a single destination, the transition will be deleted.

A.2.5 Setting Initial and Goal States

As described in section 2.3 the SMS traces all paths (sequences of states and transitions) from the initial states to the goal state and then finds a Boolean expression to describe the event occurrences along those paths. Setting initial states is defining the state model initial condition for the beginning of the analysis. Setting goal states defines the termination of the path finding analysis. The SMS finds a separate Boolean expression for each goal state/function pair.

To identify a state as being an Initial or Goal state, right click the state and select Properties from the popup menu (Figure A.9). On the property page (Figure A.11), select Initial State to identify this state as an initial state. Select Goal State and select a function from the Goal State combo box to identify this state as a goal state. After exiting the state properties page, the state model for System A is as shown in Figure A.12. Note the appearance of the "I" and "G" to designate the states as initial and goal.

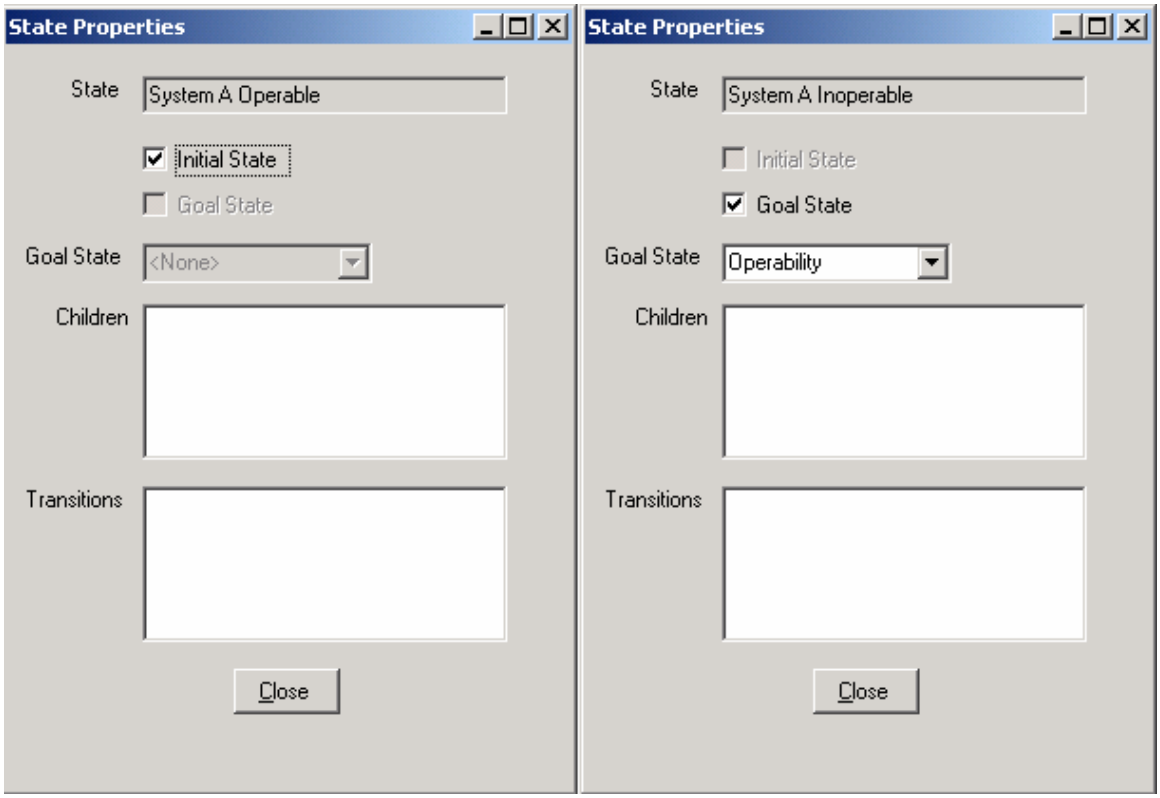

Figure A.11 State Property Pages for an Initial State and a Goal State

Figure A.12 exhibits a typical initial state and goal state pair. The state model begins in the initial state, System A Operable, and somehow transitions to the goal state, System A Inoperable. The transition can occur according to the terms you define. Do certain events have to occur (trigger) or does the state model have to have reached certain states (guard)? The mechanics of defining transitions and their guards and triggers begins in section A.2.6.

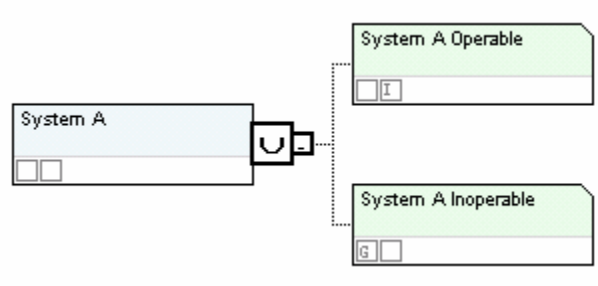

Figure A.12 States Shown as Initial States and Goal States

A.2.6 Adding Transitions, Define Source State

A transition has four properties, at least three of which must be defined – its source state, its destination state(s), and either a guard or a trigger, or both. You define the source state for the transition in the act of creating the transition. Select the desired source state and then select Add Transition from the toolbar on the left side of the editor screen (Table A.1). This automatically pairs the source state to the transition and the Transition Wizard appears (Figure A.13) where you define the remaining properties for the transition.

Figure A.13 Transition Wizard

A.2.7 Adding Transitions, Define Destination States

The Transition tab for the Transition Wizard initially lists all states that have been defined in the state model, except the source state for the transition, in the list box on the left hand side. Declare a destination state by highlighting the state name and selecting the right arrow. The state name will be moved to the list on the right hand side. States may be removed from the list box on the right by highlighting the state name and selecting the left arrow.

In Figure A.13 the transition for the virus detector state model is from the VS-Running state to the VS-Found state. If there is a use for a transition having multiple destination states, they can all be declared here. Alternatively, multiple transitions could be defined each having a single destination state.

A.2.8 Adding Transitions, Define Guards and Triggers

Clicking either the Trigger tab or the Guard tab on Figure A.13 brings up an editor whose appearance and mechanics are quite similar for either task. Figure A.14 shows the form as tailored for the guard expression. On display is the current expression for the guard. The names are state names and the plus sign indicates union (or). So the guard in Figure A.14 is true if the state model occupies any of the states listed. Thus, if the UAV loses its communications, electrical, mobility, or sensing functions, the guard expression is true. Presumably this guard belongs to a transition from the UAV Operable state to the UAV Inoperable state.

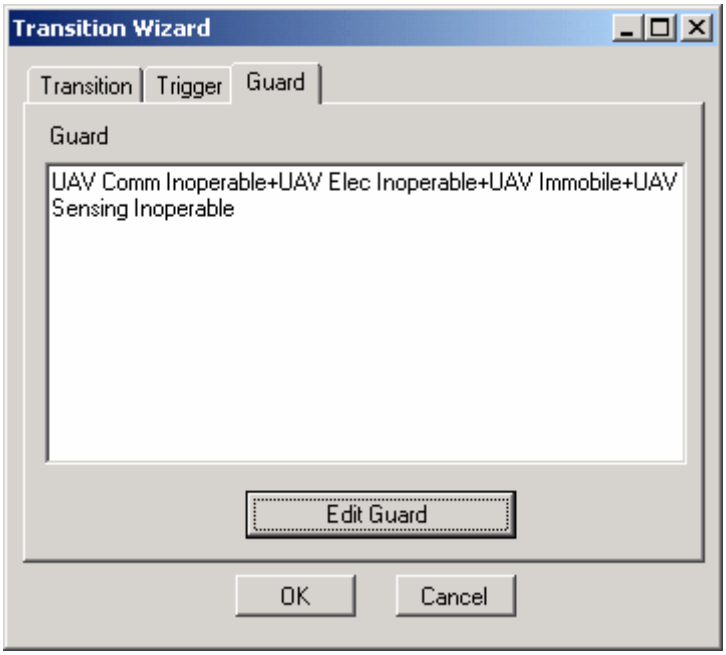

Figure A.14 Guard Display Form

For the initial definition of a transition, the display box will be empty. To enter an expression or to edit an existing guard expression, click the Edit Guard button. For

triggers there is the display of the trigger expression and an Edit Trigger button. The form that appears (for editing guards) is shown in Figure A.15.

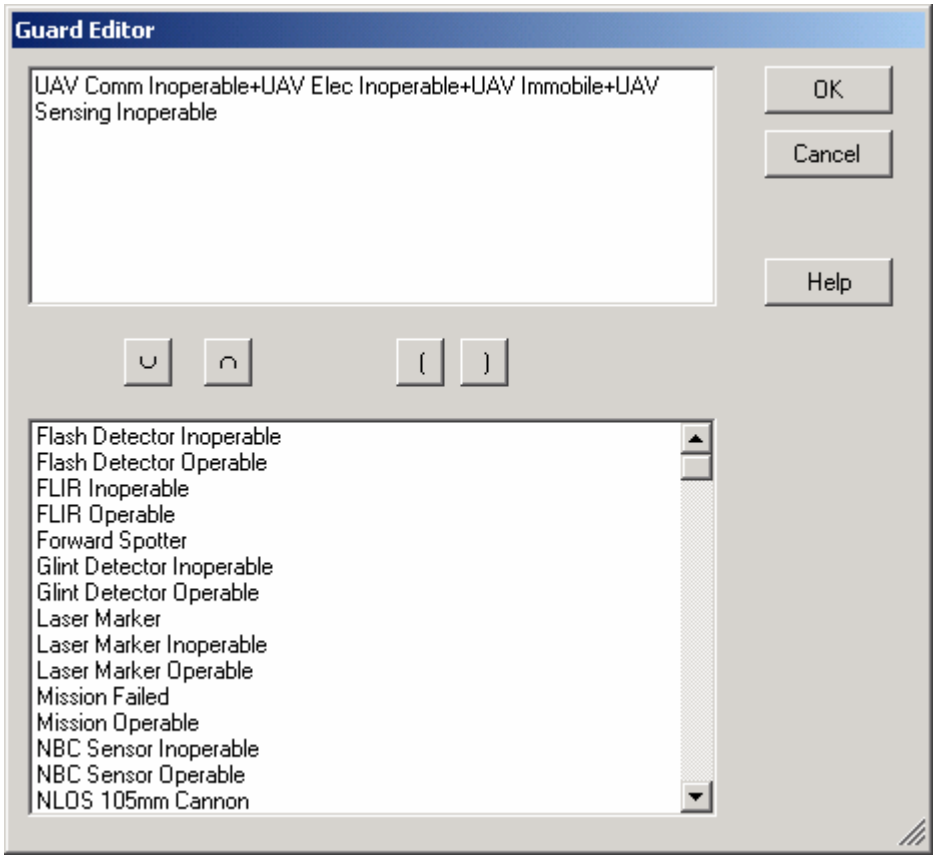

Figure A.15 Guard Editor Form

The guard expression, if there is one, is repeated in the edit box at the top of the form. To help enter or edit the expression the entire list of states in the state model is displayed in the bottom list box in alphanumeric order. To insert a state name at a particular point in the expression, first place the cursor at that point in the top box. Then, double-click the desired state name from the bottom box.

The only other symbols that can appear in a guard expression are the union and intersection symbols and the left and right parentheses. These can be inserted by placing the cursor at the desired point in the top box and clicking the symbol shown.

The difference for editing triggers is that the choices of names for the expression are event names. The SMI lists the failure mode IDs found in the attached data library (section A.1.3) in the bottom box.

A.2.9 Navigating Transitions

Figure A.16 shows how the SMI displays transitions. Transition number 3 connects its source state UAV Propulsion Operable to its destination state UAV Propulsion

Inoperable. The source state is denoted by a lavender circle and an outward arrow. The destination state is denoted by a peach home plate and an inward arrow.

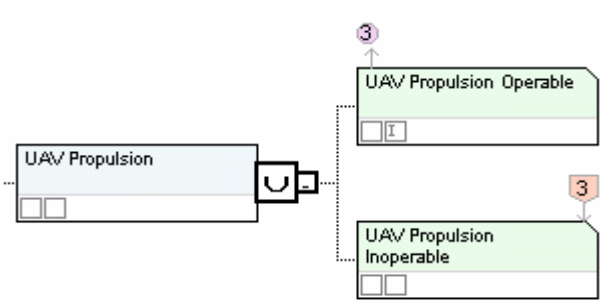

Figure A.16 Example Transition Display

The SMS assigns each transition a unique number for identification and bookkeeping purposes. The transition numbers are not editable. The viewer matches the outward transition number with the inward number to know which states are being connected.

For large models two connected states may not be displayable on the same screen at the same time. If the source state is viewable you can change the Editor Screen to display the destination state (and vice versa). Right click the input or output transition symbol to display a popup menu (Figure A.17). The third and fourth items show that the display can be moved to either the source or destination state.

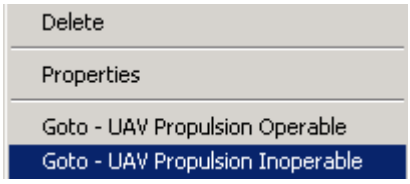

Figure A.17 Popup Menu for Transitions

A.2.10 Editing Transitions

To edit an existing transition, right click the transition and select Properties from the popup menu (Figure A.17). The Transition Wizard will be displayed and you will have the same options as when creating the transition (Figure A.13).

A.2.11 Deleting Transitions

To delete a transition, right click the transition and select Delete from the popup menu (Figure A.17). The editor prompts for confirmation of your intent to delete the transition.

A.2.12 Adding Function Symbols

Each state can have one or more function symbols displayed within its border. In the state model fragment shown in Figure A.18 the loss of the 105mm cannon causes the NLOS Cannon to lose lethality. If it loses lethality, it loses operability. Thus, the states for the 105 mm cannon affect those two functions. The cannon itself relies on other

inputs for targeting rather than the reverse. So the state of the cannon does not affect the ability to identify and locate potential targets. That's why the symbols for the targeting functions do not appear in the cannon states.

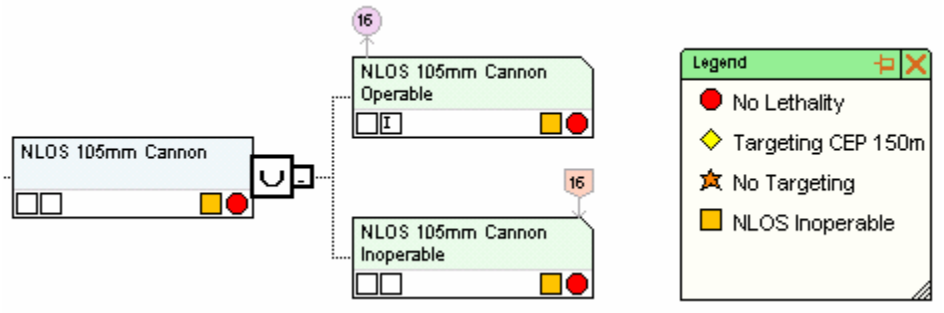

Figure A.18 Placing Function Symbols on States

The user can place the symbols into each contributing state using the appropriate function icons on the vertical toolbar (Table A.1). The legend for the functions can be of assistance here, as shown in Figure A.18. The legend is further described in section A.3.

It is not always obvious in a complex model whether all states have been correctly marked according to the functions to which they contribute. However, there is an automated way to verify the assignments described in section A.4.1, Path Validator.

A.2.13 Displaying Function Symbols

It may be useful at times for the analyst to be able to focus in on the states that affect just a single function. The focus symbols for the functions (Table A.1) act as toggle switches. Pressing in a switch dims all states that do not contain that symbol. Releasing the switch brightens the appropriate states. The master switch brightens all states that contain at least one symbol.

A.3 Overlays

There are two overlays that initially float over the Editor Screen. They provide additional information for state model construction. The Map overlay presents a bird's eye view of the model and allows the user to move their point of reference by moving the overlay's scrollbars. The Map overlay will always show the current portion of the model within a viewing window. Functions and associated symbols are displayed in the Legend overlay. Both are shown in Figure A.19.

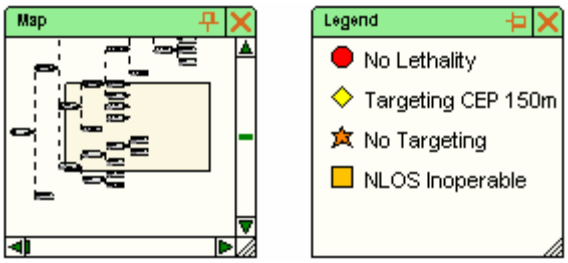

Figure A.19 The Map and Legend Overlays

From the initial location of the overlays, the user may:

- Pin the overlay into position by switching the pin button to pinned or unpinned. The Map is shown pinned and the Legend is shown unpinned in Figure A.19.
- Resize the overlay by clicking on at the bottom right corner of the overlay.
- Close the overlay using the close icon in the upper right.
- Display the overlay by selecting the overlay to display from the View menu.
- Return the overlays to their home positions by selecting Home overlay from the View menu.

A.4 The Build Menu

The two sub-items of the Build menu discussed here are Path Validator and Build Output. Each causes the State Model Interpreter to run in order to create the output described in the following subsections.

A.4.1 Path Validator

The primary function of the State Model Interpreter is to trace paths between the initial states and the goal state (section 2.3). The states that are passed through along these paths are the states that affect the function associated with the goal state. When you select Path Validator from the Build menu, these states are identified by the interpreter and displayed for the current goal state (Figure A.20). The current goal state appears in the title for the display.

Feature of this form include:

- *States* column shows the states that were determined by the interpreter to be in the relevant paths to the goal state, plus any states that had been manually checked by the user (section A.2.12) that the interpreter did not detect.
- *Original* column has a check mark for each of the states that were assigned to this goal state prior to the current run of the Path Validator. These would include those manually checked by the user (section A.2.12) and those found from a previous run of the Path Validator. In Figure A.20 there were no previous assignments made for any state so none are checked.

- *Reflected* column shows the states that were determined by the interpreter to be in the relevant paths for the goal state. Each is shown as checked
- *Apply* button will make the assignments for the current goal state/function to the states checked in the Reflected column.

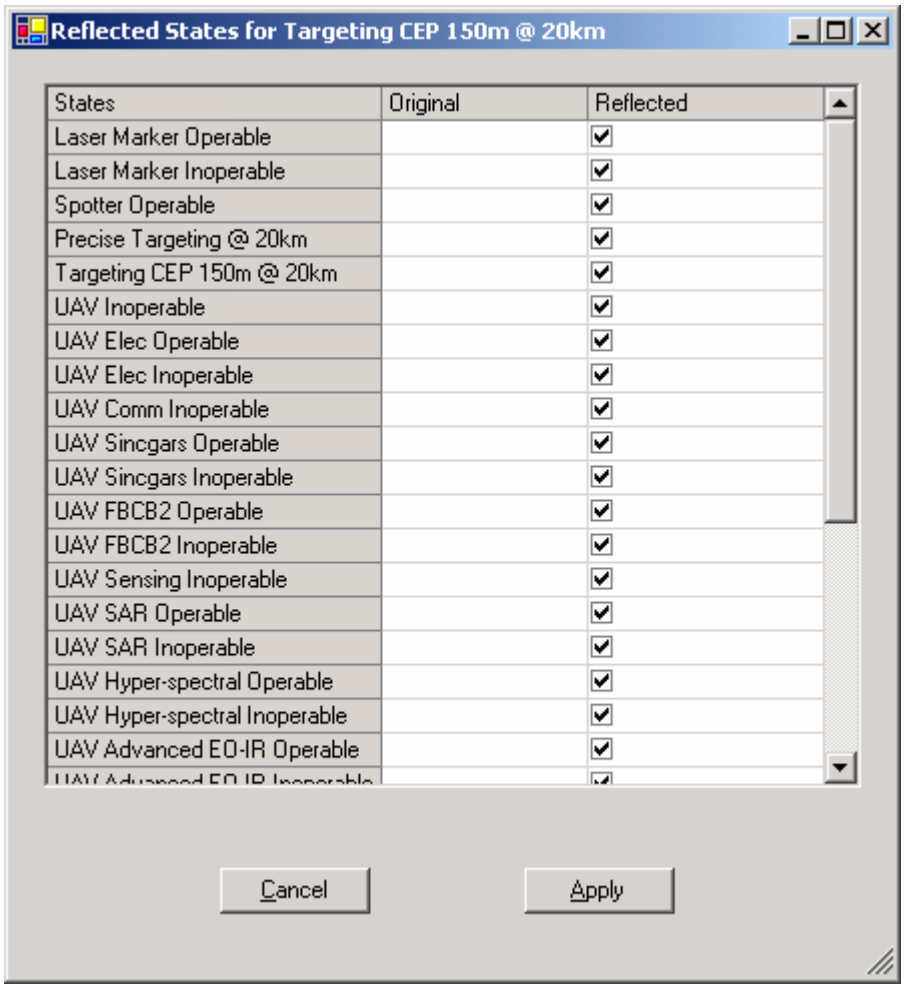

Figure A.20 Example Reflected States

A.4.2 Build Output

To evaluate the state model and generate output files, select Build Output from the Build menu. The SMI will prompt for an output file name, generate multiple output files, and run the SyOp ResultsViewer with pre-loaded model output files.

Because the SMS has been integrated into SyOp, it must follow SyOp's file handling and naming conventions. The file name requested by the SMI is called a fault tree group file in SyOp and has an FTG extension. This is the only file name that the user must provide.

Each file within a fault tree group must have an RFT extension. The SMS creates these files, one for each goal state/function pair. The root names of the RFT files are formed

by appending the goal state name to the name you provide for the FTG file. Each RFT file will contain the name of the associated data library (section A.1.3).

Finally, the results of the run are placed into files with an MOD extension. These are paired with RFT files so they have the same root name as the RFT file. So SyOp/SMS creates two files for each goal state, one with an RFT extension and one with an MOD extension.

The SyOp documentation provides complete definitions of the outputs available from the Results Viewer. The sample problem in section 2.4 shows some examples.

Appendix B: Input Description for SMO Simulation

The amount of input required by the SyOp Simulation software (SMO Simulation) varies by problem size but in general it can be quite large. A system-of-systems simulation may have multiple system types (platform types) and multiple instances of each type. Examples of system types could be NLOS cannons, RSVs, C2Vs, etc. If the simulation involves 10 NLOS cannons, then there are 10 instances of the NLOS cannon system type.

The input for a single instance of a system can be quite lengthy. Fortunately for most problems much of the input for one instance of a system is identical to that of every other instance of the same system type. So, considerable time can be saved if you enter the input for one instance and then duplicate it. That is, define the input for one NLOS cannon and then tell SMO Simulation to make 9 copies.

This alpha version of SMO Simulation has been developed quickly to meet an evolving need – the analysis of complex SoS with dependencies across systems. As a result, the organization of the input is certainly not optimal. Furthermore, many of the features described here have not been fully tested. While the user interface is serviceable, no claim is made as to its robustness. The user should expect to encounter some input problems and inconsistencies. We expect to redesign the user interface over the coming months to make it easier to use. Until then, the user is cautioned to be patient and careful.

The items under SMO Simulation's Edit menu are shown in Figure B.1. Selecting an item brings up an associated editing form. Each of these forms is displayed and discussed in the subsections of this section.

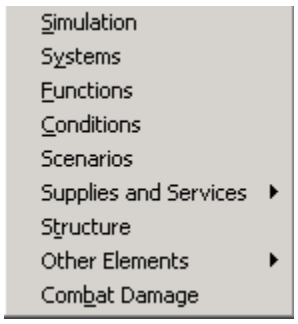

Figure B.1 Items under the Edit Menu

B.1. Simulation Parameters Input

The form for defining simulation parameters is shown in Figure B.2. Each input parameter is described below.

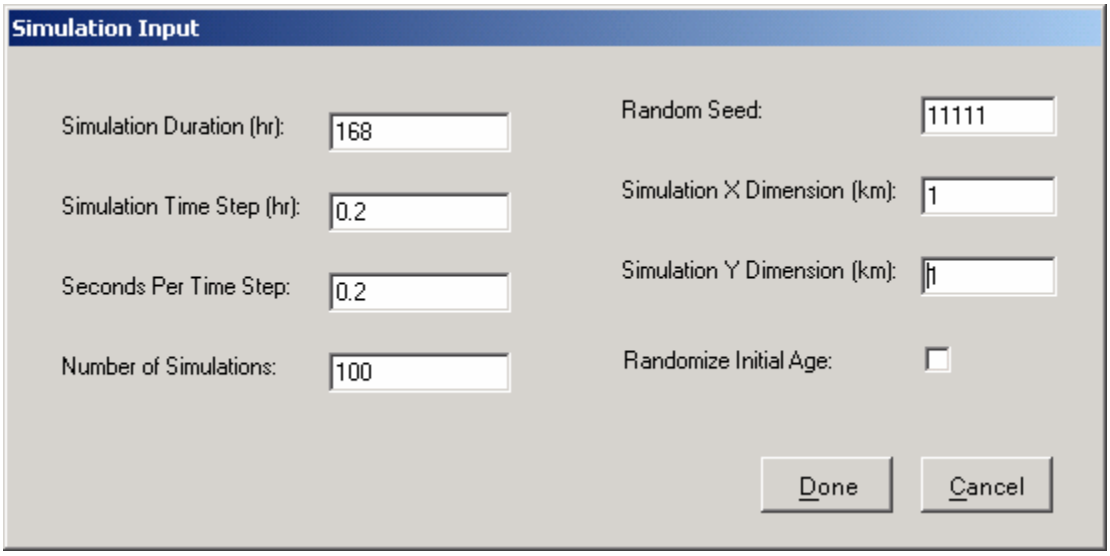

Figure B.2 Simulation Input

- *Simulation Duration* is the mission time in hours.
- *Simulation Time Step* is the size of time step SMO Simulation will take for simulation in hours.
- *Seconds per Time Step* is the amount of clock time between updates of the display that shows progress for the simulation. This input can be used to slow down the simulation when you want to watch real-time results.
- *Number of Simulations* is the number of times SMO Simulation repeats the simulation.
- *Random Seed* is used to initialize the random number generator.
- *Simulation X Dimension* is maximum distance any system can move in the x-direction, in kilometers.
- *Simulation Y Dimension* is maximum distance any system can move in the y-direction, in kilometers.
- *Randomize Initial Age* should be checked if you want SMO Simulation to randomly assign initial ages to each primary element. These initial ages will change from simulation to simulation. *This feature is not yet fully implemented into SMO Simulation*.

B.2. Systems Input

The form for defining systems has three tabs. They are described in sequence in the following subsections. Special buttons on each tab and any supporting forms are also described. If the simulation will include external elements (Section B.8.1) or consumables (Section B.6.2), you should input that data before entering systems

information. Then, information on external elements and consumables will be available as you create the individual systems of the simulation.

B.2.1 System Properties

Figure B.3 shows the first tab, the System Properties tab. You cannot navigate the tabs in this form by clicking the tab labels at the top of the form. Instead, you must use the indicated buttons to move from tab to tab. In this case the Edit Elements button takes you to the Primary Elements tab, for example.

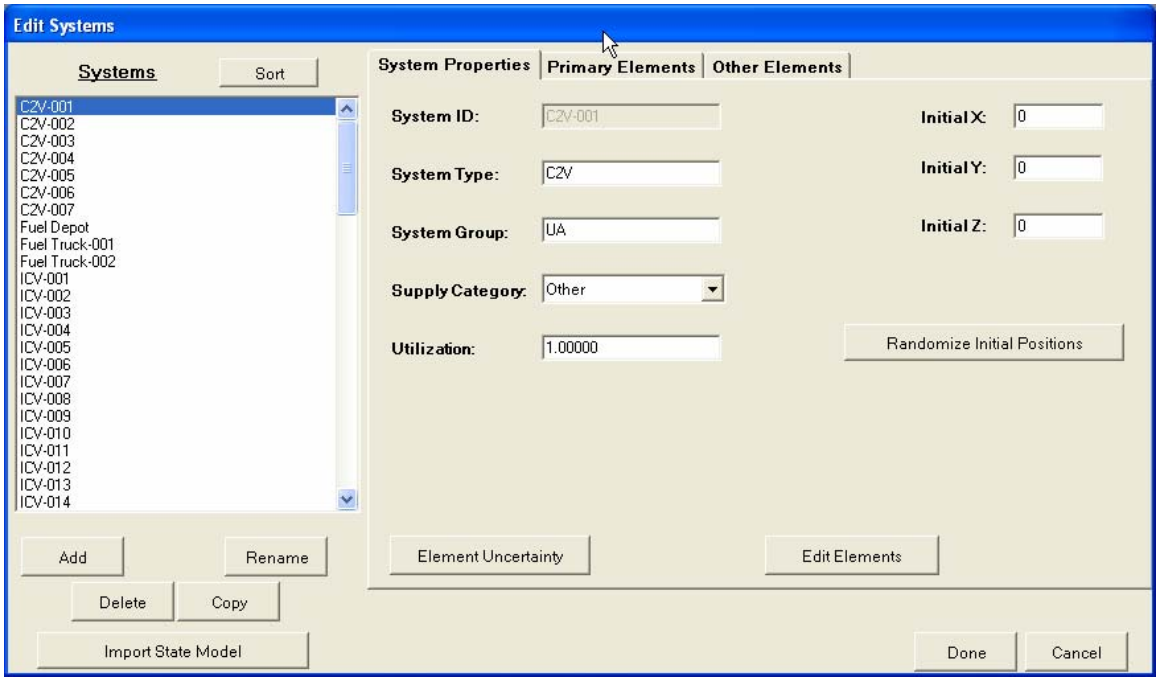

Figure B.3 Edit System Properties Tab

- *The List of Systems* is initially blank for a new setup. Use the Add button to enter a new system ID, which brings up another form (Figure B.4).

If you plan to simulate multiple systems of one or more system types, you should enter one instance of each system type first. After completing all input for the system (all 3 tabs here and the Functions for the system, section B.3), you can return here and duplicate the system as many times as desired using the Copy button. At that point SMO Simulation will assign a unique identifier to each copy. If you do plan to have multiple copies of a given system type, you should enter the name for the first one as Name-001. Subsequent copies of the system will then automatically be given names as Name-002, Name-003, etc. For example, if your first instance of a system type has a sequence number appended such as C2V-001, copies will continue the sequence as C2V-002, C2V-003, etc. So if you want to create additional copies of a system, select the highest number in the current sequence to be copied.

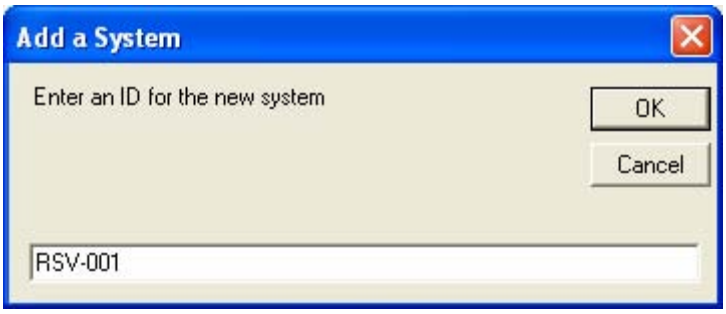

Figure B.4 Form to Add a New System

The remaining input for the system properties applies to the system selected in the list of systems. SMO Simulation repeats the appropriate system identifier in the grayed-out System ID field. As will be seen below, SMO Simulation shows the list of systems on the left for each subsequent tab, with the selected system highlighted.

- *System Type* is the type assigned to the system. For example, C2V-001, C2V-002, etc. might all be of type C2V.
- *System Group* This string identifies the group to which a system belongs. This input is not currently used.
- *Supply Category* indicates whether the system carries consumables to resupply other systems (not to be confused with carrying spares). Currently there are only two choices: Supply Vehicle or Other.
- *Utilization* is the fraction of time during the mission that the system is supposed to be operating. It is used to partition system up time between operating time and operable time, which is important for availability calculations. A more direct way to determine the operational time for a system is by defining detailed scenarios that specify when a system is expected to operate (see Section B.5).
- *Initial X, Initial Y, and Initial Z* are the initial position of the system in the x, y, and z-directions, in kilometers. These coordinates act in conjunction with the Simulation X Dimension and Y Dimension (section B.1) to locate the system in the plane. The system moves from this position forward according to its assigned speed. *Since the focus of the simulation is on the time behavior of systems rather than their location, this input may become obsolete.*
- *Randomize Initial Positions* should be clicked if you want SMO Simulation to assign initial positions to the system. The form shown in Figure B.5 appears where you give ranges for the x, y, and z positions. SMO Simulation will select coordinates at random within those ranges for the system. Note that if you duplicate this system, each new instance will be subject to these ranges but will have different randomly generated coordinates. *Because positioning of a system may not be supported in future versions of SMO Simulation, this property may become obsolete*.

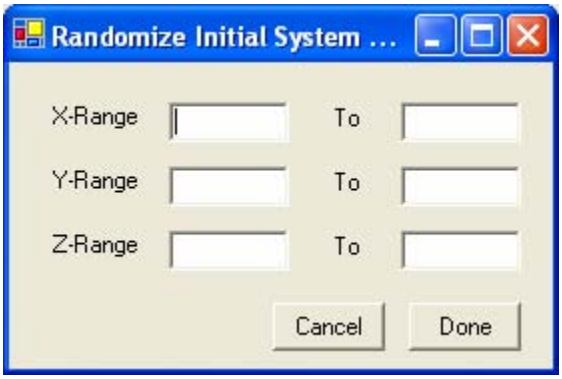

Figure B.5 Form for Initial Position Ranges

- *Element Uncertainty* button is present to be used in the following special case. As you will see, the button should be used only after you have entered the elements on the Primary Elements tab and have made all required copies of the system type.

  Suppose we can model the time-to-fail distribution for the transmission for a C2V with an exponential distribution, which has a constant failure rate. Further, we know a range for this failure rate. Specifically from historical records, the failure rate is somewhere between R1 and R2 and as is equally likely to be anywhere in that range. Because we are including the transmission as a primary element for the C2V, we want to use this information in SMO Simulation. Suppose there are 7 C2Vs in the mission. To represent our knowledge, sample 7 failure rates from the distribution of failure rates. In this case the distribution is a uniform distribution on the range [R1, R2]. We then want to use these sampled values as our estimates of the constant failure rate for the 7 transmissions. This means that we need to assign the sampled values to the required parameter (failure rate) for the exponential distribution for the 7 transmissions found on the 7 C2Vs. When you click the Element Uncertainty button (Figure B.3), the form shown on Figure B.6 appears to help facilitate the assigning of the sampled values. A couple of cautions are useful here. When the number of systems of a given type is small, as in the above example, you might want to use median stratified sampling to select the values to represent uncertainty in the time-to-failure parameter. For median stratified sampling, first divide the distribution into N (7 in the above example) intervals of equal probability or area. Then select the median value from the interval to represent that interval. This approach ensures that a small sample is still spread over the distribution which might not be the case for a small, random sample. The second caution is to ensure that the sampled values are randomly mixed before use so that you do not inadvertently place all high or low reliability systems in a particularly grouping.

The properties of the Element Uncertainty form include:

- *List of Elements* shows all of the elements pertinent to the system type. In our example the desired element is the Transmission.

- *Failure Rate* column is to contain the sampled values for the failure rate. Note that as a preliminary step you must ensure that each transmission element for the C2Vs has an exponential time-to-fail distribution. Otherwise, the headings for the grid on the right would be the parameters for a different distribution. SMO Simulation recognized that there are 7 C2Vs each with one transmission, so it placed 7 rows in the grid. Enter the sampled failure rates in the 7 rows. Currently, you must do the sampling outside SMO Simulation and enter the sampled values by hand. Future versions of SMO Simulation may automate this process.
- *Paste* button will paste values in the selected column of the grid if you have copied an appropriate number of values from a source such as a spreadsheet. The values in the grid are assigned to the appropriate time-to-failure parameter values when you click the *Done* button.

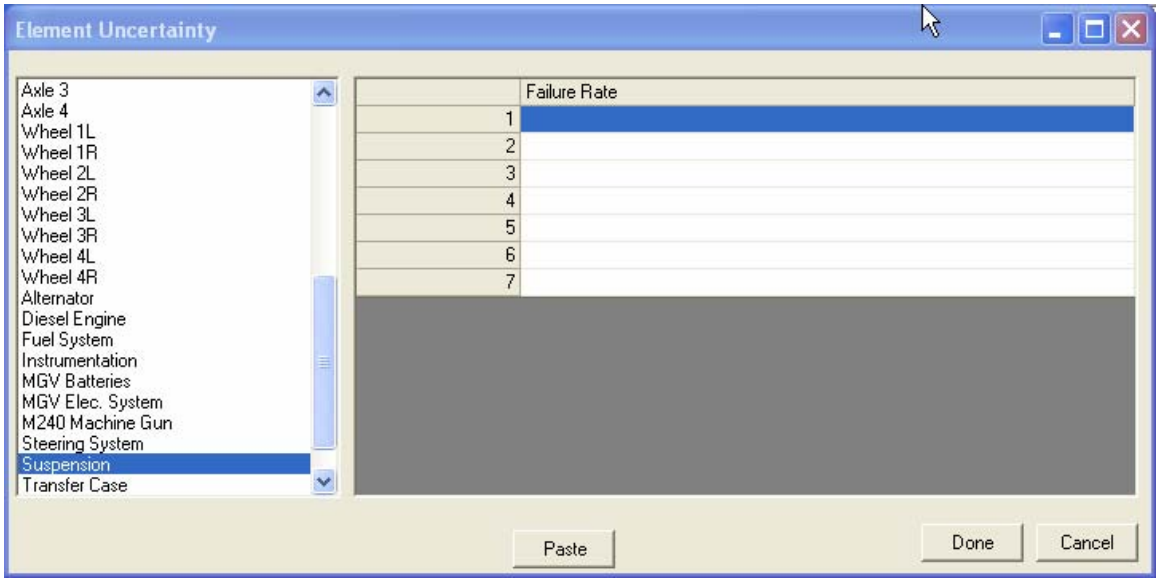

Figure B.6 Form for Element Uncertainty

B.2.2 Primary Elements

Figure B.7 shows the Primary Elements tab. The information on this tab is specific to the highlighted system on the System Properties tab (Figure B.3). It is reached by clicking the Edit Elements button on that tab.

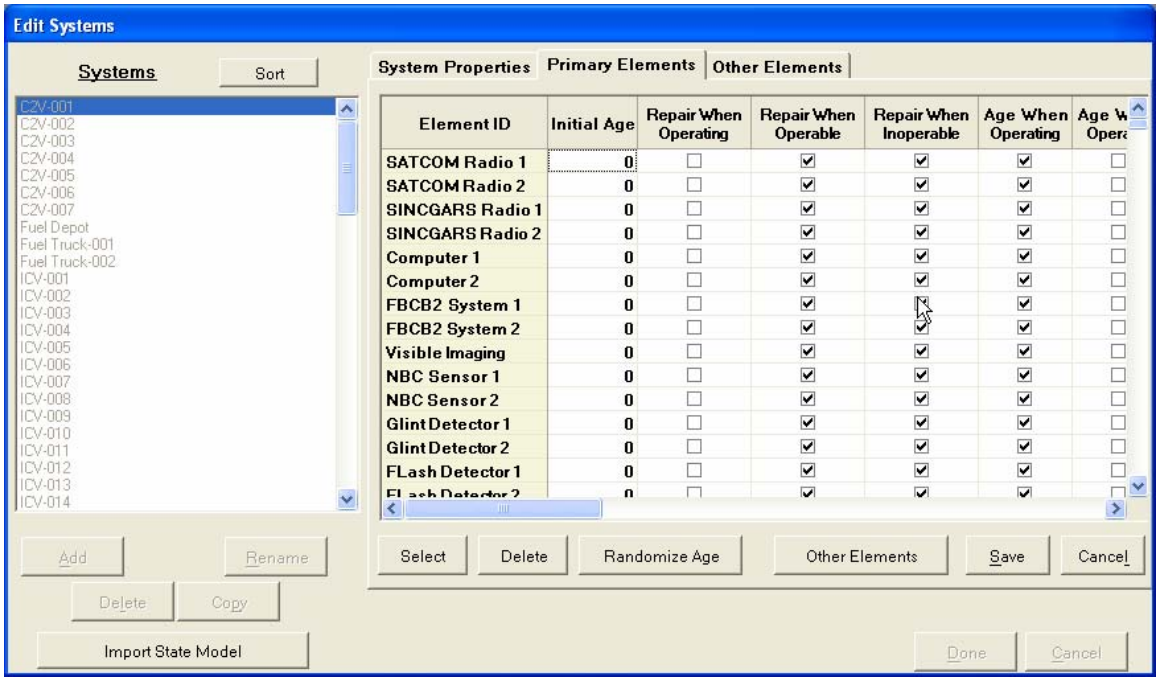

Figure B.7 Edit Primary Elements Tab

- *The List of Systems* includes all systems defined on the first tab and the selected system will be highlighted here. All editing changes you make will be applied to the selected system. If you need to edit a different system, use the Save or Cancel buttons, as appropriate, to return to the System Properties tab to highlight the new system.
- *Element ID* is a unique identifier for the element.
- *Initial Age* is the age in hours of the element at the start of the mission. The initial age for the SATCOM Radio 1 of C2V-001 most likely differs from the initial age for the SATCOM Radio 1 of C2V-002. So after making all copies of C2V, theoretically you would have to return here to enter the initial ages for each element of each instance of the system. Practically however, this information is probably not known. If you are using an exponential time-to-failure distribution, the initial age does not matter so it can be left at the default value of 0. However, if you are using a wearout or Weibull distribution, the initial age of the elements can be very important. If you do not know the initial age, you may want to use the option to randomize the initial age. Clicking the **Randomize Age** button displays the form below (Figure B.8). Selecting the **Default Option** will randomize initial element ages of elements between 0 and their median (50^{th} percentile) age based on the time-to-failure distribution for the element. Otherwise, you can specify a range as shown in Figure B.8 and the initial ages of all the elements will be randomly initialized within the specified range. If the **Apply to All Systems of Same Type** option is selected, all systems of the same system type will assign like elements the same initial age.

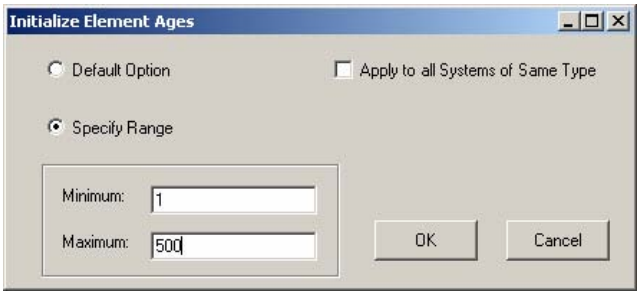

Figure B.8 Randomize Initial Element Ages

- *The Repair When Columns* are used to indicate the system states during which the repair of the elements can be accomplished. Suppose a C2V has two computers either of which can fully support the C2V. If one goes down the C2V has not lost any functionality. The question here is, can the failed computer be replaced while the C2V is operating, while the C2V is operable (but not operating), or when the C2V is inoperable. Check as many boxes as apply.
- *The Age When Columns* are used to indicate the system states during which the elements age. Most elements will probably only age while the system is operating. But it is possible for tires to deflate or seals to dry out when the system is idle (operable or inoperable). Check as many boxes as apply.
- *The Repair At Columns* (not visible on Figure B.7) are used to indicate the system locations at which the repair of the elements can be accomplished (field, repair facility, other). If a system is in the field but its required repair cannot be done in the field, the repair has to wait until the system reaches a repair facility. This usually requires that the system must reach a new segment in its scenario. Check as many boxes as apply. Future versions will probably generalize and expand the possible locations.
- *Spare ID* is used to identify the spare part required to make the repair (Figure B.9). All available spares are listed during spares input (section B.6).
- *Required Service* identifies a service that is required to return the element to its useful state when it fails. See section _ for maintenance services input.
- *Time to Fail Distribution* describes the failure probability versus time for the element. The cumulative probability at time t is the probability that the element failed at some time prior to time t. There are currently three distributions to choose from: wearout, Weibull, and exponential.

  The *wearout distribution* can account for all three phases of an element's lifetime: burn-in, random failures, and wear out. The SMO Simulation version of this distribution requires 5 parameters. The first two parameters assume that the end-of-life portion of the time-to-fail distribution can be described by a normal distribution.

  - The mean time to wear out (hours)

- The standard deviation for the wearout phase (hours)
- The fraction of failures that occur during the random failure phase
- The fraction of failures that occur during the burn-in phase
- The duration of the burn-in phase (hours)

The *Weibull* distribution can account for one of the three phases of an element's lifetime: burn-in, random failures, and wear out. The SMO Simulation version of this distribution requires 3 parameters.

- The shape parameter
- The scale parameter (hours)
- The location parameter (hours)

The *exponential* distribution models the constant failure rate assumption. The distribution requires 1 parameter, the failure rate ($hours^{-1}$).

- *Time to Repair Distribution* describes the time to repair the element. There are currently five distributions to choose from: uniform, triangular, normal, lognormal, and fixed. In addition to repair time, there are several simple delay time distributions required by the input. At each instance you will be referred back to here for distribution definitions. Simply replace the word "repair" below with the appropriate time category.

  The *uniform* distribution requires 2 parameters.

  - The minimum repair time (hours)
  - The maximum repair time (hours)

  The *triangular* distribution requires 3 parameters.

  - The minimum repair time (hours)
  - The best estimate repair time (hours)
  - The maximum repair time (hours)

  The *normal* distribution requires 2 parameters.

  - The mean repair time (hours)
  - The standard deviation for the repair time (hours)

  The *lognormal* distribution requires 2 parameters.

- The mean repair time (hours)
- The standard deviation for the repair time (hours)

The fixed distribution assumes the repair time is known and hence requires only one parameter, the repair time (hours).

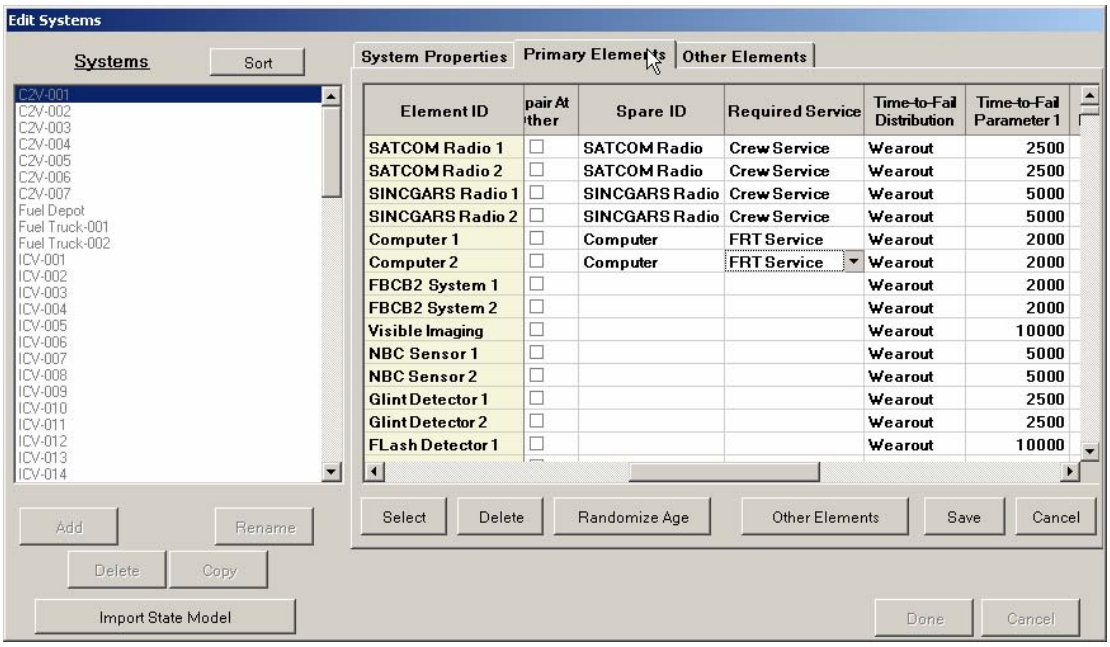

Figure B.9 Edit Primary Elements Tab

B.2.3. Other Elements

Figure B.10 shows the **Other Elements** tab. It is reached by clicking the **Other Elements** button on the Primary Elements tab (Figures B.8 and B.9). The consumables (section B.6.2) and external elements (section B.8.1) are eligible entries here.

All External Elements are listed on the lower half of the window and the input for an external element is simple - simply select the ones that affect the system. If you select an external element for the system, the element is available to affect a function of the system (section B.3).

We describe the properties for consumables on the upper half of the window.

- *Name* is one of the names you specified for consumables in section B.6.2. Click on a cell under the Name column and SMO Simulation gives you a drop-down list to select a name.

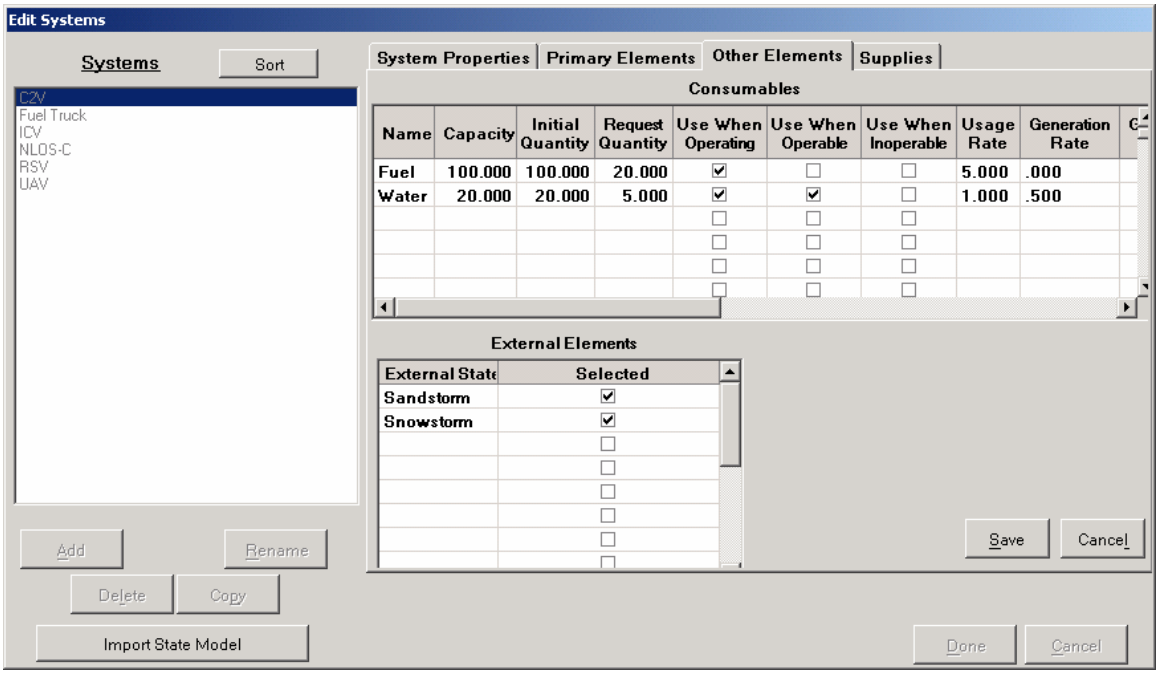

Figure B.10 Edit Other Elements Tab

- *Capacity* is the amount of the consumable that the system can hold. The units here must be those declared for the consumable (section B.6.2).
- *Request Quantity* is the amount of the consumable remaining for the system that is sufficiently low to trigger a request for replenishment. The units here must be those declared for the consumable (section B.6.2).
- *Use When Columns* are used to indicate the system states during which the consumable is used by the system. Check as many boxes as apply.
- *Usage Rate* is the rate at which the system uses the consumable, when it is being used. The numerator for the rate must be in the units declared for the consumable (section B.6.2) and the denominator is in hours.
- *Generation Rate* is the rate at which the system generates the consumable, when it is being generated. The numerator for the rate must be in the units declared for the consumable (section B.6.2) and the denominator is in hours. It is possible that some future combat systems will generate water, for example.
- *The Generate When Columns* are used to indicate the system states during which the consumable is generated by the system. Check as many boxes as apply.

B.3. Functions

The form for defining functions has three tabs. They are described in sequence in the subsections. It saves input time if you have only one instance of a system defined at this point. That way when you duplicate the system to the required number of instances, all of this input is also duplicated (section B.2.1).

B.3.1. General Function Properties

Figure B.11 shows the first tab, the General tab.

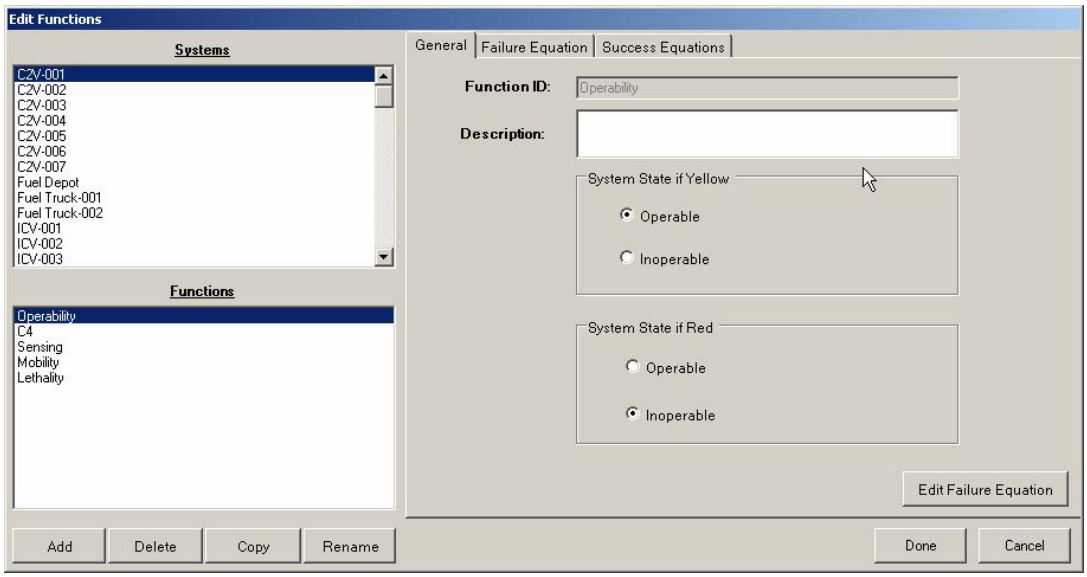

Figure B.11 General Tab for Functions

- *List of Systems* contains all systems defined thus far. All input on this tab applies to the highlighted system.
- *List of Functions* contains all functions defined thus far for the selected system. It is initially blank. You add a function by clicking the **Add** button, which brings up the simple form shown in Figure B.12. Enter the function name. The highlighted function is repeated in the grayed out Function ID box. All editing applies to this function of the highlighted system.

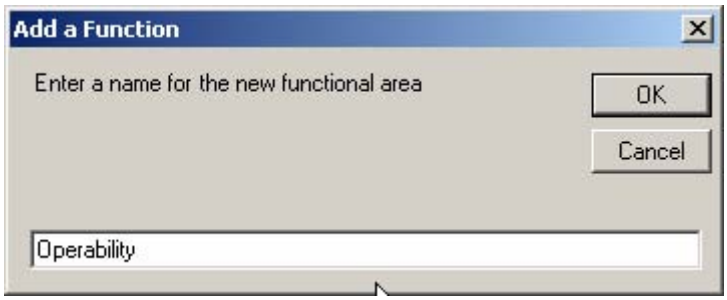

Figure B.12 Form to Add a Function Name

- *Description* is a place for you to add any notes about this function.
- *System State if Yellow* declares the state of the system if this function is yellow. A function can be green (fully functional), yellow (partially functional), or red (non-functional). SMO Simulation assumes that if the function is green, it does not reduce the state of its system. On the other hand, yellow and red functions can

reduce it to operable or inoperable. Select the system state if this function is yellow. An approach to using these options has evolved that seems to work well. For each system, the recommended approach is to define a function called Operability that represents all needed capabilities for the scenario being analyzed. Then only the Operability function would cause the system to fail. Other functions can then be added to represent such things as mobility, lethality, etc. but these would not necessarily cause the system to fail.

- *System State if Red* declares the state of the system if this function is red. Select the system state if this function is red.

B.3.2 Failure Equation

Figure B.13 shows the Failure Equation tab. You access this tab by clicking the Edit Failure Equation button on the General tab (Figure B.11). The currently selected system and function are highlighted in the lists on the left side. The list on the upper right side shows all of the elements that were assigned to the system (sections B.2.2 and B.2.3).

The failure equation is the logical "or" of a set of cut sets. If every member of a cut set occurs (fails) the cut set fails. The logical "or" implies that if any cut set fails, the function fails. If a cut set contains a single entry, called a simple cut set here, then the failure of that entry causes the function to fail. If a cut set contains N entries, all N have to fail to cause the function to fail.

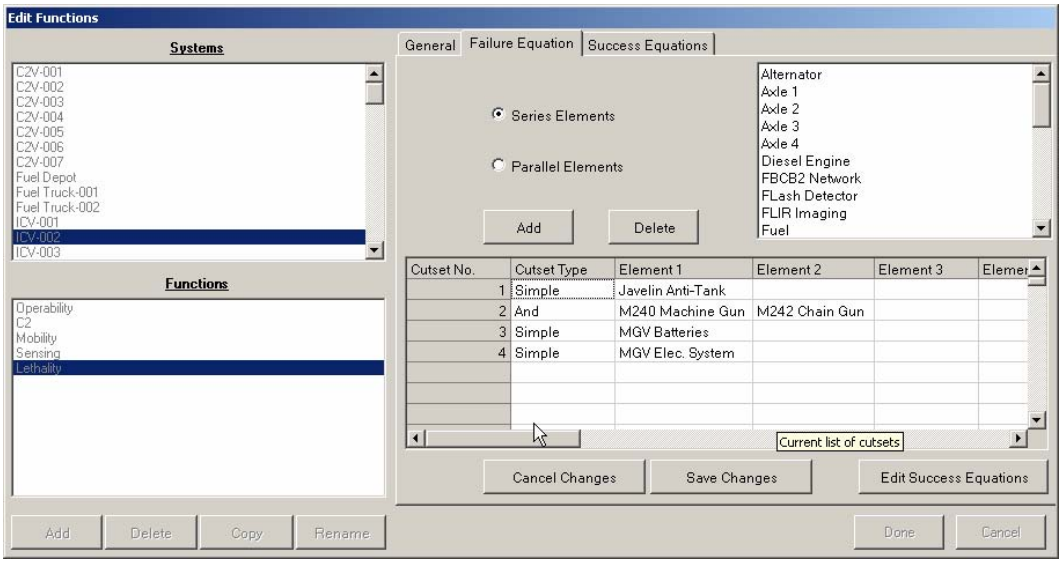

Figure B.13 Failure Equation Tab for a System/Function

- *Series or Parallel Elements* indicate the cut set to be added is to have one member (simple) or multiple members (and). When you are ready to add a cut set to the list, first select the appropriate radio button.

- *Select Cut set Members* is the next step to adding a cut set. Click on the elements to be included in the cut set from the list on the right. For a simple cut set, select one element. For a multiple cut set select as many elements as desired. To select multiple elements, hold down the Ctrl key while clicking on the elements.
- *Add* will add a cut set of selected type consisting of the highlighted elements. It adds the cut set to the first free row in the cut set grid.
- *Delete* will remove the cut set currently highlighted in the cut set grid. Cut sets cannot be edited. So, if you make a mistake while adding a cut set, delete it and then add it again.

B.3.3. Success Equations

Figure B.14 shows the Success Equations tab. You can access this tab page by clicking the **Edit Success Equations** button on the Failure Equation tab (Figure B.13). The currently selected system and function are highlighted in the lists on the left side. The list on the right side shows all of the elements that were assigned to the system (sections B.2.2 and B.2.3) except those elements that appear in a simple cut set in the failure equation.

Success paths are similar to cut sets in their structure. To understand how to input them, it helps to know how SMO Simulation treats success paths.

1. Success paths are only examined in a time step if no cut set occurs. That is, if a cut set occurs (fails), the function is red and no further checking is necessary, so steps 2 through 5 are skipped.
2. A success path is true if all its elements are true. Otherwise the success path is false (down). SMO Simulation counts the number of success paths that are down, say D.
3. If D = 0, then none are down and the function is green
4. If D = the number of success paths, then all are down. In that case, the function is red.
5. For any other count, the function is yellow and the state of the function is the first success path that is up.

Note in Figure B.13 we have shown the M240 Machine Gun and M242 Chain Gun in parallel. That means that we have no lethality only if both these weapons fail. Otherwise we have full or partial lethality. Suppose now that we want to distinguish between lethality with the M240 machine gun, the M242 chain gun, or both. We can do this with success equations.

In Figure B.14 we have defined two very simple success equations; 1) the M240 machine gun is operating and 2) the M242 chain gun is operating. At any time, the SMO Simulation will check the failure equation for lethality function of ICV-002. If the lethality function has not failed according to the failure equation, the success equations

will be evaluated. If the M240 Operating success equation evaluates to true while the M242 Operating equation evaluates to false, we know that ICV-002 only has the M240 machine gun operable at that time and its lethality function is at an intermediate state between operable and failed.

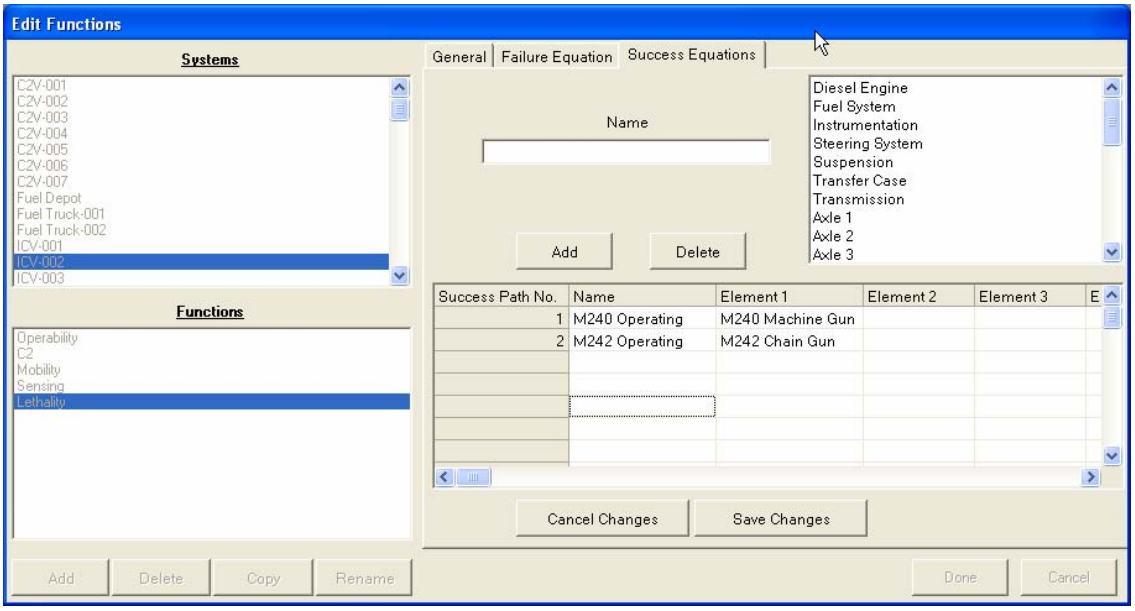

Figure B.14 Success Equations Tab for a System/Function

- *Name Edit Box* is the name given to the success path. Provide the name as the first step in adding a success path. This is used to identify the actual level of functionality if a function state is yellow.
- *Select Success Path Members* is the next step to adding a success path. Click on the elements to be included in the success path from the list on the right. For a success path with multiple elements, hold down the Ctrl key while clicking on the elements.
- *Add* will add a success path at the first free row in the success path grid.
- *Delete* will remove the success path currently highlighted in the success path grid. Success paths cannot be edited. So, if you make a mistake while adding a success path, delete it and then add it again.

B.4. External Conditions

The form for defining external conditions is shown in Figure B.15. Recall that external conditions can affect the properties of the elements of a system or of the system itself. External conditions can enter a simulation by being assigned to a scenario segment (Section B.5).

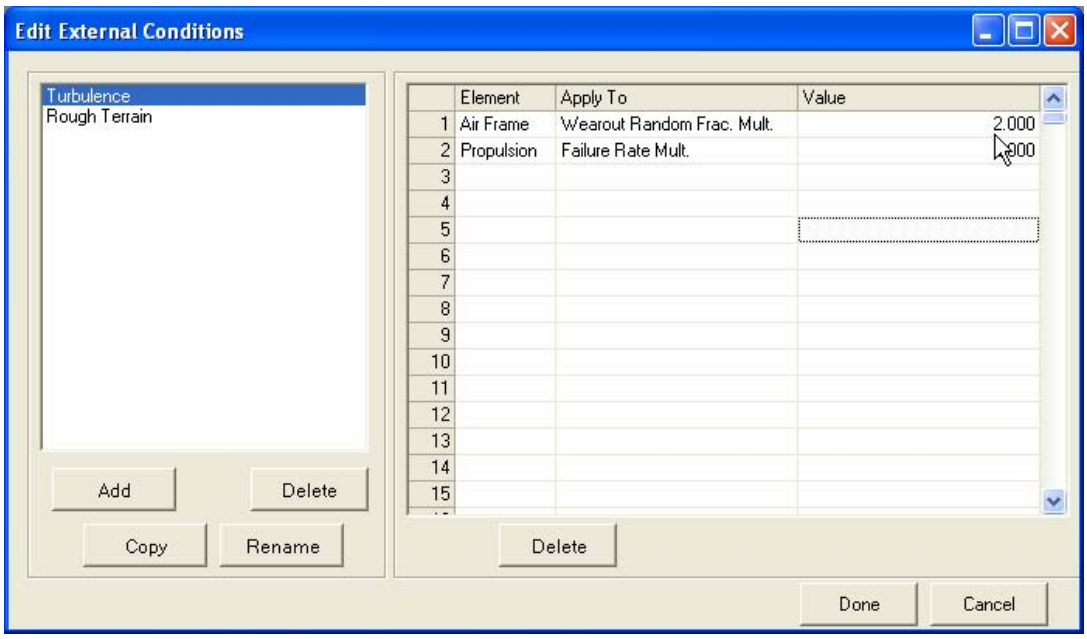

Figure B.15 Editing External Conditions

- *List of External Conditions* shows the external conditions defined thus far. To add a new condition, click the Add button. This brings up the form shown in Figure B.16, where you enter a name for the condition.

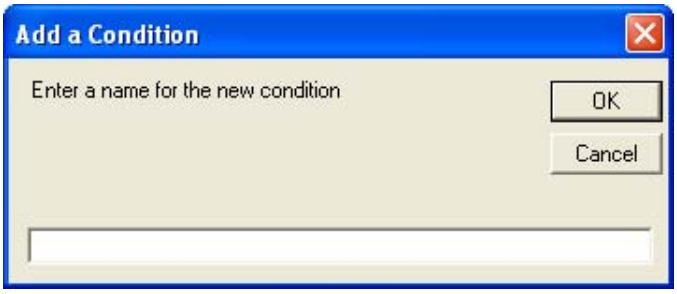

Figure B.16 Entering a New External Condition Name

- *Element* is the list of elements affected by the external condition, or this could be the system itself. To add an element, go to the first blank row. Click on the cell in the Element column and SMO Simulation gives a drop-down list of primary elements, except for the first entry which is "System". Select the affected element or select "System".
- *Apply To and Value* columns define the specific property affected and how it is affected. There are drop-down lists that depend on what was entered under the Element column.

  If you selected "System" under the Element column, the drop-down list of choices in the Apply To column will be:

- Maximum speed. Enter the new maximum speed for the system. Note that the speed of a system is used to determine position of the system and hence the distance traveled. However, the concept of distance traveled as a means of exiting a segment of a scenario may not be supported in future versions of SMO Simulation, so this feature may become obsolete.

- Combat damage rate multiplier. Enter a multiplier that will be applied to the base combat damage rate for the any combat damage model that is attached to the systems being simulated (section B.9).

- Combat damage change. Enter the ID for a new combat damage model (section B.9). Doing the input in this order, these IDs are not yet defined. So, you need to anticipate what you will be entering in section B.9.

If you selected a specific element under the Element column, the drop-down list of choices in the Apply To column will be:

- Failure rate multiplier. This is a valid choice only if the element has an exponential time-to-fail distribution. The constant failure rate will be multiplied by the factor you enter.

- Downtime multiplier. SMO Simulation will multiply the sampled repair time by the factor you enter.

- Wearout mean life multiplier. This is a valid choice only if the element has a wearout time-to-fail distribution. The mean (and standard deviation) of the wearout portion of the distribution will be multiplied by the factor you enter.

- Wearout random fraction multiplier. This is a valid choice only if the element has a wearout time-to-fail distribution. The fraction of failures that are assumed to occur randomly will be multiplied by the factor you enter.

- Aging rate multiplier. This multiplier typically accelerates the aging for the element, thereby causing it to wear out faster. The amount of time an element ages during a time step is multiplied by the factor you enter here.

- State change. This potentially causes the element to change state. The state that the element changes to is selected from the drop-down list in the Value column, which simply contains True and False.

The choices of ways in which external conditions can influence systems will be expanded as SMO Simulation is further developed.

B.5. Scenarios

The form for defining scenarios is shown in Figure B.17. Scenarios are made up of segments that occur in sequence. Each system could have its own scenario, but it is likely that several systems have the same planned scenario.

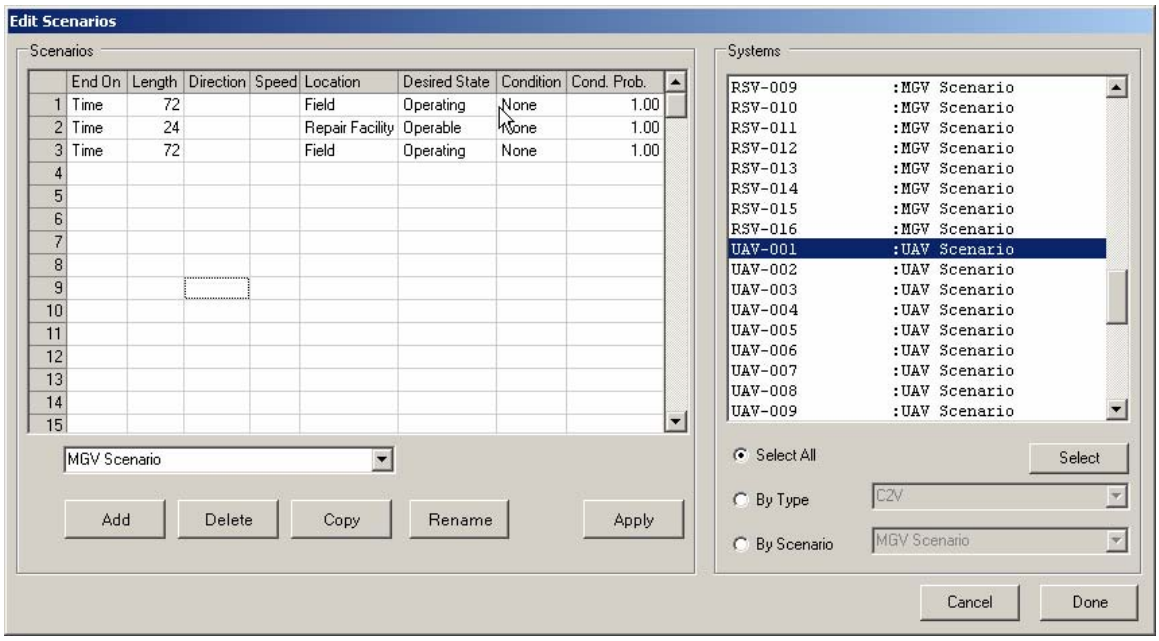

Figure B.17 Editing Scenarios

- *Selected Scenario* appears in the box at the bottom left of the form (Figure B.17). There is a drop-down list of scenarios defined thus far. To add another scenario click the Add button and the window shown in Figure B.18 will appear. Enter a unique scenario name. There can be any number of segments for the scenario. These are edited in the grid that appears above the scenario name.

Figure B.18 Entering a New Scenario Name

- *End On* column has a drop-down list for the 3 ways a segment can end. We currently recommend only using time as the ending criterion. *The other 2 may not be available in future versions of SMO Simulation*.

Distance. When the system travels a given amount of distance, this segment ends for that system and the next segment begins for that system. The next 3 columns must be filled in for this ending criterion. *This option is not recommended in the current version.*

- *Length*. The distance that must be traveled.
- *Direction*. Enter 0 for north, 90 for east, etc.
- *Speed*. Enter the desired speed in kilometers per hour.

Time. When the system has spent the specified time in this segment, it moves to the next segment in this scenario. Of the *Length*, *Direction*, and *Speed* columns, only *Length* (hours) is required.

All States True. The system will exit this segment and enter the next segment only when all element states for the system are True. This can apply when the system is at a repair facility, for example. None of the *Length*, *Direction*, and *Speed* columns are required in this case. *This option is not recommended in the current version.*

- *Location* column has a drop-down list for the 3 locations for the segment: field, repair facility, and other. The system will be at the selected location throughout the segment.
- *Desired State* column has a drop-down list for the 3 system states: operating, operable, and inoperable. Select the one that describes the state you want the system to be in while it in is this segment. For example, for a segment that is specified as *field*, the desired state should be operating. For a segment that is specified as a *repair facility*, the desired state should be operable even though the system might undergo repairs in that segment.
- *Condition* column has a drop-down list for the external conditions you defined (section B.4). The list also contains the word *None*. Select *None* or select the external condition that could apply to this segment of this scenario.
- *Condition Probability* column contains the probability that the specified external condition will occur. This is not required if you selected *None*. Otherwise enter a number between 0 and 1.
- *Systems List* contains all of the systems (section B.2).
- The **Apply** button assigns the displayed scenario to the highlighted system(s). First select the scenario to be assigned then select the system(s). One way to select a system is to scroll the list to the desired system and click on it. Multiple selections can be made by holding down the Ctrl key (or shift key for a block of systems) while clicking on the system IDs.

The controls below the systems list give you additional ways to select the systems to be assigned. These appear as a group multiple times throughout the input. For each occurrence you will be referred back to here for the description.

- To select all of the systems, choose the **Select All** option then click the **Select** button.
- To select all systems of a given system type (section B.2.1), choose the **By Type** option and a drop-down list of system types will appear. Choose one system type and then click the **Select** button to select all systems of that type.
- To select all systems that have been assigned a particular scenario, choose the **By Scenario** option and a drop-down list of scenario names will appear. Choose a scenario and click the **Select** button, and all systems that have been assigned that scenario will become highlighted. If you then click the **Apply** button, all of these systems will be assigned the chosen scenario.

B.6. Supplies and Services

This menu item allows the user to set up spares, consumables and maintenance services. It also provides access to an input form for defining the supply chain or network through which these supplies and services can be accessed.

B.6.1. Spares Inventories

The form for defining spares has three tabs. The first tab is used for defining all possible spares for the mission. The second tab is used to aggregate the spares into kits or inventories. The final tab is used to assign kits to systems.

B.6.1.1. Spares Tab

The Spares tab is shown in Figure B.19. Enter all spares that will be available for the mission.

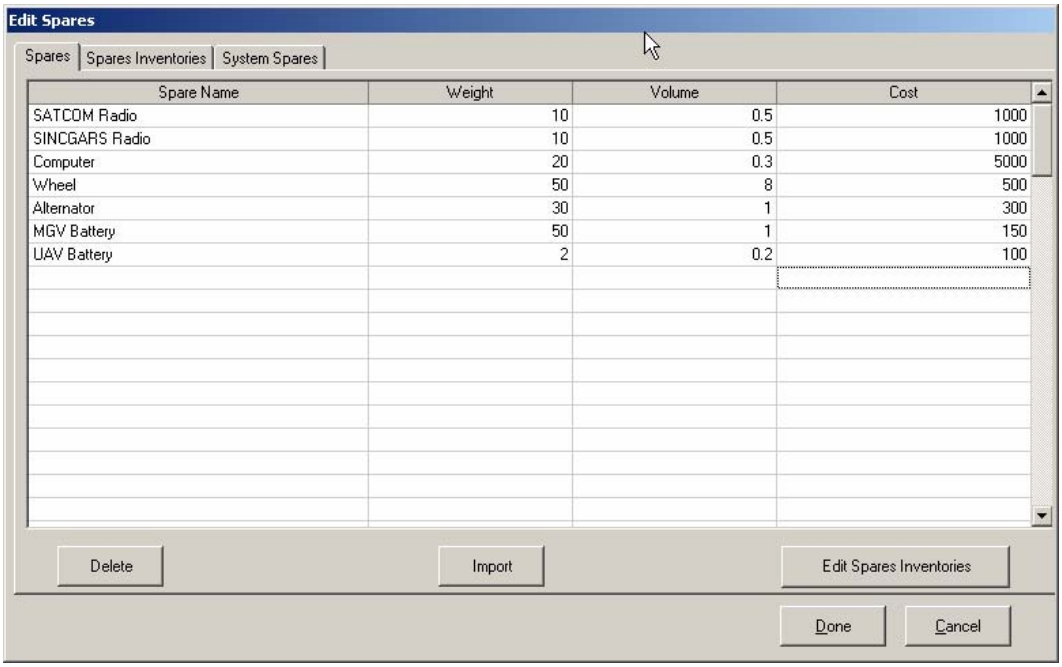

Figure B.19 Defining Spares

- *Spare Name* is the unique name for the spare. To add a new name click on the first blank cell in this column and enter the name.
- *Weight* is the weight of one spare. SMO Simulation does not require specific units here but they should be consistent across all spares and consistent with weight units for consumables.
- *Volume* is the volume of one spare. SMO Simulation does not require specific units here but they should be consistent across all spares and consistent with volume units for consumables.
- *Cost* is the cost of one spare.

B.6.1.2. Spares Inventories Tab

The Spares Inventories tab is shown in Figure B.20. This form is accessed by clicking the **Edit Spares Inventories** button on the Spares tab (Figure B.19). It is anticipated that spares will typically be carried in kits or inventories. To access a spare, you have to access the kit that contains it. So, every spare in the kit will have the same access time. This does not preclude having a spare part carried independently, that is, a kit can contain a single spare.

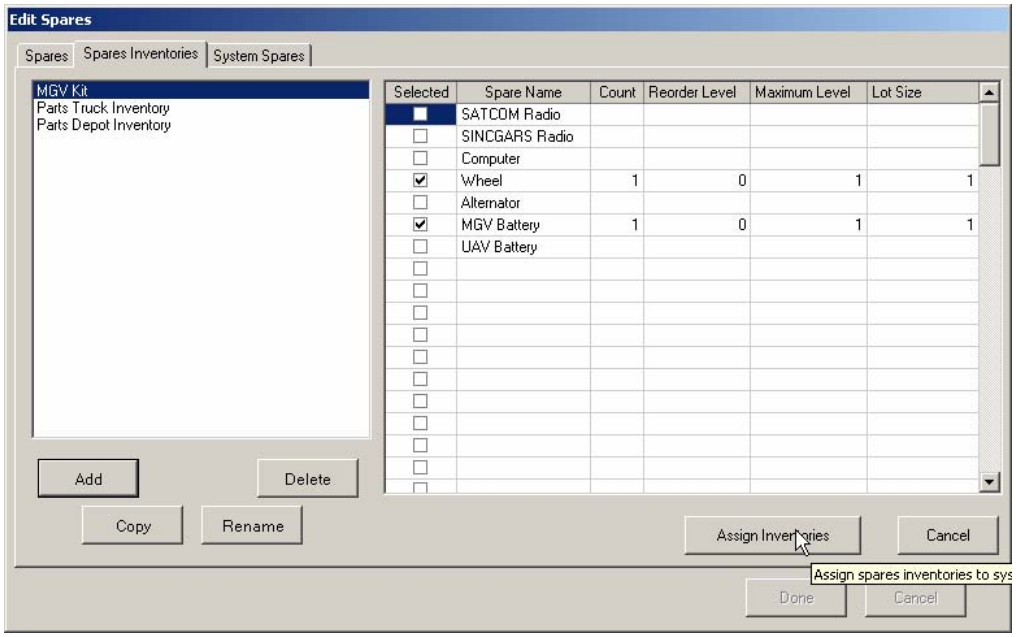

Figure B.20 Defining Spares Inventories

- *Spares Inventories List* contains the unique names for each spares kit. To add a new name click on the Add button. Enter the name onto the form shown in Figure B.21. The contents of the grids on the right side of Figure B.20 apply to the selected spares kit in this list.

Figure B.21 Adding Spares Inventory Name

- *Spare Name* column contains all of the spares defined on the Spares tab (Figure B.19). SMO Simulation fills in this column and it cannot be edited.
- *Selected* is checked if the spare belongs in the highlighted kit.
- *Count* is the normal number of spares in the kit.
- *Reorder Level* is the level at which an order is placed to replenish the inventory.
- *Maximum Level* is the maximum number of a spare that the inventory will accept. This allows the inventory to temporarily go above the desired level if necessary to accommodate incoming parts orders.
- *Lot Size* is the number of spares ordered at a time.

B.6.1.3. System Spares Tab

The System Spares tab is shown in Figure B.22. This form is accessed by clicking the **Assign Inventories** button on the Spares Inventories tab (Figure B.20). Each system can carry at most one kit or inventory. So if there is a system that carries several spares kits, you must combine those into a single inventory on the Spares Inventories tab, and assign the inventory to the carrying vehicle here.

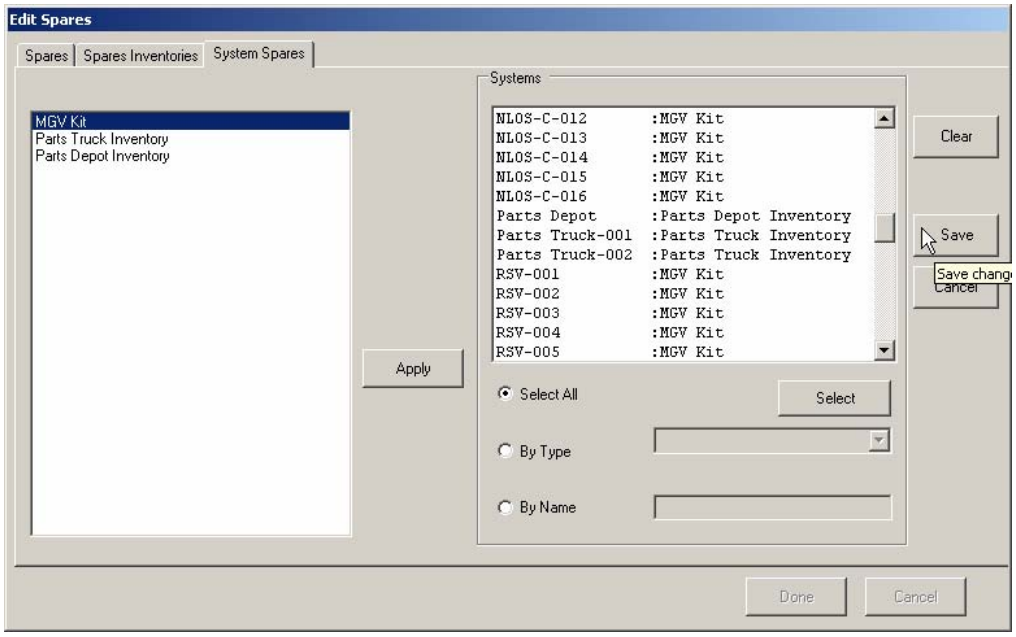

Figure B.22 Assigning Spares Inventories to Systems

- *Spares Inventories List* contains the unique names for each spares kit. You defined these on the Spare Inventories tab (section B.6.1.2) and the list is not editable here.
- *Systems List* contains all of the systems (section B.2.1).
- The **Apply** button assigns the highlighted kit to the highlighted system(s). First select the kit to be assigned then select the system(s) to which the kit belongs. One way to select a system is to scroll the list to the desired system and click on it. Multiple selections can be made by holding down the Ctrl key (or shift key for a block of systems) while clicking on the system IDs.
- The **Select** button gives you additional ways to select the systems to be assigned (see section B.5).

B.6.2. Consumables

The form for defining consumables has three tabs that parallel spares input. The first tab is used for defining all possible consumables for the mission. The second tab is used to aggregate the individual consumables into inventories. The final tab is used to assign consumables inventories to systems that will act as suppliers. Recall that consumable use

is defined for each system is defined on the third tab of the systems input form (Figure B.10).

B.6.2.1. Supplies Tab

The Supplies tab is shown in Figure B.23.

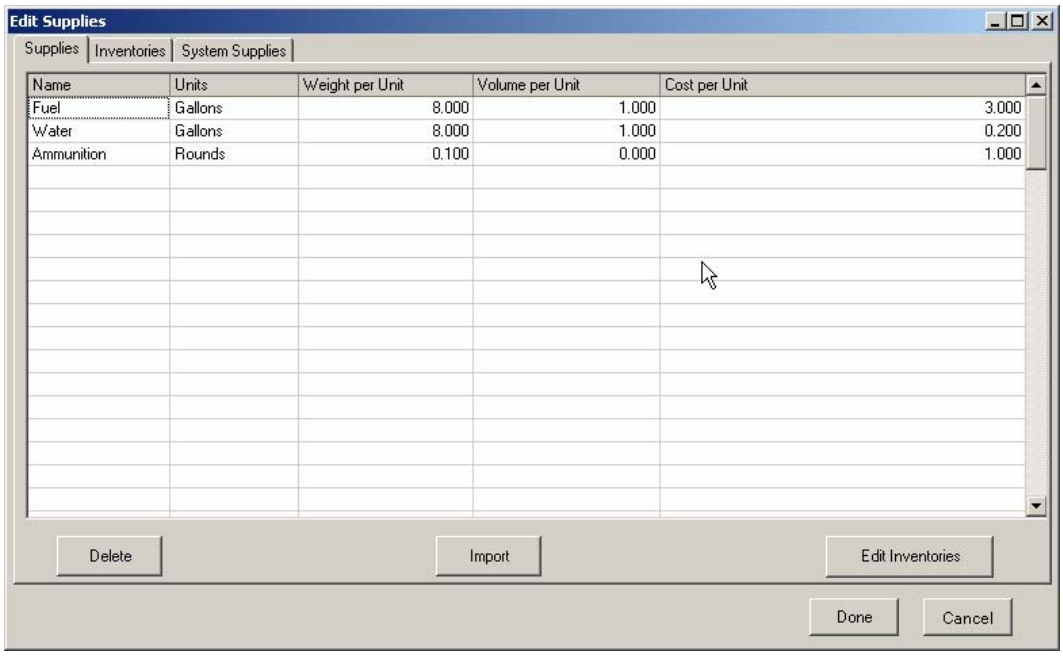

Figure B.23 Supplies Tab for Consumables Input

The columns in the Supplies tab grid contain the following information:

- *Name* is the name of the consumable that will be used throughout the simulation.
- *Units* is the name of the units in which the supply will be consumed and replenished. This allows a consumable to be simulated in units other than the primary weight and volume units used in the analysis.
- *Weight per Unit* allows the consumable quantity to be converted to the primary weight units of the analysis.
- *Volume per Unit* allows the consumable quantity to be converted to the primary volume units of the analysis.
- *Cost per Unit* is the cost associated with a unit of the consumable.

The **Import** button allows information on consumables to be imported from a text file.

B.6.2.2. Inventories Tab

The Supply Inventories tab is shown in Figure B.24. This form is accessed by clicking the **Edit Inventories** button on the Supplies tab (Figure B.23). An inventory can contain

one or more consumables. This tab defines the quantity and other parameters for each consumable in an inventory. The list on the left side of the form shows all current consumables inventories. To create an inventory click the Add button and enter a name for the inventory. All consumables are listed in the grid on the right side of the form. The values in the grid represent the selected inventory in the list on the left side.

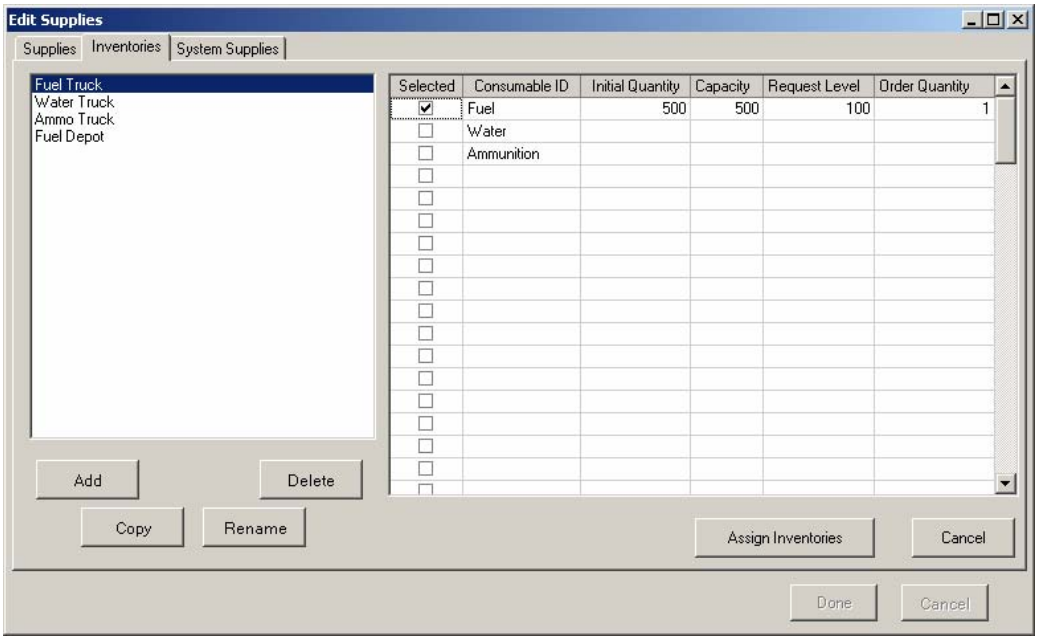

Figure B.24 Supplies Inventory Input

- The *Consumable ID* column contains all of the consumables defined on the Supplies tab (Figure B.23). SMO Simulation fills in this column and it cannot be edited.
- *Selected* is checked if the consumable belongs in the highlighted inventory.
- *Initial Quantity* is the initial amount of the consumable in the units specified for it in Figure B.23. The same units are required for the next two items.
- *Capacity* is the maximum amount of the consumable that the inventory can hold.
- *Request Level* is the level at which an order is placed to replenish the inventory.
- *Order Quantity* is similar to the lot size for spares. For example, if water is ordered in 10 gallon containers, the order quantity would be 10.

B.6.2.3. System Supplies Tab

The System Supplies tab is shown in Figure B.25. This form is accessed by clicking the **Assign Inventories** button on the Inventories tab (Figure B.24).

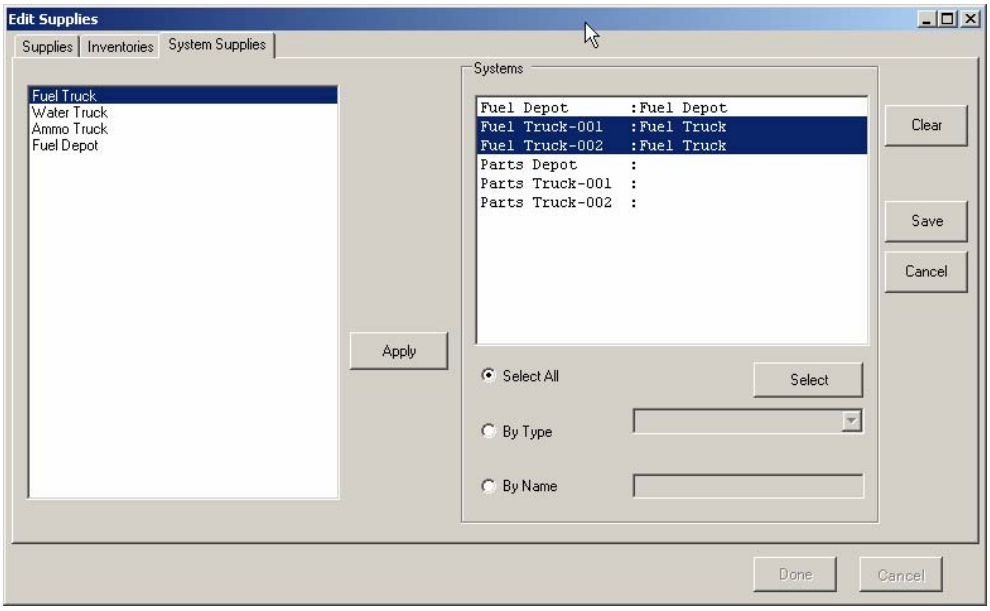

Figure B.25 Assigning Consumables Inventories to Systems

- *Consumables Inventories List* contains the unique names for each consumables inventory. You defined these on the Inventories tab (section B.6.2.2) and the list is not editable here.
- *Systems List* contains all of the systems that have been identified as supply vehicles (section B.2.1).
- The **Apply** button assigns the highlighted kit to the highlighted system(s). First select the inventory to be assigned then select the system(s) to which the inventory belongs. One way to select a system is to scroll the list to the desired system and click on it. Multiple selections can be made by holding down the Ctrl key (or shift key for a block of systems) while clicking on the system IDs.
- The **Select** button gives you additional ways to select the systems to be assigned (see section B.5).

B.6.3. Services

Maintenance resources are services that may be required to repair a failed primary element. Such services may be provided by the crew of the system being repaired or may be provided by another system that is equipped for the service. The service may be the actual element repair in which case the time to perform the service is the repair time. Or, a service may required in order to allow the repair to take place; e.g., towing. The form for editing services has four tabs:

6. A tab to define basic services,
7. A tab for grouping basic services into combinations that may be required by systems for repair (user services),
8. A tab for combining basic services for assignment to service providers, and

9. A tab to assign provider services to systems.

B.6.3.1. Basic Services Tab

This tab (Figure B.26) allows you to define the individual basic services that will be used throughout the simulation.

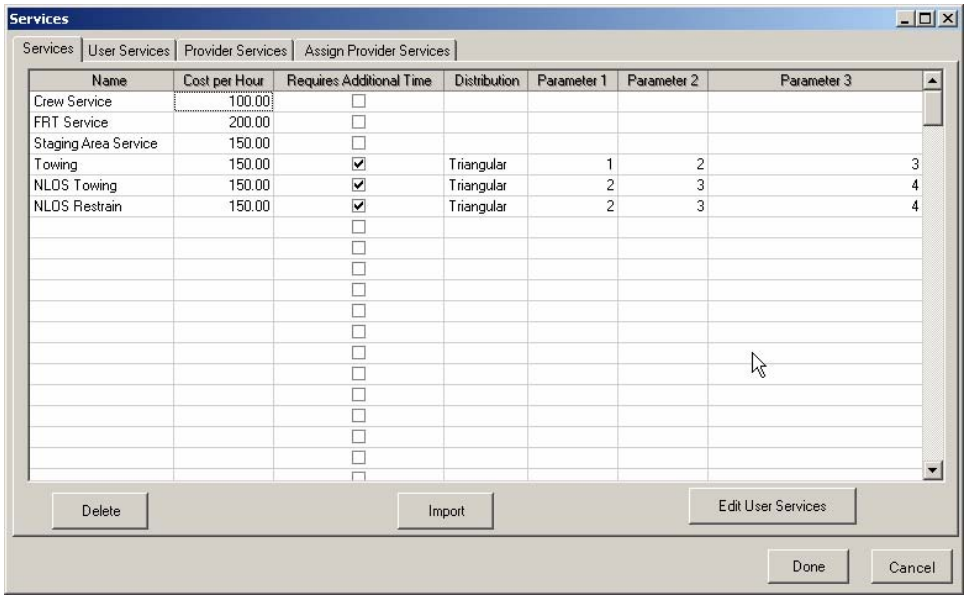

Figure B.26 Input Tab for Basic Services

The columns in the services input grid contain the following information:

- *Name* is the name of the basic service. To add a new service, first enter its name on the first nonblank row.
- *Cost per Hour* is the cost of providing the service.
- *Requires Additional Time* is a Boolean property that indicates whether the service requires time other than the normal repair time of the element. If this column is not checked, the service is assumed to be the actual element repair and the time required for the service is the repair time for the element. If the column is checked, the service is assumed to require time in addition to the element repair time.
- *Distribution* and *Distribution Parameters* columns define the uncertainty in the time required to perform the service. This input is only needed if the Requires Additional Time column is checked. The available distributions and the definitions of their parameters can be found in section B.2.2.

B.6.3.2. User Services Tab

The tab for defining user services is shown in Figure B.27. User services are required because more than one basic service may be required to return a failed element to a useful condition. An example might be a vehicle with a failed engine that requires towing to an

area where engine repairs can be performed. The list on the left side of the form contains all defined user services. To create a new user service, click the **Add** button and enter a unique name for the service. The information in the grid on the right side of the form reflects the currently selected user service.

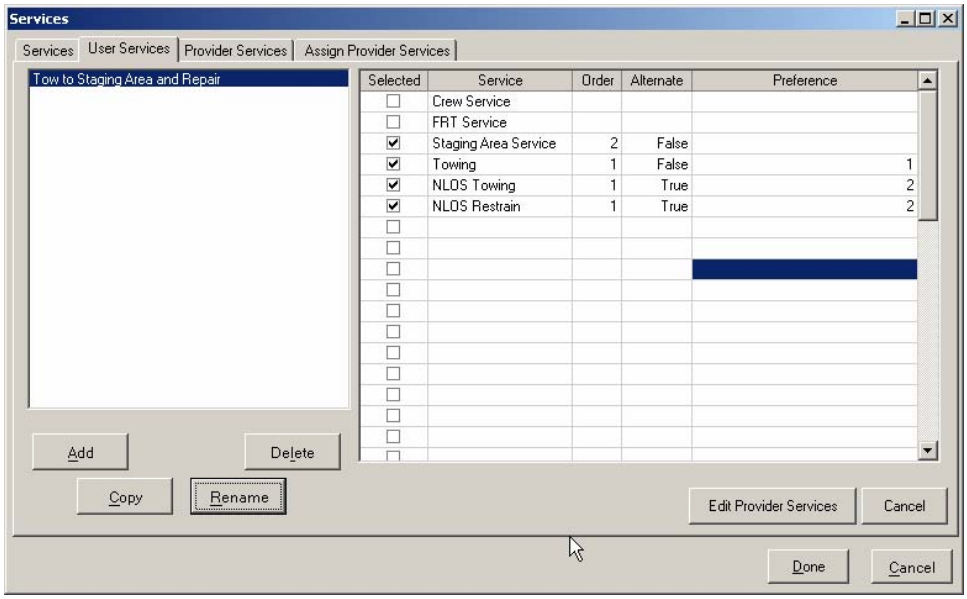

Figure B.27 User Services Input Tab

The columns in the user services input grid contain the following information:

- The *Service* column contains names of all the basic services defined in the Services Tab (Figure B.26). This column is not editable.
- The *Selected* column indicates whether basic service is included in the selected user-service grouping.
- The *Order* column indicates the order in which basic services will be performed. In Figure B.27, towing would be performed prior to staging area service.
- *Alternate* indicates whether the basic service is an alternative to another basic service. In Figure B.27, NLOS Towing and NLOS Restrain are alternates to Towing. The interpretation is that the user service first tows the failed system to a staging area where it is repaired. If a tow vehicle is not available, an alternative is to use two NLOS cannons, one in front to pull and one behind to restrain.
- *Preference* must be included for services that have, or are, alternatives to indicate which alternative should be sought first.

B.6.3.3. Provider Services Tab

Provider services parallel spares or consumables inventories. They represent a collection of basic services that can be assigned to a system that will then be able to provide any of the services to a system that requires service. The Provider Services tab is shown in Figure B.28 and can be reached by clicking the **Edit Provider Services** button on the

User Services tab. The list on the left side of the form contains all provider services and the grid on the right shows the basic services that are included in the selected provider service. To create a provider service, click the Add button then enter a unique name for the provider service.

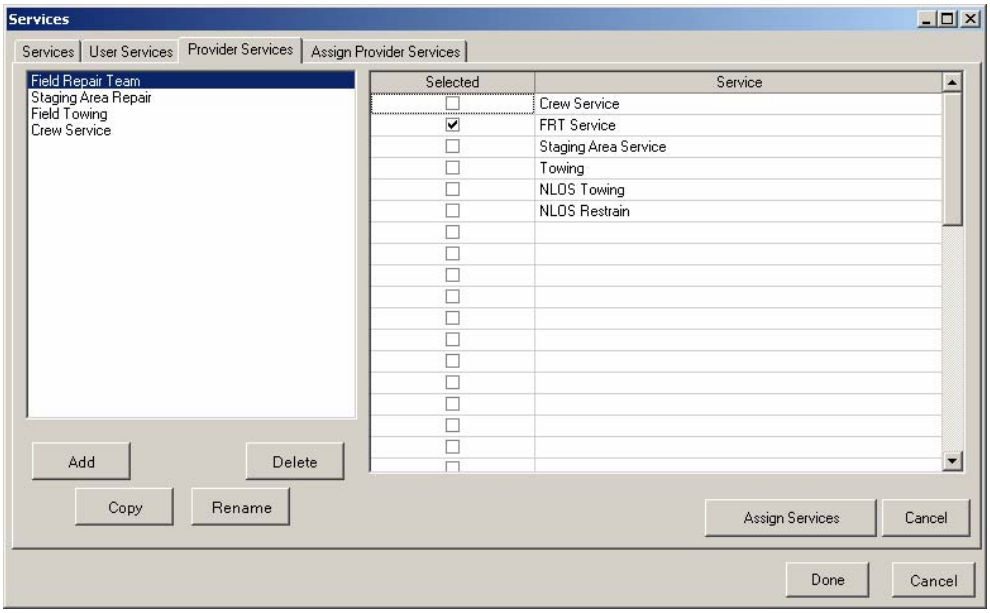

Figure B.28 Provider Services Input Tab

The Service column in the provider services grid contains all the basic services that were defined in the Services tab (Figure B.26) and it is not editable. The Selected column indicates whether a basic service is included in the currently selected provider service.

B.6.3.4. Assign Provider Services Tab

The Assign Provider Services tab is shown in Figure B.29. You can reach this form by clicking the **Assign Services** button on the Provider Services tab (Figure B.28). The left side of the form displays a list of all provider services defined on the previous tab. To assign a provider service to one or more systems, select the provider service on the left side of the form, select the desired systems on the right side of the form, then click the **Apply** button.

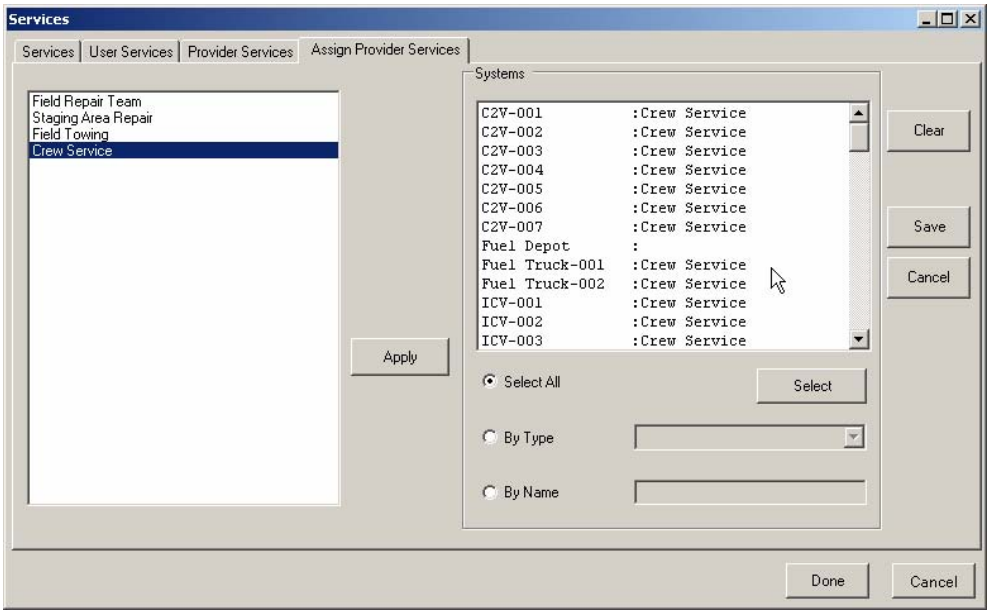

Figure B.29 Assigning Provider Services to Systems

The select controls below the list of systems on the right side of the form provide a short-cut means for selecting groups of systems (see section B.5).

B.6.4. Supply Connections

Supply connections provide the information needed to allow systems to access supplies and maintenance services. Supply connections establish a relationship between users and providers of supplies and services. Figure B.30 shows the supply connections input form.

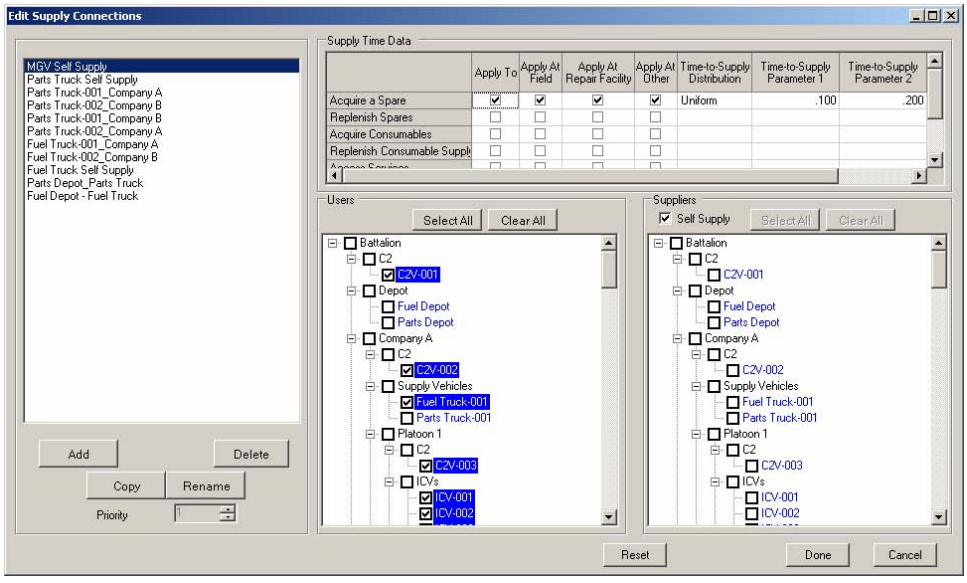

Figure B.30 Input Form for Supply Connections

To create a supply connection click the **Add** button and enter a unique name for the connection. The list on the left side shows all the supply connections. Properties of the selected connection are displayed and can be edited in the grid at the top right of the form and the two tree controls under the grid. A connection can be specified to allow any of the following actions:

- *Acquire a spare* – the connection can be used to obtain a spare when needed for a repair.
- *Replenish spares* – the connection can be used to replenish a spares inventory or kit when the numbers of one or more spares falls below their reorder levels.
- *Acquire consumables* – the connection can be used to obtain a consumable when the amount remaining for a user falls below the request level.
- *Replenish consumables supply* – the connection can be used to replenish a consumable for a supplier whose supply has fallen below the reorder level.
- *Access services* – the connection can be used by a system that needs repair to access a maintenance service.

The time for any of these actions to occur via the connection is determined by an uncertainty distribution as specified in the time-to-supply distribution shown in the Supply Time Data grid. The available distribution types and the definitions of their parameters can be found in section B.2.2.

The tree control labeled Users in Figure B.30 identifies the systems that can use the selected connection to acquire needed supplies or services. The tree control labeled Suppliers identifies the systems that will provide the supplies or services when the connection is used. In figure B.30, the Self Supply option is checked meaning that the MGV Self Supply connection involves the User systems providing spare parts for their own needed repairs. The Supply Time Data grid shows that the user systems can self-supply spare parts for repairs at any location (field, repair facility or other) and the time required is uniformly distributed between 0.1 and 0.2 hours.

Figure B.31 shows the supply connections input form with the Parts Truck-001 of Company A connection selected. In this case, the supplier is Parts Truck-001. Users of the connection include all manned ground vehicles in Company A plus the Battalion level C2V-001. The connection can be used at any location to acquire a spare that is needed for a repair and the required time is a triangular distribution with parameters 2, 3, and 5 hours. The connection can also be used to replenish a spares kit or inventory but only when the user is at a repair facility. In this case, the time required also follows a triangular distribution with parameters 0.5, 1.0, and 2.0 hours. Note that this connection has priority 2. Any time a supply connection is used, self-supply is preferred although self-supply is probably limited to a few parts. When self-supply is not available, the connection with the best priority and the lowest time to provide the supply or service is chosen.

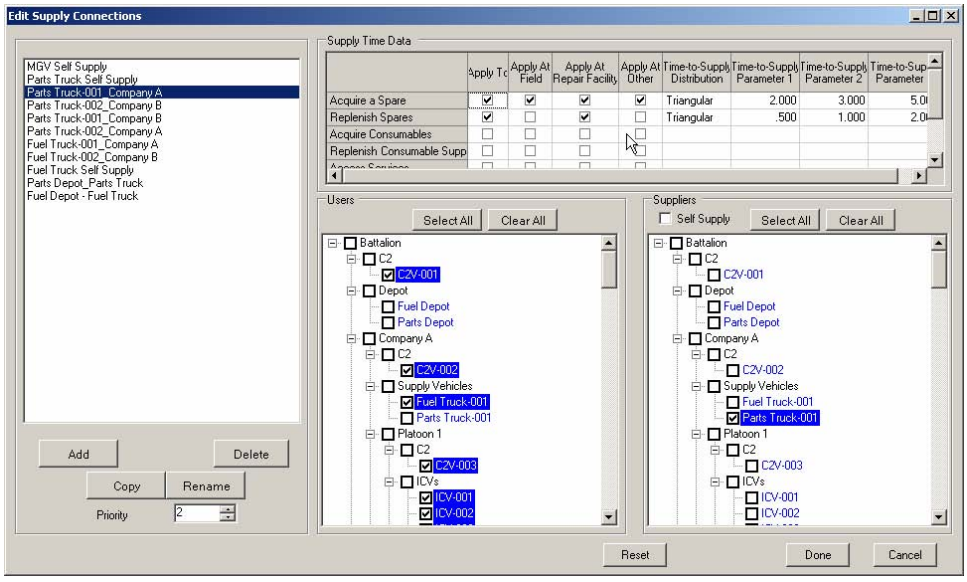

Figure B.31 Input Form for Supply Connections

B.7. Structure

This input form allows you to create a hierarchical structure for the systems in the simulation. The form has two tab pages. The first page allows you to create the hierarchical structure for the simulation. The second tab page allows the systems of the simulation to be placed within the structure. Figure B.32 shows the Structure tab of the form.

B.7.1 Structure Tab

For new files, the structure will have only one node labeled Top. In Figure B.33 we have renamed the Top node to Battalion and added a node under it. To rename a node, select the node and click the **Rename** button to display a text entry form for the new name. To add a node, select a node under which you wish to add a child node then click the **Add** button. When the text input form appears, enter a name for the new node. The buttons to the right of the hierarchy tree allow nodes to be added, deleted, copied, pasted, and renamed. The underlying rule for uniqueness in the tree node naming is that the full path to any node must be unique. The two arrow buttons in the lower right of the form allow you to move nodes up or down in the structure. This is useful for arranging the structure. Figure B.34 shows a completed hierarchy.

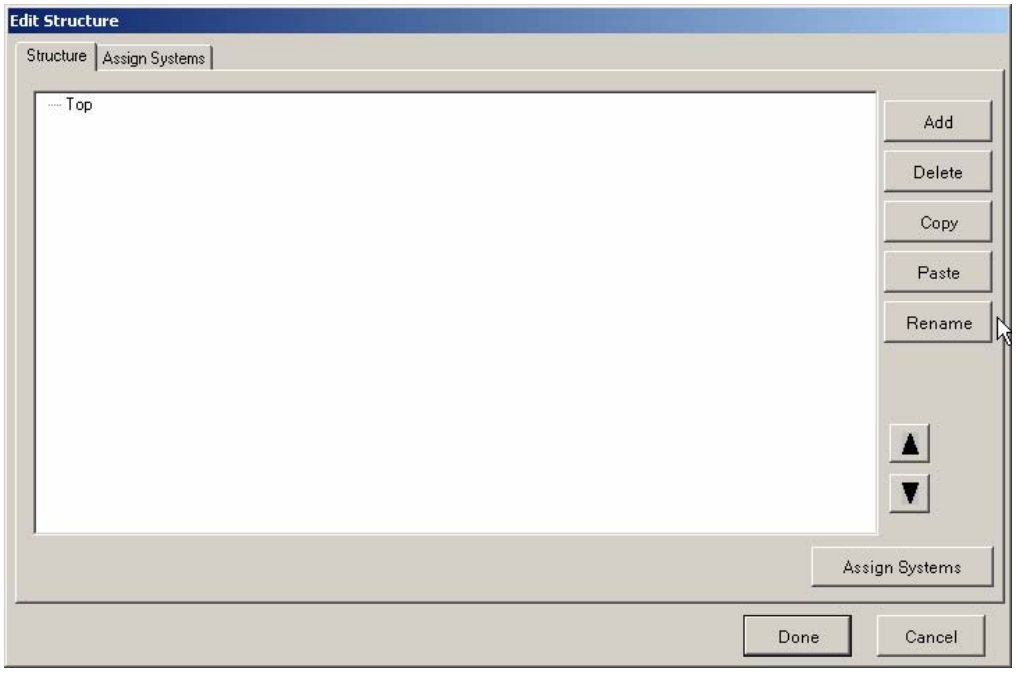

Figure B.32 Input form for Simulation Hierarchy

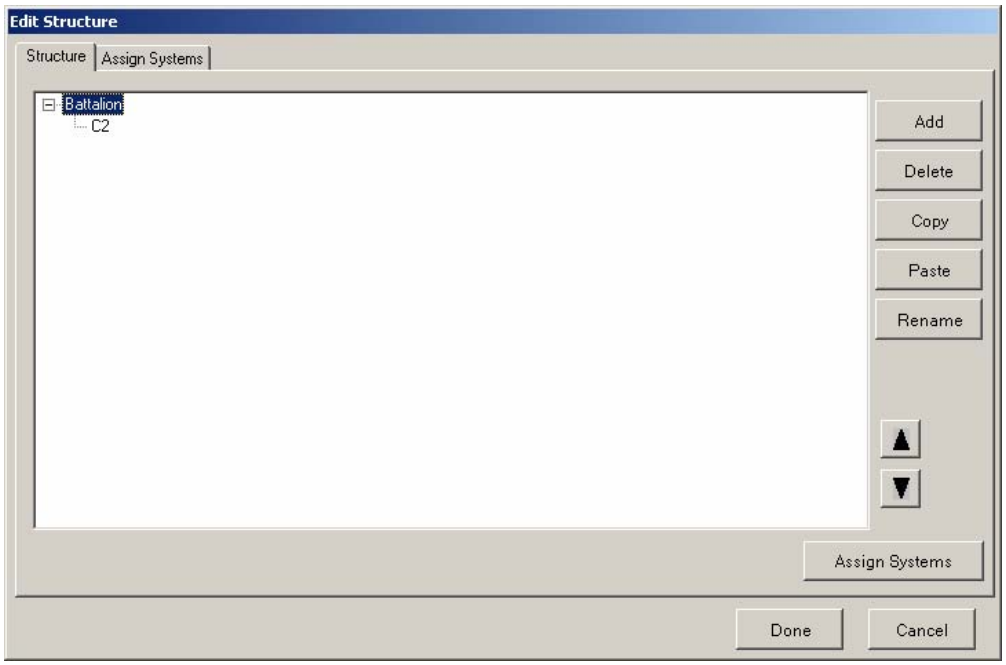

Figure B.33 Simulation Hierarchy with First Two Nodes

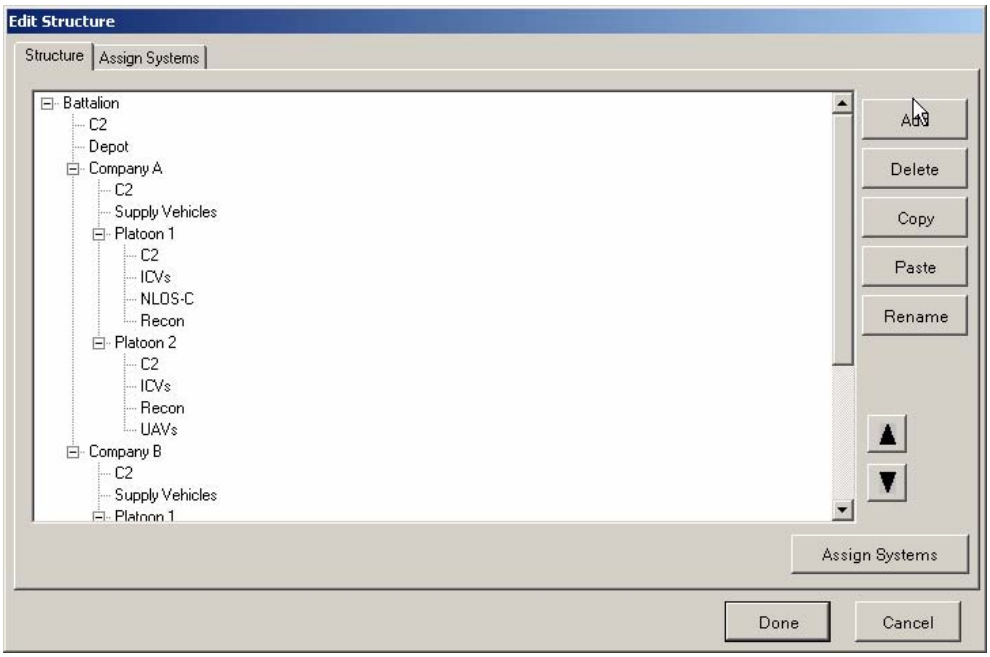

Figure B.34 Completed Simulation Hierarchy

Notice that in Figure B.34 we have built a force structure that is a battalion with two companies (Company A and Company B). Each company has two platoons (Platoon 1 and Platoon 2). Within each level of the structure we have included a node for platform types. For example, the battalion will have a C2V and two depots (fuel and parts). Platoon 1 in Company A will have C2Vs, ICVs, NLOS-Cs and Reconnaissance vehicles. Putting each platform type under its own node provides performance measures by platform type in the statistical results.

B.7.2. Assign Systems Tab

The Assign Systems tab allows you to assign the systems in the simulation to their intended place in the hierarchy. Figure B.35 shows the Assign Systems tab. You can access this tab page by clicking the **Assign Systems** button on the Structure tab (Figure B.34). When you first visit the Assign Systems tab, all systems of the simulation will be listed on the left side and the structure is shown on the right. To assign systems to the structure, select one or more systems from the list on the left and select a node under which they will be assigned on the right. Then click the >> button to move systems from the list to the structure. Figure B.36 shows the result. If you made an incorrect assignment, select the system in the structure tree and click the << button to move it back to the list. The structure with all systems assigned is shown in Figure B.37.

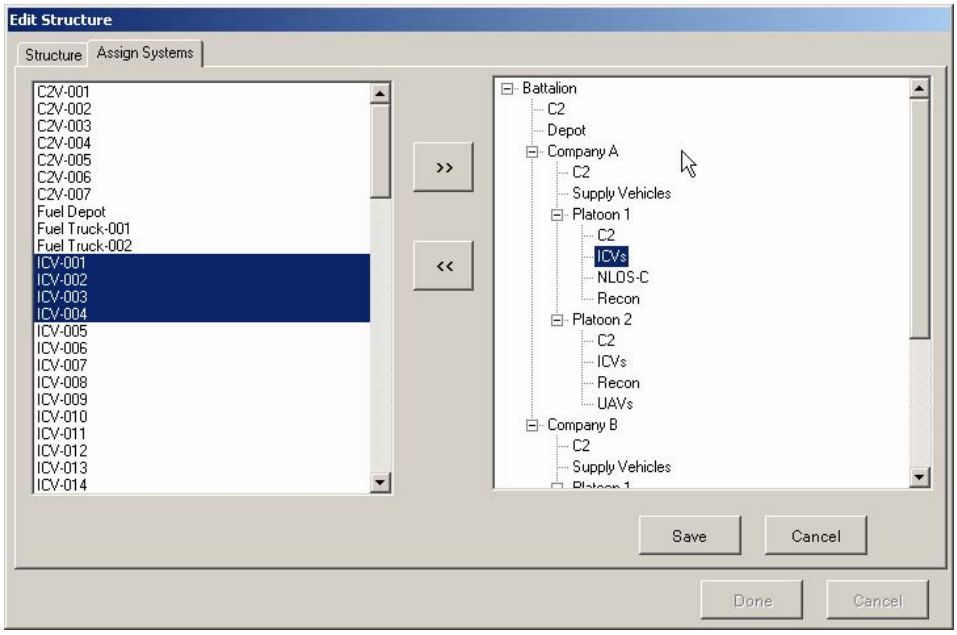

Figure B.35 Tab Page to Assign Systems to Hierarchy

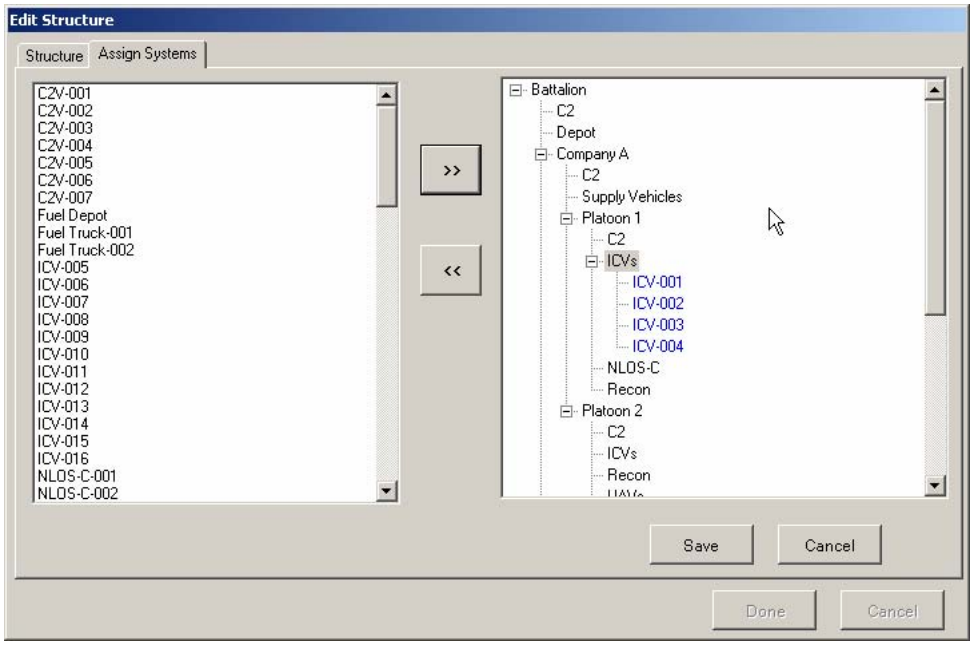

Figure B.36 ICVs Assigned to the Structure

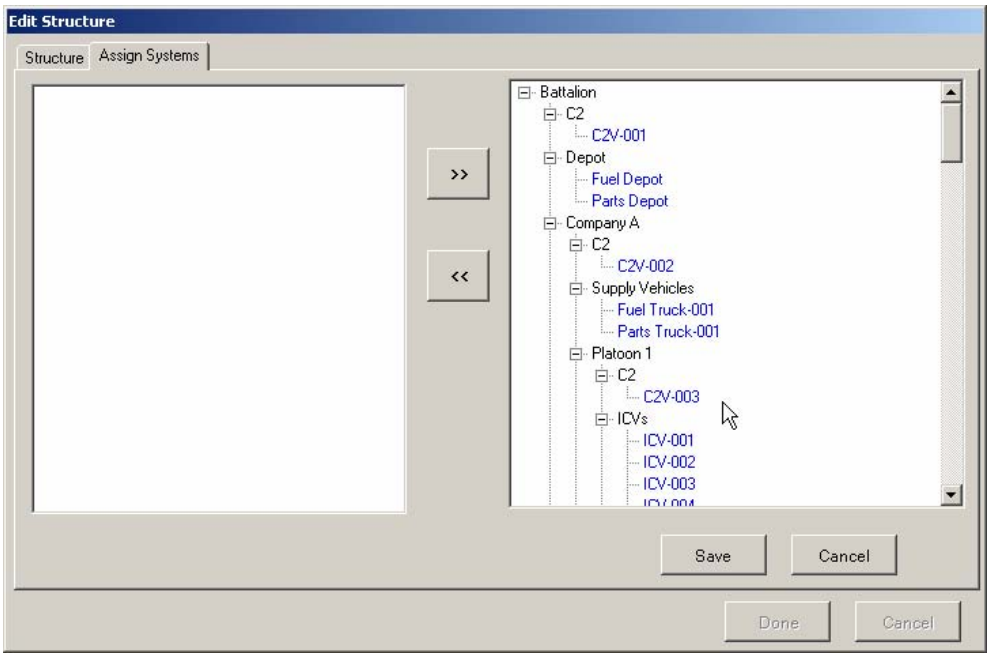

Figure B.37 Structure with All Systems Assigned

B.8. Other Elements

In addition to primary elements and consumables, other element states can affect the states of a system and its functions. Two additional categories of states are external and reference. External states represent elements that are external to the systems that can affect the performance of the systems. An example might be a sandstorm or a change in terrain. A reference state provides a means for expressing the dependence of a system on the functionality of another system.

B.8.1. External Elements

The form for editing external states is shown in Figure B.38. The input is simple and consists of the following:

- *Name* is the name of the external element.
- *Initial State* specifies whether the initial state of the element is true or false.
- *Desired State* is the state that is desired for the external element. Recall that external elements, once defined, can be included for any system (Figure B.10) and thereby become available for inclusion in the failure and success equations for any function of the system (section B.3). As a term in a failure or success equation, an external element returns True when it is in the desired state and False when it is not.

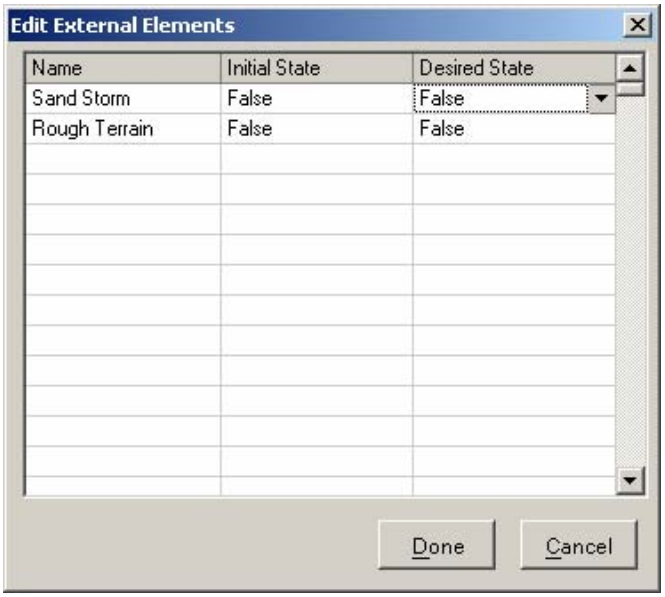

Figure B.38 Form for Editing External Elements

B.8.2. Reference Elements

References provide a means for establishing dependencies between systems. Figure B.39 shows the form for editing reference elements. The tree control on the left side of the form shows systems organized within the simulation hierarchy. On the right side of the form, a tree control shows the same systems including their functions. To create a reference, first select the system that will add the reference to its elements then click the **Add** button on the lower left side of the form. When the text input form appears (Figure B.40) enter the desired name for the reference.

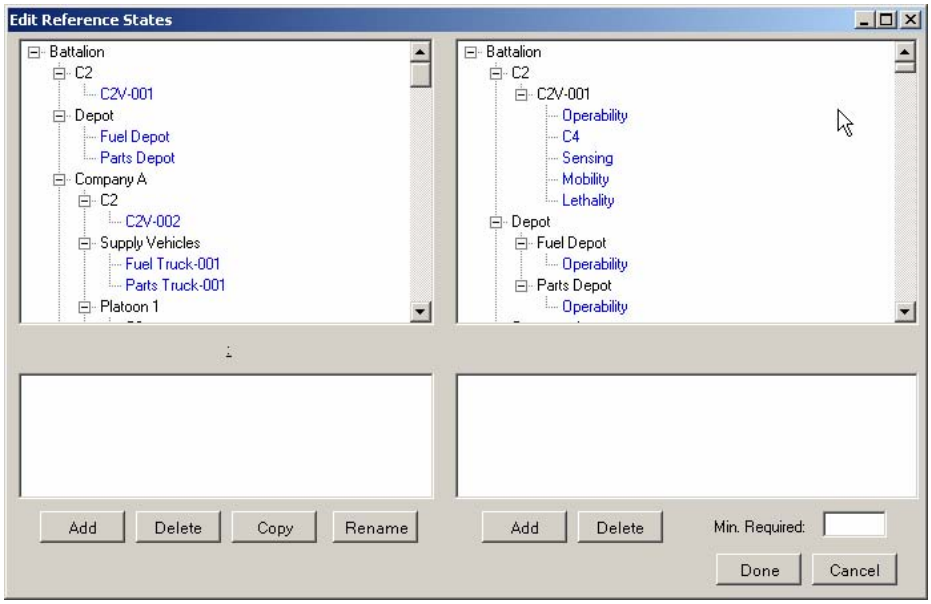

Figure B.39 Form for Editing References

Figure B.40 Adding a Reference

Once the reference has been created the form should show the new reference and the system that will use it as shown in Figure B.41. Now the actual references can be added by selecting the function that will be referenced from the tree control on the right. Once the desired system and function have been selected, click the **Add** button.

In Figure B.42 we have referenced the sensing function for UAV-001 and UAV-002 in Company A, Platoon 2. Note from the lower right hand side that the minimum requirement is for one of the two functions. The result of this exercise is that Remote Sensing is now an element for NLOS-C-001. If, for example, we include remote sensing in the failure equation for lethality for NLOS-C-001, lethality will fail if the sensing function for both the UAVs fails. As long as either of the two UAVs has its remote sensing function operable, lethality of the NLOS-C-001 will not be affected by its dependence on remote sensing.

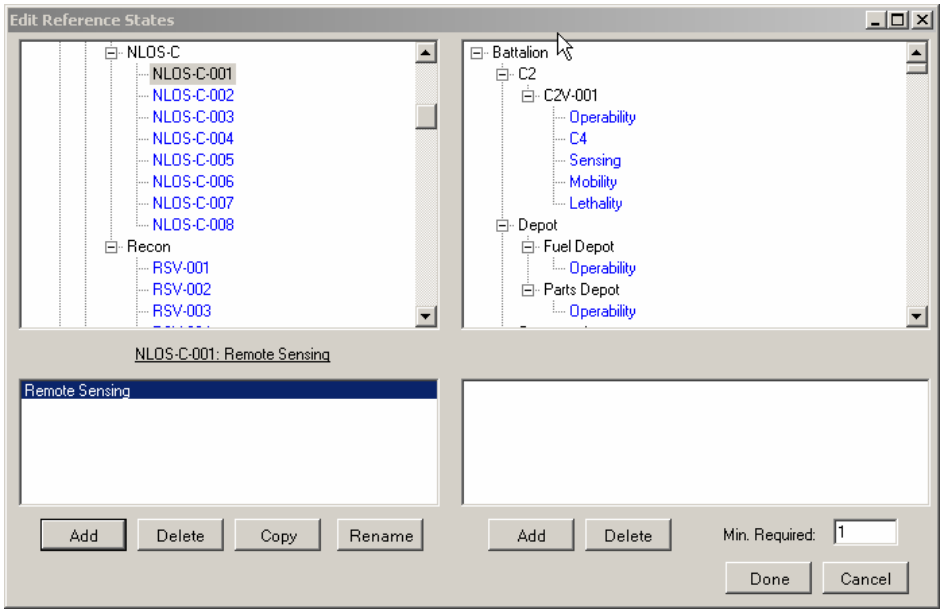

Figure B.41 Adding a Reference

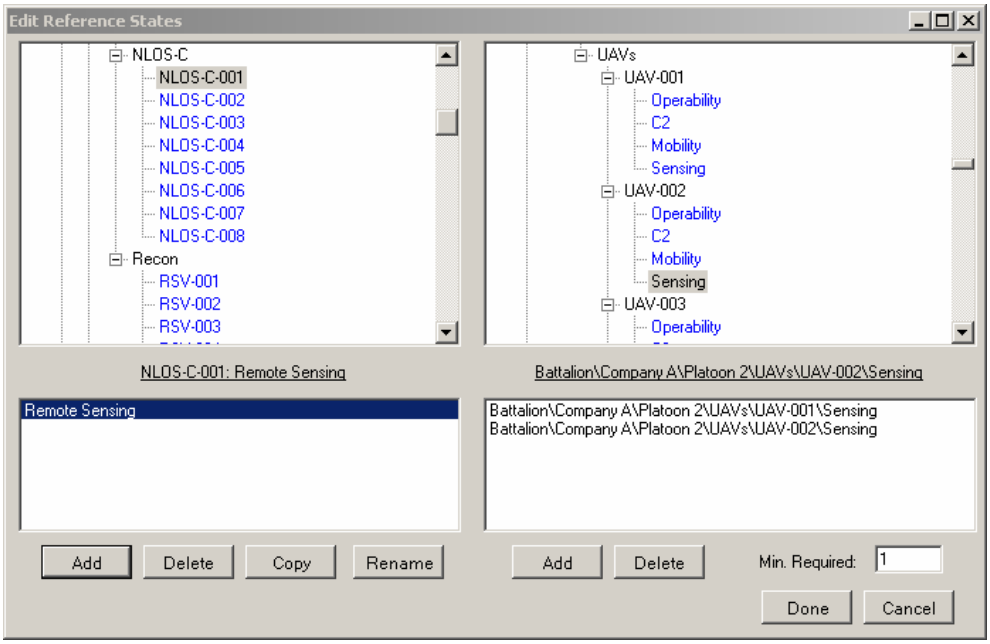

Figure B.42 The Remote Sensing Reference for NLOS-C-001

B.9. Combat Damage

The form for editing combat damage is shown in Figure B.43. The form includes two tabs – one for creating combat damage definitions and the second for assigning combat damage definitions to systems. On the first tab, the left side of the editing form (Figure B.43) will display a list of all combat damage definitions. The right side will display the currently selected combat damage definition in tree form.

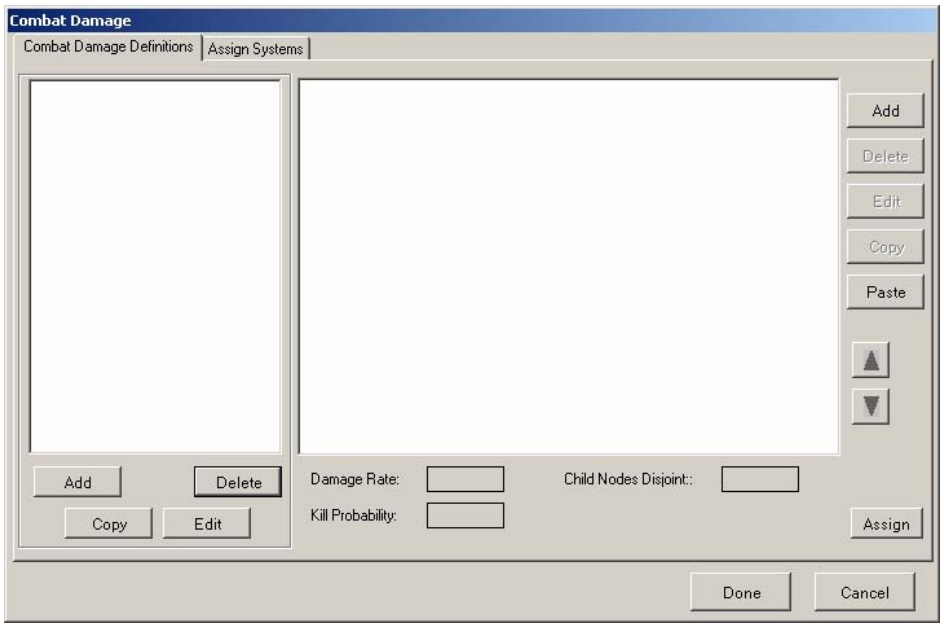

Figure B.43 Form for Editing Combat Damage Input

B.9.1. Combat Damage Definitions

Combat damage definitions allow SMO Simulation to answer the following questions:

1. Was there any combat damage to this system on this time step?
2. If so, was the system totally disabled?
3. If damaged but not disabled, which elements of the system were damaged to the point that they require replacement?

Combat damage is represented using a tree structure. The tree has a combat damage rate which is used by SMO Simulation (assuming a Poisson process) to answer the first question. Each branch of the tree can have a nonzero "kill" probability that may be used to answer the second question. The leaves, or ending branches, of the tree are elements. If the branch probabilities lead to the leaf for Element A, then SMO Simulation assumes that Element A is damaged and requires replacement.

To create a new combat damage definition, click the **Add** button on the lower left side of the form to display the form shown in Figure B.44.

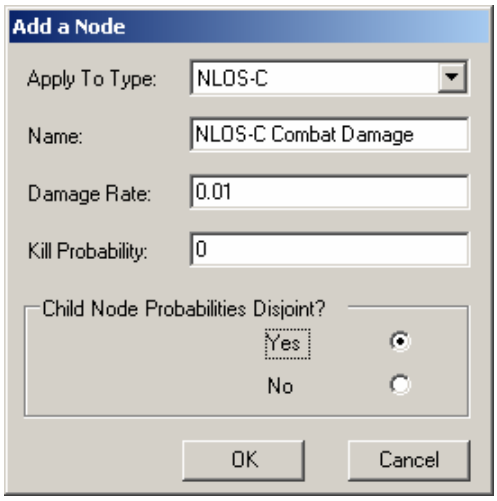

Figure B.44 Creating a New Combat Damage Definition

The form for adding a new combat damage definition (Figure B.44) requires the following input:

- *Apply To Type*. Because the tree is normally tailored to a system, first select the system type to which this combat damage model will apply. SMO Simulation provides a drop-down list of system types for convenience.
- *Name*. SMO Simulation makes the name for the top node in the tree the same as the name of the combat damage model. Enter a meaningful name here.
- *Damage Rate*. Only the top branch of the tree has a damage rate. For other nodes in the tree SMO Simulation wants the probability of the branch. The damage rate

is the probability per hour that combat damage will occur when the definition is active.

- *Kill Probability*. The combat damage model could be as simple as "kill or no kill". Suppose that you do not enter any other branches to the model. Then on a time step if combat damage occurs, the system is either disabled or not based on the kill probability. If you add more branches to the tree, they are examined in case the system is not disabled.
- *Child Node Probabilities Disjoint* refers to the branches under the node. If only one of the branches can occur at any point in time, then select Yes. If any combination of the branches can occur, select No.

When you first add a combat damage definition, only the top branch is shown. You add more branches by clicking the Add button on the right. This brings up the form shown in Figure B.45.

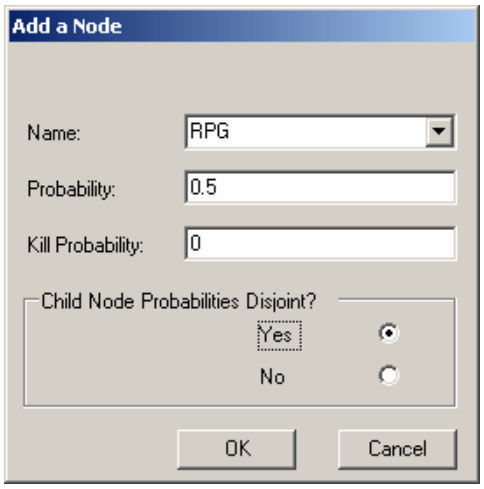

Figure B.45 Form to Add a Node to the Combat Damage Tree

- *Name*. If this is an intermediate node, enter the name of your choosing, such as RPG. If this is a leaf branch, select from the drop-down list of elements that are relevant for the selected system type.
- *Probability*. This is the probability of the branch. For the RPG branch it is the probability that the system gets hit by a RPG given that combat damage occurs. So, probabilities below the top branch are conditional.
- *Kill Probability*. This is the probability that if the branch is true, the system is disabled. In this case we do not want to include kill probabilities until the tree is developed further to include the direction from which the hit occurred.
- *Child Node Probabilities Disjoint* refers to the branches under the node. If only one of the branches can occur at any point in time, then select Yes. If any combination of the branches can occur, select No.

In Figure B.46 we have further developed the combat damage definition tree. In the example of Figure B.46, there are two possible munitions considered: RPG and Mortar. We specified that these are disjoint, that is, it is highly unlikely that more than one of these munitions would hit the system at the same time. On the other hand if a Mortar hits the vehicle, it will hit the front, left, rear or right side. The RPG is represented similarly. Clicking on any node in the tree displays properties for that node.

Figure B.47 shows the combat damage definition tree with all nodes expanded. We have shown only the wheels to keep the example simple. Note that if an RPG hits the vehicle from the right side, any or all wheels on that side may be damaged (the child nodes are not disjoint). Also, notice that a kill probability has been specified at the level that we know what type of munition hit the NLOS-C and the direction of the hit.

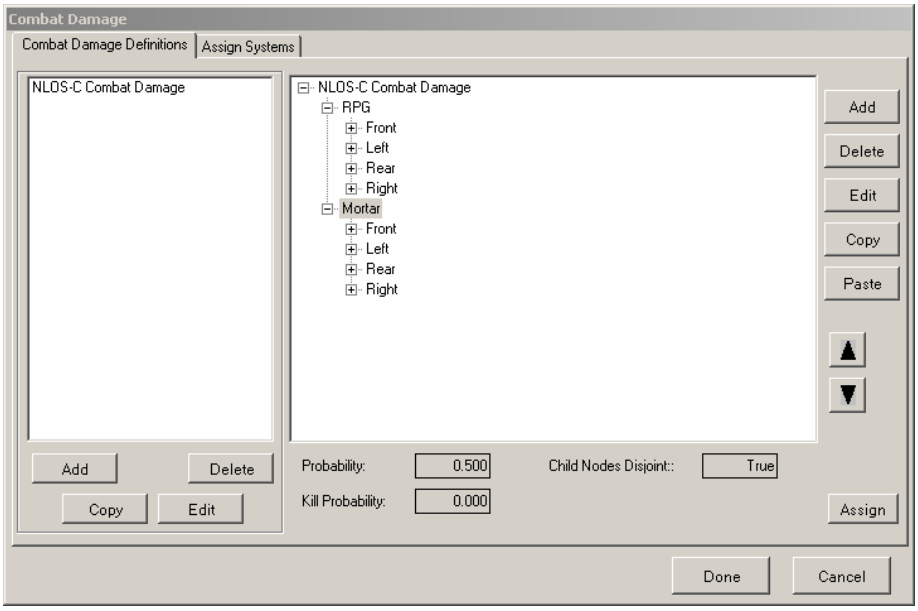

Figure B.46 Combat Damage Definition Including RPG and Mortar

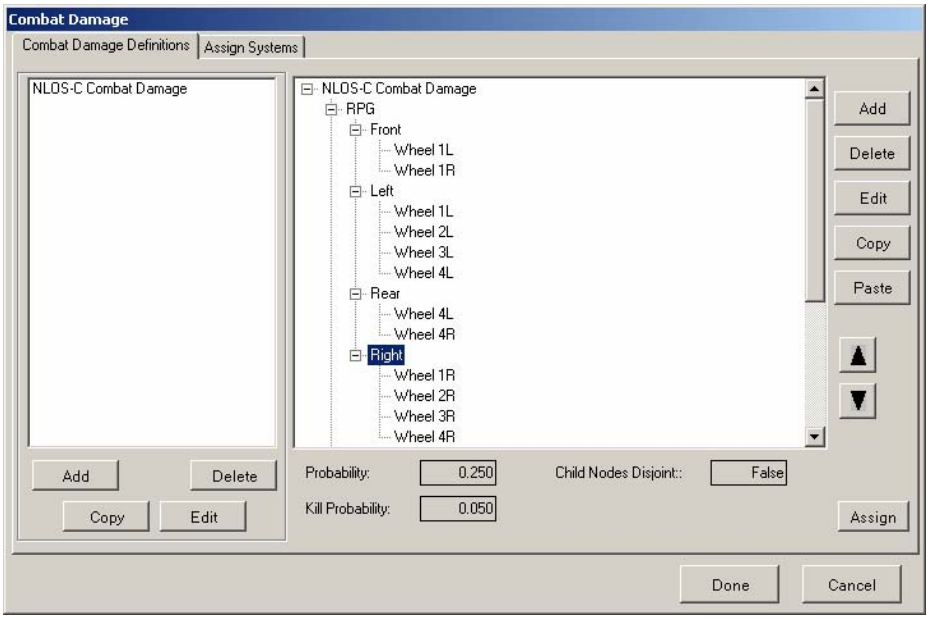

Figure B.47 Combat Damage Tree Expanded

B.9.2. Assign Systems Tab

The Assign Systems tab is shown in Figure B.48. This form is accessed by clicking the **Assign** button on the Combat Damage Definitions tab (Figure B.47). The task here is to assign combat damage models to the systems. This assignment can change during a simulation due to the affects of an external condition that can arise during a segment of the system's assigned scenario. Note that combat damage is optional so not every system need be assigned a combat damage model.

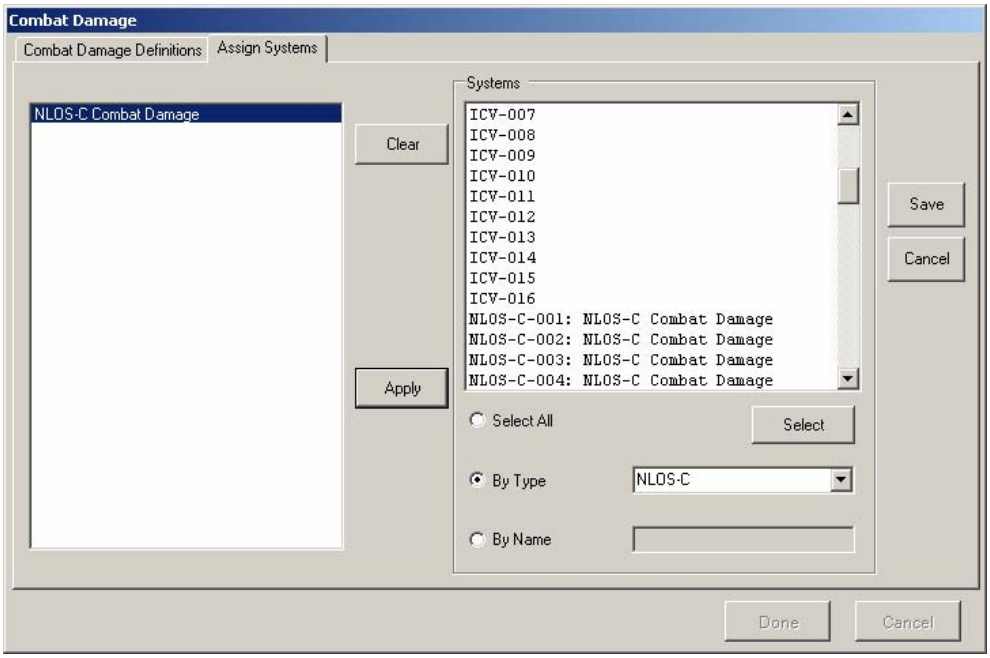

Figure B.48 Assigning Combat Damage Models to Systems

- *Combat Damage Model List* contains the unique names for each combat damage model. You defined these on the Combat Damage Definitions tab and the list is not editable here.
- *Systems List* contains all of the systems (section B.2).
- *Apply* button assigns the highlighted combat damage model to the highlighted system(s). First select the model to be assigned then select the system(s) to which the model belongs. One way to select a system is to scroll the list to the desired system and click on it. Multiple selections can be made by holding down the Ctrl key (or shift key for a block of systems) while clicking on the system IDs.
- *Select* button gives you additional ways to select the systems to be assigned as discussed in section B.5.

Distribution List

Internal:

10	1176	Anderson, Dennis J.., 15242
10	1176	Campbell, James E., 15242
1	1176	Chapman, Leon D., 15242
5	1176	Cranwell, Robert M., 15242
1	1176	Shirah, Donald N., 15242
1	1176	Thompson, Bruce M., 242
1	9018	Central Technical Files, 8945-1
2	0899	Technical Library, 9616

External:

Longsine, Dennis
9111-A Research Blvd.
Austin, Texas 78758
512-425-2022