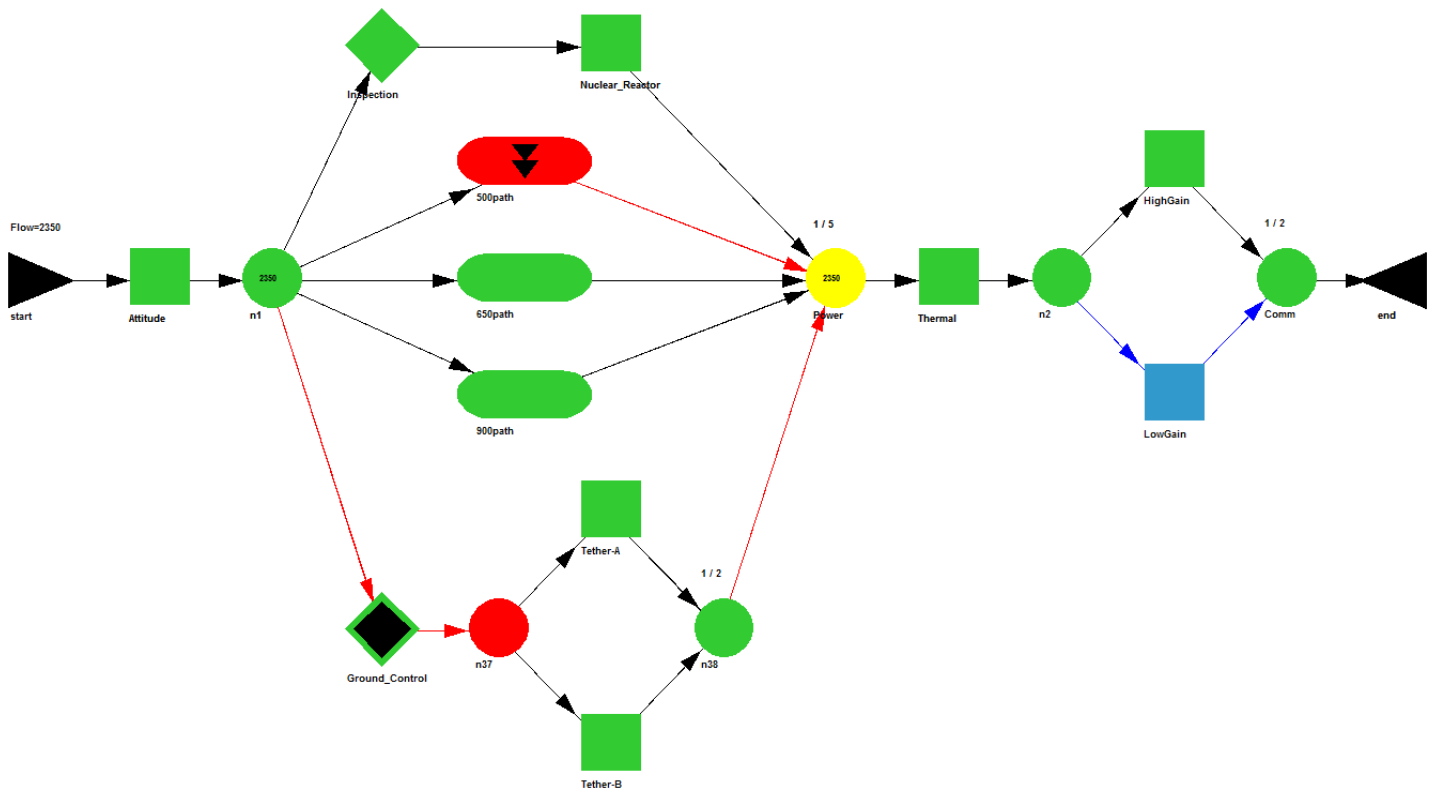


RAPTOR

Version 7



Tutorial Workbook

RAPTOR 7.0 Tutorial Workbook

Fourth Edition

Program Management
Steve Padilla
raptor@bah.com
<http://raptorplus.com>

Program Development
Kenneth E. Murphy
Charles M. Carter
Elizabeth A. Grimes
Anthony W. Malerich

Copyright © 2013 by Booz Allen Hamilton. All rights reserved. Printed in the United States of America. Except as permitted under the United States Copyright Act of 1976, no part of this publication may be reproduced or distributed in any form or by any means, or stored in a data base or retrieval system, without the prior written permission of the publisher.

Table of Contents

Overview	vii
Raptor Workbook Tutorial	
Chapter 1 Elements and Views	
Problems	1-6
Chapter 2 A Basic Reliability Block Diagram	
Adding a Block	2-2
Copying and Pasting	2-3
Adding Nodes and Links	2-5
Simulating	2-7
Simulation Toolbar	2-8
Simulation Results	2-9
Problems	2-11
Chapter 3 Distributions	
Selecting a Failure Distribution	3-3
Distribution Nomenclature	3-5
The Empirical Distribution	3-16
Problems	3-19
Chapter 4 Logistics Constraints	
Custom Sparing	4-4
Spare Pool	4-7
Infinite Sparing	4-7
Creating a Spare Pool	4-8
Creating a Resource Pool	4-11
Sparing Results	4-13
Repair Cycle	4-15
Problems	4-15
Chapter 5 Hands On Example I	
Problems	5-7
Chapter 6 Preventive Maintenance	
Definition	6-2
Setting a Block to Perform PM	6-2
Setting Duration of Preventive Maintenance	6-4
Defining a Trigger	6-7
Tying PM to a Trigger	6-9
Starting a Simulation in Step Mode	6-13
Problems	6-16

Chapter 7	Dependency	
	Definition	7-2
	Dependent Block Diagrams	7-2
	General Block Properties	7-3
	System Dependence	7-6
	Local Dependence	7-7
	Elemental Dependence	7-8
	Generating Detailed Event Log	7-13
	Problems	7-14
Chapter 8	Standby	
	Definition	8-2
	Node k-star Value	8-3
	Node Priority	8-4
	Imperfect Switchover	8-5
	Standby Stress	8-6
	Perfect Switchover	8-9
	Standby and Dependence	8-9
	Problems	8-11
Chapter 9	Transients	
	Definition	9-2
	Delayed Statistics Gathering	9-2
	Displaying Ao Plots	9-3
	Graph Window.....	9-4
	Setting a Graph's Y-Axis	9-6
	Components With Exhausted Life	9-17
	Problems	9-18
Chapter 10	Failure Effects View	
	Definition	10-2
	Initiating Failure Effects View	10-2
	Failing a Component	10-2
	System Status Summary	10-3
	Induced Dependency	10-4
	Refreshing the RBD	10-4
	Create Single Point Failure File	10-5
	Using the Interactive Log	10-5
	Problems	10-6
Chapter 11	Weak Link Analysis	
	Conducting a Weak Link Analysis	11-3
	Weak Link Analysis Type	11-4
	Defining Thresholds	11-4
	Domain Diagram	11-5
	Node Analysis Results	11-5
	Block Analysis Results	11-6
	Availability & Dependability Equations	11-7

Reliability Equations	11-16
Weak Link Analysis View	11-16
Problems	11-17
Chapter 12 Cost Analysis	
Block Cost Fields	12-3
Spare Pool Cost Fields	12-6
Resource Cost Field	12-6
System Cost Fields	12-7
Block Costs	12-8
Example Cost Report	12-11
Interim Cost Report	12-14
Problems	12-15
Chapter 13 Hands On Example II	
Problems	13-9
Chapter 14 Capacity Analysis	
Flow & Capacity Equations	14-2
Flow Generation & Cost of Loss	14-3
Nominal & Maximum Flows	14-5
Capacity Summary Results	14-9
Node Capacity Results	14-10
Usage & Capability Equations	14-10
Capacity Plot	14-12
Problems	14-18
Chapter 15 Events	
Adding an Event	15-3
System Status Indicators	15-6
Complex RBDs	15-7
Problems	15-7
Chapter 16 Hierarchy	
Collapsing Elements into a Hierarchy	16-4
Home and Up Navigation Buttons	16-5
Adding a Hierarchy	16-6
Dragging Elements into a Hierarchy	16-7
Hierarchical Model	16-8
Importing an RBD as a Hierarchy	16-9
Simulating a Hierarchical RBD	16-11
Displaying the Hierarchical Tree	16-13
Goto Hierarchy Button	16-15
Descendant Status Indicator	16-16
Problems	16-19
Chapter 17 Phasing	
Definition	17-2

Setting a Block to be Phased	17-3
Setting an Event to be Phased	17-4
Setting a Node to be Phased	17-5
Setting a Hierarchy to be Phased	17-6
Defining Phases	17-7
Assign Block Phases	17-9
Component Functioning State Colors	17-11
Component Link and Cut State Colors	17-12
Assign Node Phases	17-13
Assign Event Phases	17-14
Assign Hierarchy Phases	17-15
Phased DSI Colors	17-16
Phased Elements in the Workspace View	17-16
System Availability and Dependability Equations	17-17
Phased Elements in the Simulation View	17-17
Cycle Terminated Simulation	17-21
Problems	17-22
Chapter 18 Hands On Example III	
Problems	18-3
Chapter 19 Simulation Mechanics	
System Random Numbers	19-2
Example I Time Line	19-3
Example II Time Line	19-5
Problems	19-6
Chapter 20 Miscellaneous Topics	
Preferences Dialog Box	20-2
Failure Rate Mode	20-3
System Settings Dialog Box	20-4
Simulation Settings	20-5
Setting System Streams	20-5
Mass Edit Dialog Box	20-7
Import & Export Libraries	20-10
Print Tables	20-11
Print Window	20-12
Find Item Dialog Box	20-12
Linear Growth or Decay	20-14
Geometric Growth or Decay	20-14
Asymptotic Growth or Decay	20-15
Random Numbers	20-16
Block Defaults	20-17
Advanced Simulation Options	20-19
Problems	20-19

Appendix A: Solutions to Problems

Chapter One Solutions	A-1
Chapter Two Solutions	A-1
Chapter Three Solutions	A-2
Chapter Four Solutions	A-4
Chapter Five Solutions	A-5
Chapter Six Solutions	A-7
Chapter Seven Solutions	A-8
Chapter Eight Solutions	A-9
Chapter Nine Solutions	A-11
Chapter Ten Solutions	A-12
Chapter Eleven Solutions	A-13
Chapter Twelve Solutions	A-13
Chapter Thirteen Solutions	A-14
Chapter Fourteen Solutions	A-14
Chapter Fifteen Solutions	A-17
Chapter Sixteen Solutions	A-19
Chapter Seventeen Solutions	A-20
Chapter Nineteen Solutions	A-23
Chapter Twenty Solutions	A-24

Appendix B: Menus and Button Bars

Workspace View Menu Bar	B-1
Simulation View Menu Bar	B-10
Weak Link View Menu Bar	B-14
Failure Effects View Menu Bar	B-16
Ao Graph Menu Bar	B-21
Workspace View Toolbar	B-23
Simulation View Toolbar	B-25
Weak Link View Toolbar	B-27
Failure Effects View Toolbar	B-28

Appendix C: File and Reports

All system failure times file	C-1
All system down times file	C-1
Key parameters file	C-2
Ending sim times file	C-3
Ao plot data	C-3
Capacity plot data	C-4
Final Results report	C-5
End of run report	C-10
Detailed event log	C-12
Interim cost log	C-13

Appendix D: Raptor Color and Statuses

Block Colors	D-1
--------------------	-----

Node Colors	D-5
Event Colors	D-6
Hierarchy Colors	D-9
Hierarchy DSI Colors	D-13
Marker Colors	D-14
Link Colors	D-15
System Colors	D-15
System Status Indicators	D-16
Block Status Diagram	D-18
Node Status Diagram	D-18
Event Status Diagram	D-19
Hierarchy Status Diagram	D-19
Link Status Diagram	D-20
System Status Diagram	D-20

Appendix E: Raptor Rules

Elemental	E-1
Distributional	E-1
Logistics	E-2
Cost	E-2
Phasing	E-3
Growth and Decay	E-3
Standby	E-4
Capacity	E-4
Random Start-ups	E-4
Preventive Maintenance	E-5

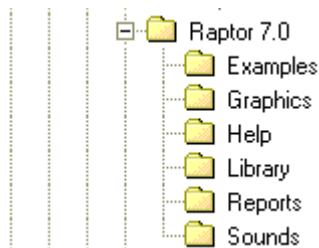
Appendix F: Raptor Definitions

System-level Parameters	F-1
Weak Link Analysis Parameters	F-3
Growth and Decay	F-6
Capacity	F-7

Overview

This workbook concentrates on the use of the Raptor software and can be subdivided into three subsections, namely: novice, journeymen, and expert subsections. Chapters 1 through 6 constitute the novice subsection. After completing these lessons, a user should be able to model most systems. Chapters 7 through 13 comprise the journeymen subsection and enhance a Raptor user's ability to conduct sophisticated analysis of most Raptor simulations. Finally, the expert subsection encompasses Chapters 14 through 20. These lessons provide a Raptor user with all the knowledge needed to conduct rigorous simulations and analysis of any system.

When Raptor is installed, it will automatically create the following folders under the file directory that was chosen for installation (default is C:\Program Files\Raptor 7.0, Program Files (x86) for 64 bit operating systems). The top-level Raptor 7.0 directory



contains the program executable files. The Examples directory contains .rbd files created by Raptor for teaching purposes and is the default directory for .rbd files created by a user. The Graphics directory includes the objects that are displayed within the different Raptor Views and the Help directory contains the help files. The Library directory is the default location for library files created by a user and the Reports directory is the default location for text files

generated by a user. Finally, the Sounds directory includes the .wav files associated with the Raptor tool. If a user re-installs Raptor, only files or directories that were created by the original installation will be over-written. Therefore, user .rbd files, libraries, configuration files and text files will not be over-written unless they have the same filename as Raptor installation files.

Several types of notation are incorporated within this workbook to enhance its usability. **Bold text** is used to indicate a file that needs to be opened or created. *Italic text* refers to either menu options that the user is instructed to follow or phrases that need emphasis. The context surrounding the italic text should easily allow the reader to discern between the different uses.

If assistance is needed in solving the problems within this workbook or clarification is needed regarding the solutions, contact the Raptor Team at raptor@bah.com. Finally, you can always obtain the latest Raptor information concerning new releases, training and consulting via the Raptor website at www.raptorplus.com.

Chapter 1



Objectives

1. Understand the role Reliability Block Diagrams (RBD) play within Raptor.
2. Understand the roles that blocks, events and hierarchies perform in an RBD.
3. Understand the roles that nodes and links perform in an RBD.
4. Be able to distinguish between the Workspace, Simulation, Weak Link Analysis and Failure Effects Views.

Recommended Problems

1, 2 and 4

Recommended Reading

Classic Bamboozles of Reliability (Murphy)

Elements and Views

A reliability block diagram is a graphical representation of a system's components arranged in a manner to convey system-level failures. Included within this representation are the logical relationships between the components and the system itself. Reliability block diagrams are denoted as "RBDs" and usually contain components arranged in a combination of series (Figure 1.1) and parallel (Figure 1.2) sub-structures.

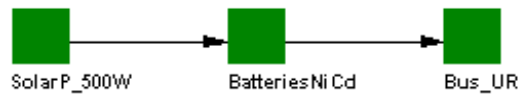


Figure 1.1 Series Sub-structure

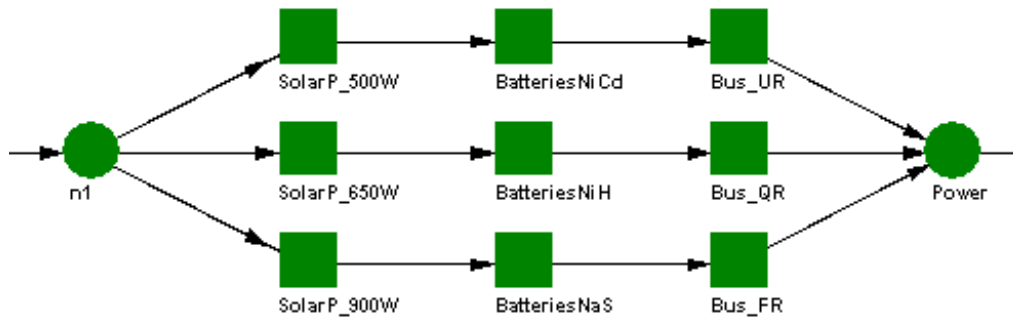


Figure 1.2 Parallel Sub-structure

All elements of a series sub-structure must function in order for the system containing these elements to work properly. A parallel sub-structure, however, is one in which a pre-specified number of elements must function properly or else the system will not work. These parallel sub-structures are usually denoted using a "*k-out-of-n*" nomenclature, where *k* is the number of parallel elements that must function properly for the system to work, and *n* is the total number of these elements within the parallel sub-structure. Raptor operates based on an RBD that represents a system for which Reliability, Availability or Maintainability (RAM) characteristics are desired. RBDs can be comprised of six different types of elements: blocks, events, nodes, links, hierarchies and markers. A block usually represents a physical piece of equipment, but can also represent a virtual concept. For example, a block could represent either software contained within a system or the failures induced by other entities (e.g., humans, commercial power, etc.). All blocks fail and repair based on probabilistic distributions as time passes through a simulation. When a block initiates its repair cycle as a result of a failure, it consumes resources, spares and time. Events represent actions that either do or do not occur. As a default, events resolve their status, or "fire", at the start of a simulation, and their impacts are assessed without the passage of simulation time. Events are based on draws from the Bernoulli distribution, which is a



Block Symbol



Event Symbol

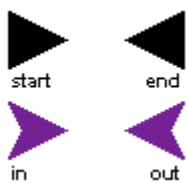


Node Symbol



Hierarchy

discrete function with only two possible outcomes. Success occurs with a probability of p while failure occurs with a complementary probability of $1-p$. Although events can either fire good or fire bad, they never initiate a repair cycle. It is, however, possible to re-fire events or change the probability of firing as time passes. Examples of events include inspections, switches, landing gear, etc. Nodes and links tie the blocks and events together such that the failure logic of a system is defined. That is, nodes and links define the series-parallel sub-structures within a system-level RBD. If a node only has one link coming into it, then it is considered a pass through or “passive” node since it does not contain any redundancy information. If a node has more than one link coming into it, then the node is known as a k -out-of- n node. These nodes house the redundancy information that defines the parallel sub-structure of a system. Hierarchical elements contain a collection of elements, including additional hierarchies themselves and often



represent major subsystems of the system being modeled. Hierarchical elements essentially represent a window into a lower level indenturing of a component. For example, a hierarchy may represent an aircraft engine that itself contains hundreds of sub-components including additional hierarchies of even lower indentured sub-systems. Start and end markers are in every RBD and if hierarchies are present, in and out markers are also included. These markers are not considered active elements of an RBD since they merely define the beginning and end of an RBD structure and its substructures.

The Raptor tool has four modes in which the user may interact with the software. The first mode is called the “Workspace View” (Figure 1.3). In this mode, the user creates, edits and assembles a system-level RBD. The Workspace view contains a toolbar that facilitates building RBDs, and a status bar that indicates attributes of an element. The second mode is called the “Simulation View” (Figure 1.4). This mode contains a simulation menu and toolbar to control the aspects of the ongoing simulation. Additionally, this mode contains: the status of the individual elements, system status, and a status bar that indicates cumulative results. The third mode, which can be accessed after a weak link analysis has been performed, is called the “Weak Link Analysis View” (Figure 1.5). This view helps users to determine the weakest components of a system. Lastly, the user can utilize the “Failure Effects View” (Figure 1.6) to determine the impact on a system due to single or multiple component failures. Examples of all four views are shown respectively in the next four figures.

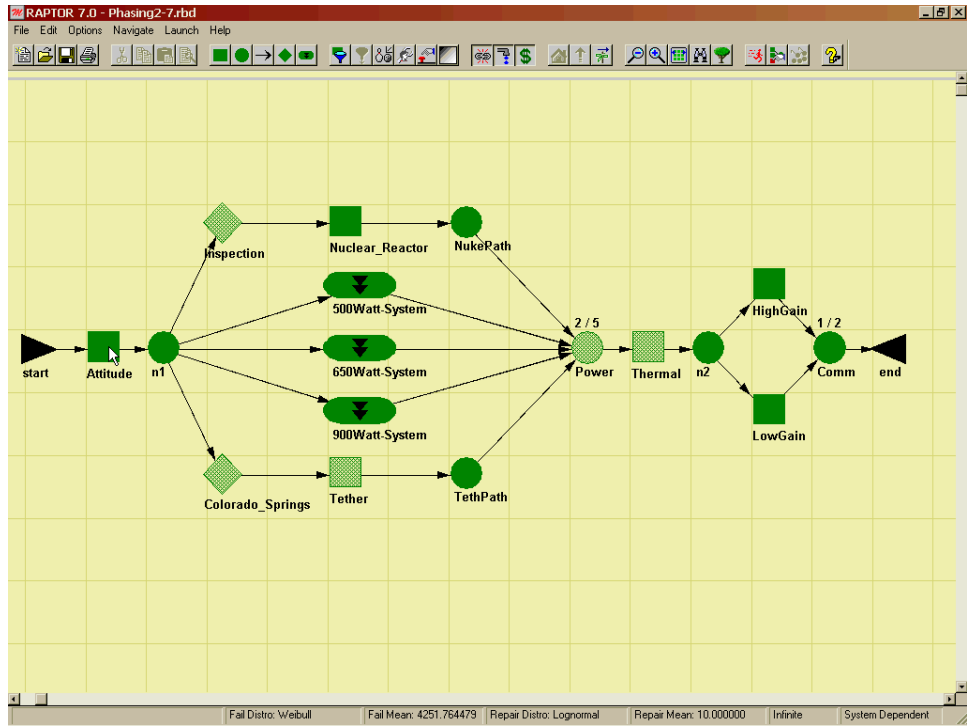


Figure 1.3 Workspace View

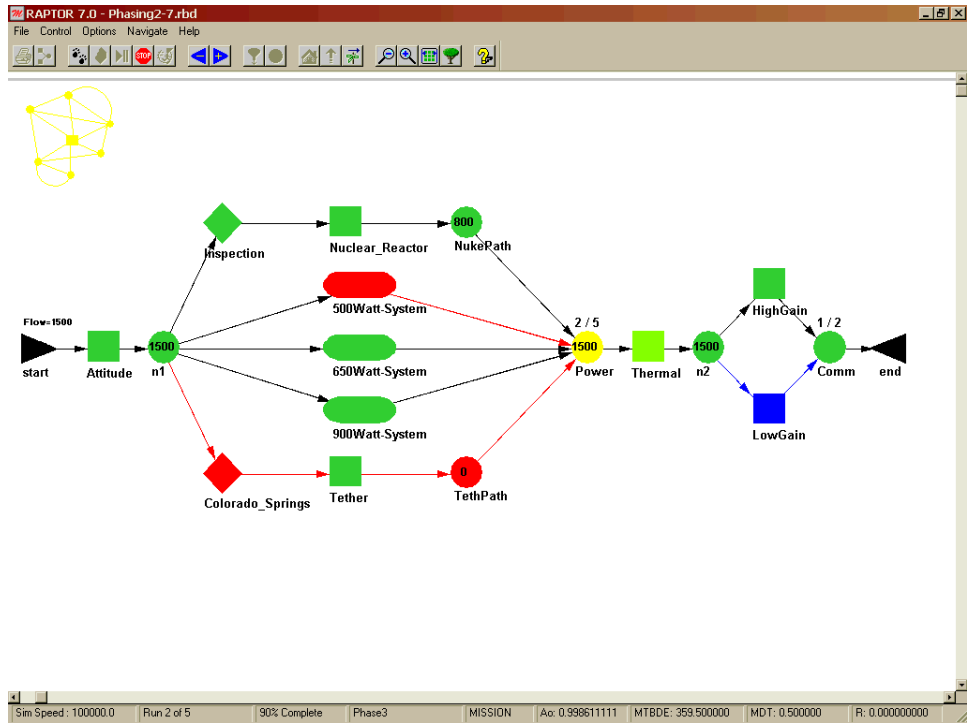


Figure 1.4 Simulation View

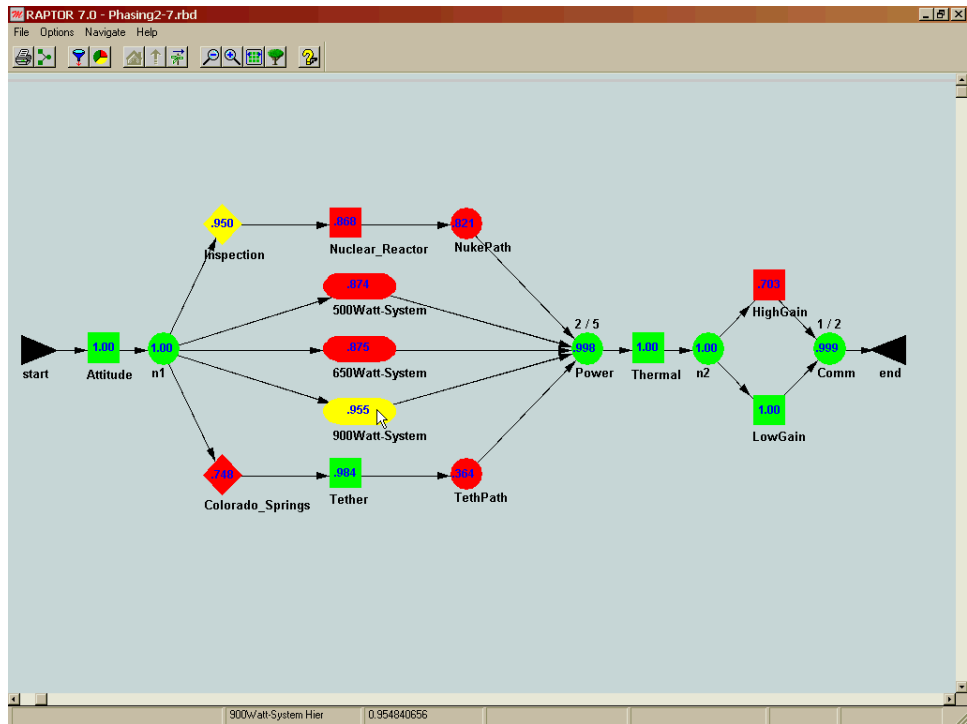


Figure 1.5 Weak Link Analysis View

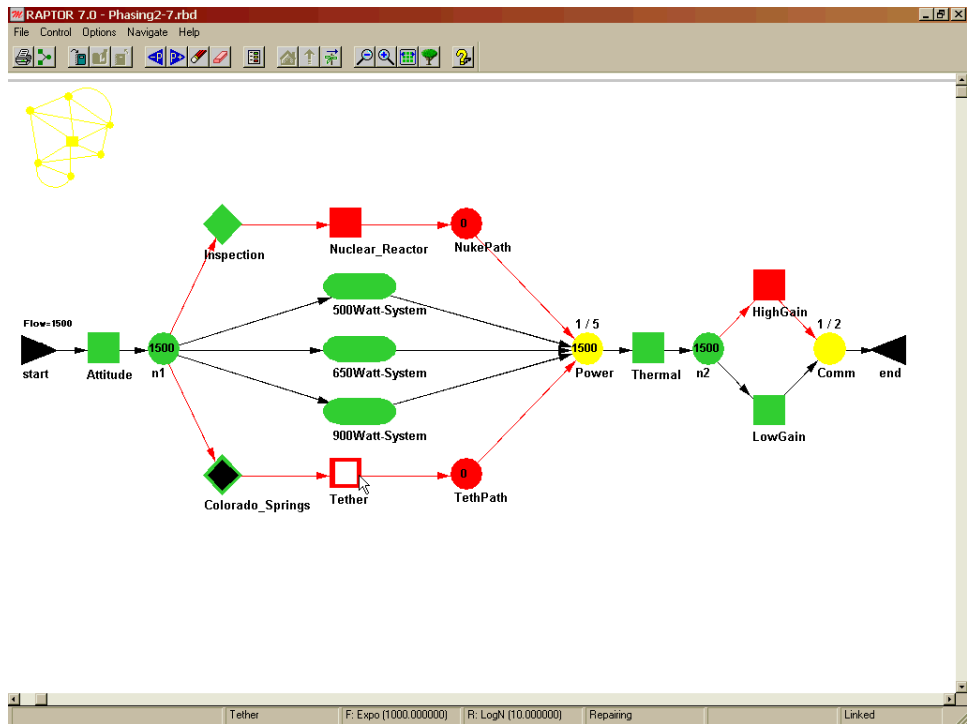


Figure 1.6 Failure Effects View

Problems

1. Place elements into the Workspace View and determine the rules for connecting links to nodes, blocks, events and hierarchies.
2. State the two fundamental differences between events and blocks.
3. What kind of information is displayed in the status bar while in the Workspace View versus the Simulation View?
4. Which view allows the user to modify an RBD and which views allow the viewing of results?

Chapter 2



Objectives

1. Build and simulate an RBD.
2. Understand the difference between time-truncated and failure-truncated simulations.
3. Understand the values displayed in the Output Tables dialog box.
4. Understand the toolbars of the Workspace and Simulation Views.

Recommended Problems

1, 2, 7, 9, 10, 12 and 15

Recommended Readings

How Long Should I Simulate, and How Many trials? A Practical Guide to Reliability Simulations (Murphy, Carter & Wolfe)

What is Truth? A Practical Guide to Comparing Reliability Equation Answers to Simulation Results (Murphy, Malerich & Carter)

A Basic Reliability Block Diagram

We want to complete the building of the RBD located within the file labeled **Basic1.rbd** in such a way that when we have completed this chapter it looks like the RBD in Figure 2.1.

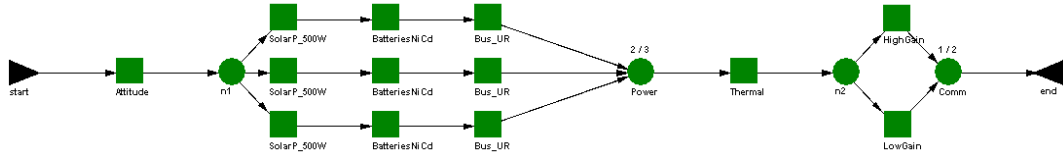


Figure 2.1 Completed RBD for Chapter 2

Open **Basic1.rbd** from the *File* menu most recently used list and notice that the structure shown in Figure 2.2 is partially built for you.

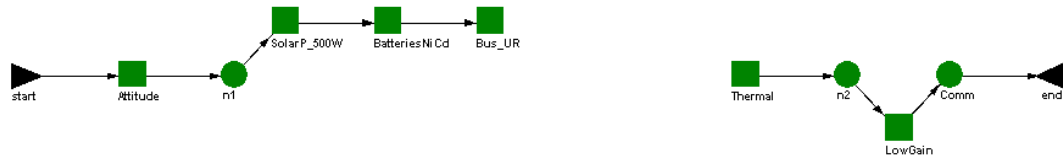


Figure 2.2 Initial RBD for Chapter 2

Add a block or component called "HighGain" (between the n2 and Comm nodes) with an exponential mean life of 350 hours, a lognormal repair mean of 72 hours and a standard deviation of 12 hours. Figure 2.3 illustrates how a component can be added via the *Edit – Add – Block* menu item.

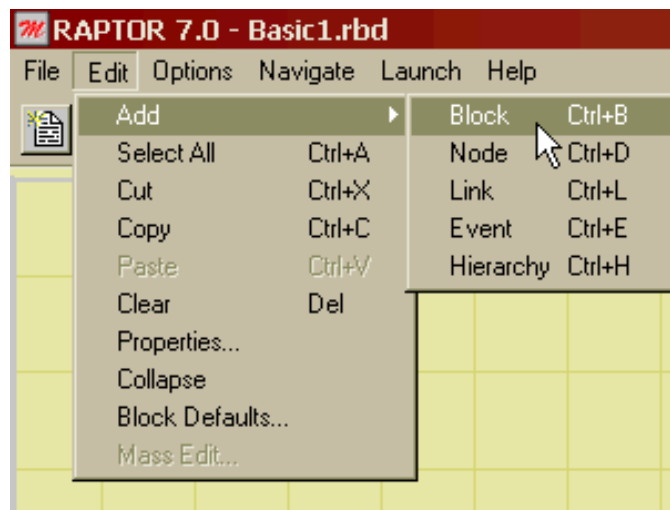


Figure 2.3 Add Block from Edit Menu Option

Figure 2.4 displays the Block Properties dialog box that appears after a component is placed within the Workspace View. The title of the dialog box changes from “unnamed” to the text specified in the “Block Name” value box when the “Update Statistics” button is clicked. In addition, the mean and standard deviation of the failure and repair distributions are determined by Raptor and displayed near the bottom of the dialog box.

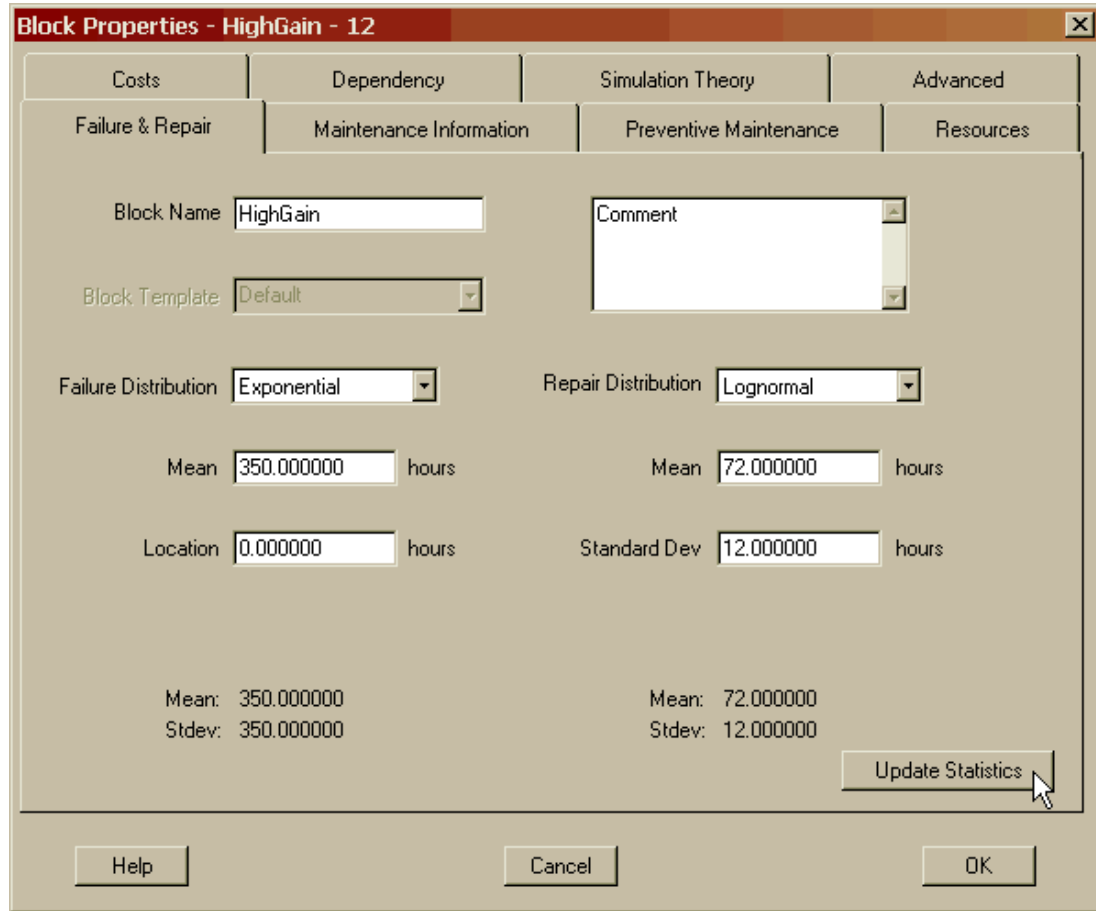


Figure 2.4 HighGain Failure and Repair Distributions

Use the right mouse click to exit from add block cursor mode. If subsystems are identical, you can use any one of four methods to copy and paste items onto the screen. In all four cases, you must first select an item by single clicking on it and thus changing its color to Raptor’s highlight color of blue. The traditional methods are to use the copy and paste options under the *Edit* menu or the corresponding buttons from the toolbar. An easier way is to use the right mouse click while the cursor is over the object, and then select the copy and paste options from the right click menu. Finally, you can use the Windows® standard for copying and pasting (i.e., Ctrl C and Ctrl V, respectively). Copy the 500-Watt solar panel (SolarP_500W) and add two more solar panels to the screen using any method you prefer. Figure 2.5 shows two of the four methods that can be used to copy or paste elements into the Workspace View.

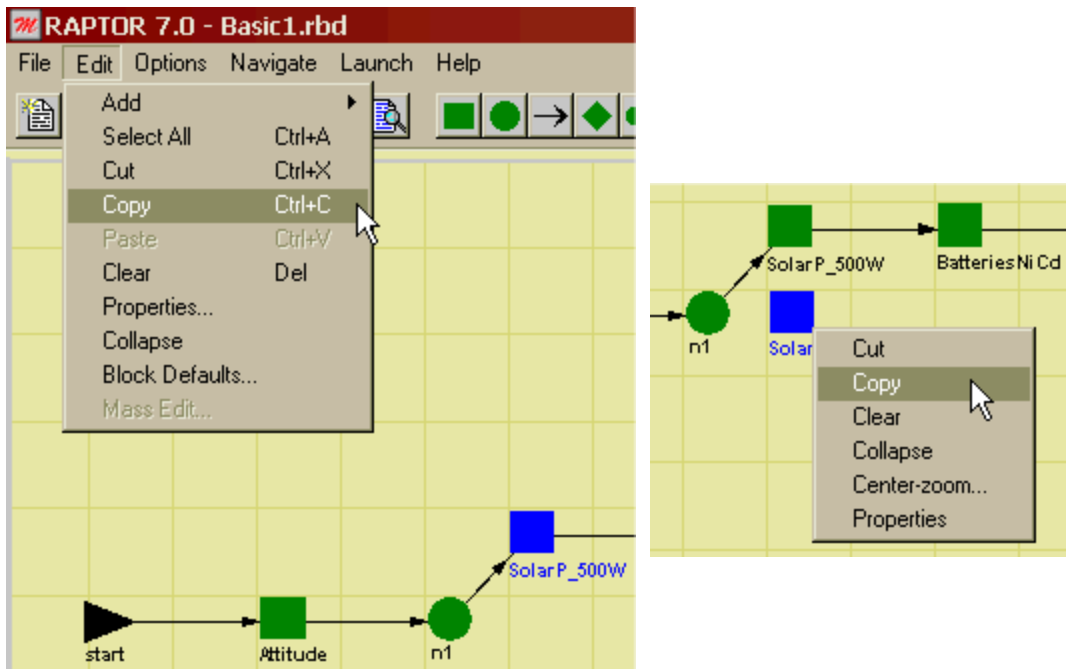


Figure 2.5 Copying Elements

Raptor is not limited to single element copying and pasting. Hold the left mouse button down and drag the mouse to activate a rubber-band box that allows you to highlight the nickel-cadmium batteries (BatteriesNiCd) and the unregulated bus (Bus_UR). While the components are highlighted, perform a right mouse click over one of the components and then select the copy option from the menu that appears. Paste the components twice such that your RBD mimics the one shown in Figure 2.6. Notice that all links not completely enclosed by the rubber-band box are not copied.

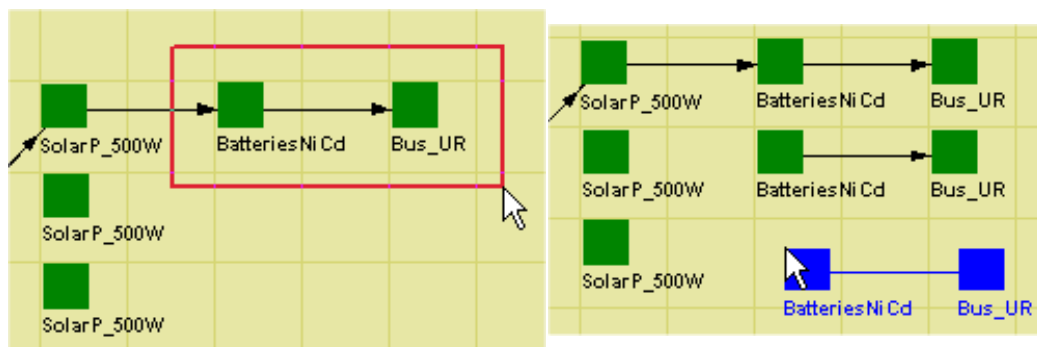


Figure 2.6 Multiple Copying and Pasting

Now that all the physical pieces of the system are placed into the Workspace, let us add the nodes and links needed to define the series-parallel logic of the system. Figure 2.7 displays a node being added to the Workspace View via the *Edit* menu option. Add a node after the bus components and before the thermal component as shown in Figure 2.8. Use the right mouse click to exit from add node cursor mode.

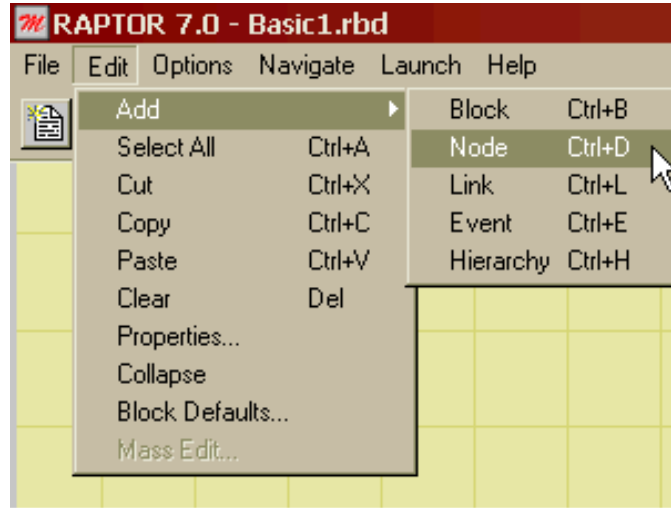


Figure 2.7 Add Node from Edit Menu Option

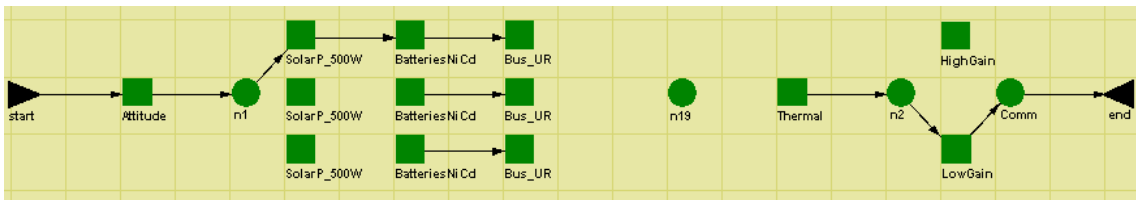


Figure 2.8 Node Added to RBD

Add links via the right mouse click menu option and connect the RBD so that it matches the RBD shown in Figure 2.9. Right click anywhere in the Workspace to halt the action of adding links to the model.

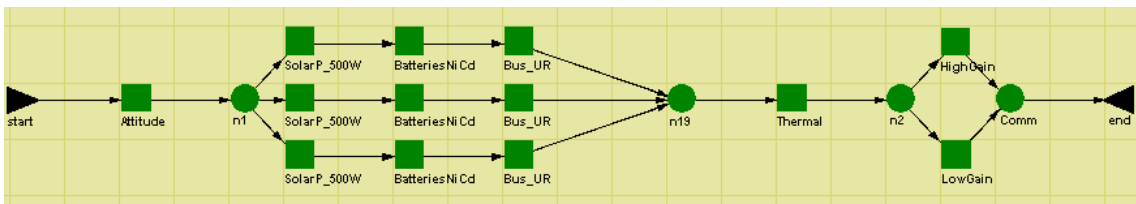


Figure 2.9 Complete RBD

Double-click the node labeled “n19” and change its name to “Power”. Set the number of good paths to be two (i.e., $k = 2$). This is the number of parallel strings (or paths) that must operate for this parallel substructure to function. The modifications to node n19 are shown in Figure 2.10.

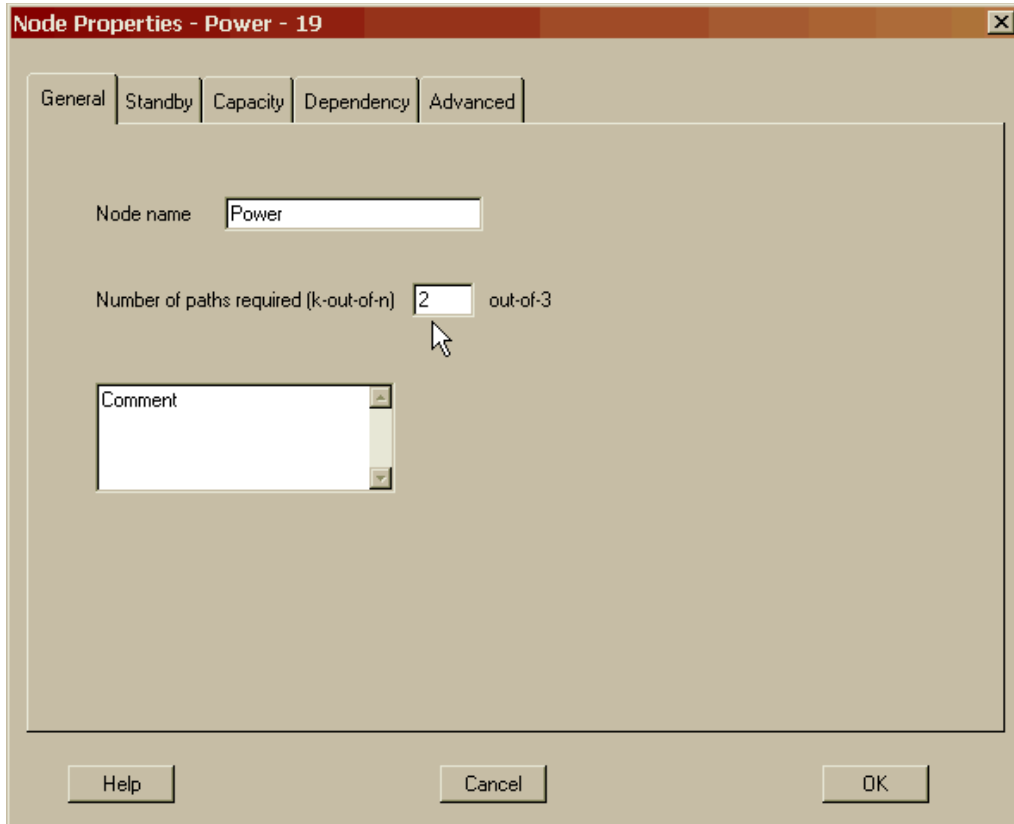


Figure 2.10 Node Properties Dialog Box

Immediately after you leave the Node Properties dialog box, the k -out-of- n information is displayed above the node, and the name is displayed below the node as shown in Figure 2.11. To finish construction of this RBD, double-click on the Comm node and ensure its k -out-of- n is set to 1-out-of-2.

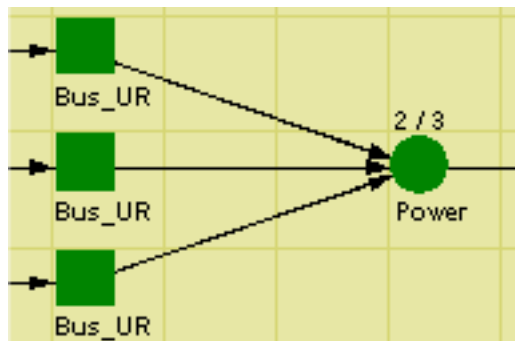


Figure 2.11 A Node’s k -out-of- n and Name

Now that we have added all the required components, nodes, and links, we have a complete RBD that is ready to be simulated. Choose the *Simulation* option from the *Launch* menu or from the right click menu as shown in Figure 2.12.

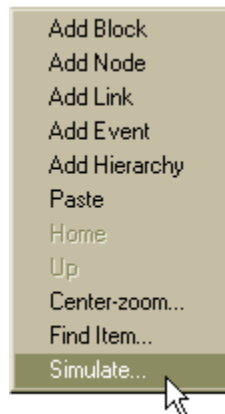


Figure 2.12 Simulation from the Right Click Menu

This menu item will bring up the Simulation Options dialog box shown in Figure 2.13. Let us conduct a time-truncated simulation for 730.5 hours. We will instruct Raptor to repeat the simulation ten times, that is, ten trials of 730.5 hours in length.

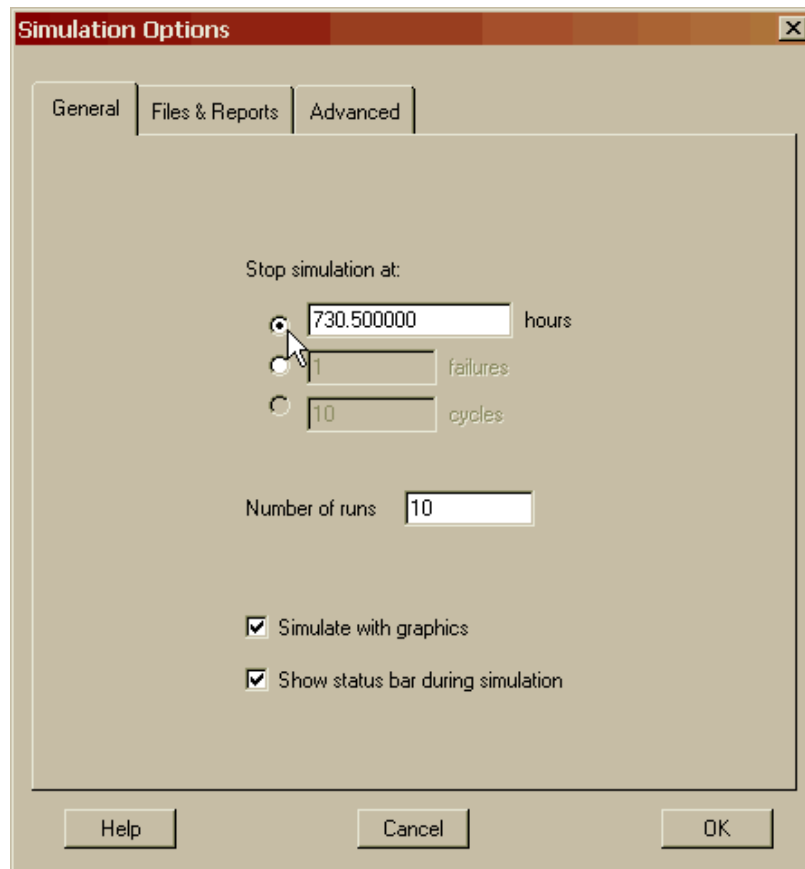


Figure 2.13 Simulation Options Dialog Box

Start the simulation by clicking the OK button of the Simulation Options dialog box and then note the changing system status indicator in the upper left-hand corner of the Simulation View. The images shown in Figure 2.14 represent the system's status.

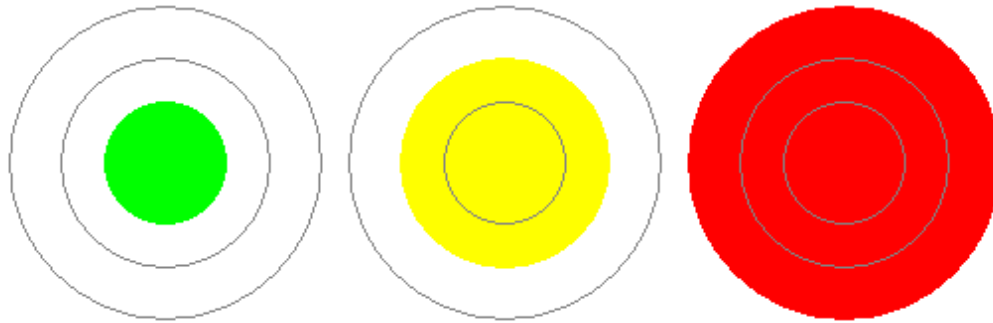


Figure 2.14 Bull's Eye Status Symbols

The green symbol indicates that all elements of the system are functioning properly. The yellow symbol notifies the user that some element or combination of elements have either permanently failed or are being repaired. The red symbol indicates that some combination of elements or possibly a single series element has failed and thus caused the entire system to fail.

The Simulation View toolbar contains a collection of buttons that control the simulation. These buttons are labeled in Figure 2.15 and their functions are explained in Appendix B.

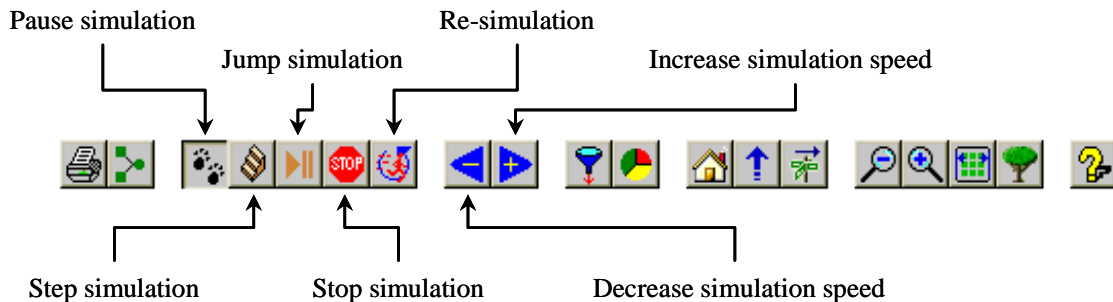


Figure 2.15 Simulation Buttons

The results of the simulation are shown in Figure 2.16 and are obtained by clicking the View Outputs button from the Simulation toolbar or directly from the *Options* menu. The reader should note the mean availability and average Mean Down Time (MDT) for this particular simulation, since we will refer to these values in future lessons. You have now completed your first construction and simulation of an RBD. Future lessons will increase the fidelity of the simulations and demonstrate analysis techniques that help provide scientific rigor to RAM decisions.

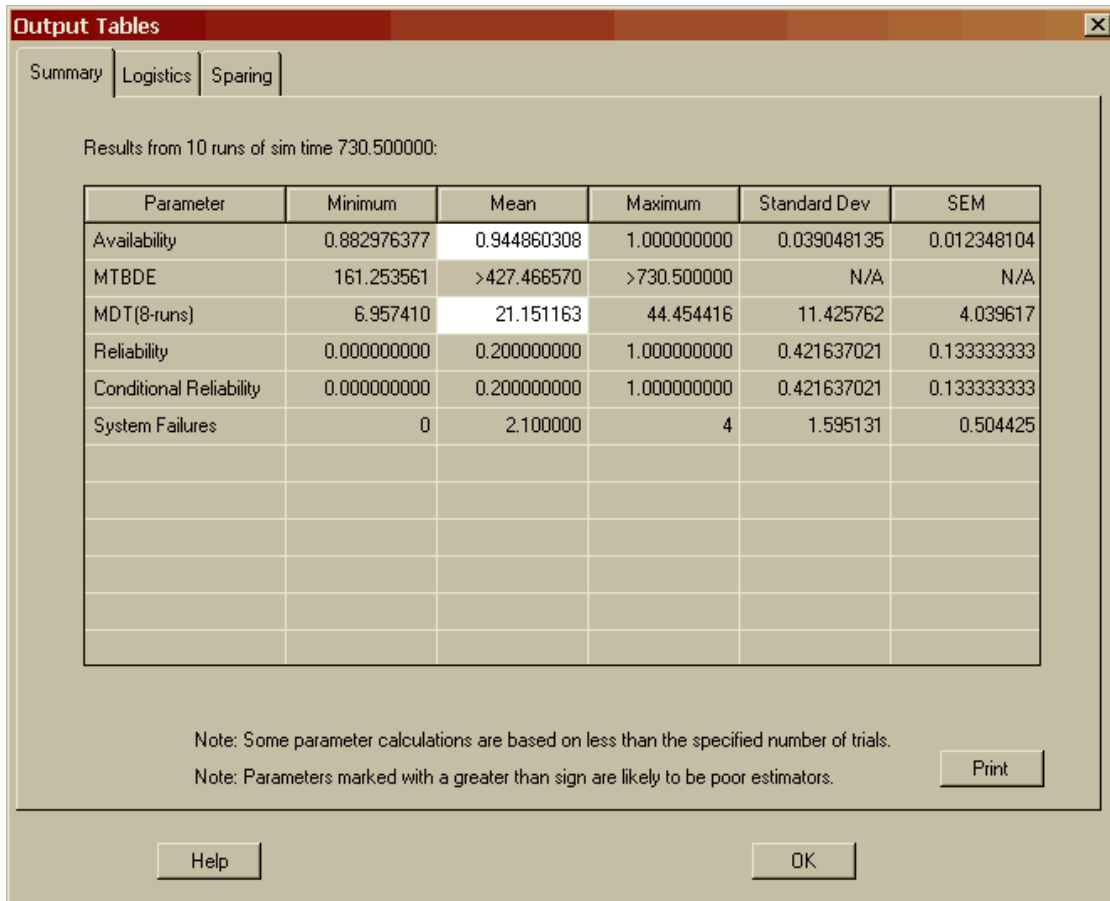


Figure 2.16 Results for Chapter 2

• • •

Click on the Return to Workspace button from the Simulation toolbar to return to the Workspace View. Now let us look at the failure and repair distributions that have been entered for this system by selecting the *Options – Tables – View Block Inputs* as shown in Figure 2.17.

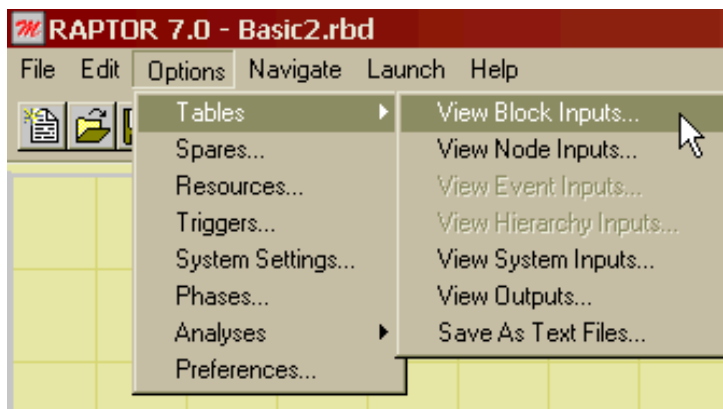


Figure 2.17 Block Inputs from Menu Options

The first tab of the Block Input Tables dialog box lists all the blocks and their associated failure and repair distributions. As shown in Figure 2.18, this RBD is rather simplistic since the failure and repair distributions are identical and somewhat easy to manipulate mathematically.

The screenshot shows a dialog box titled "Block Input Tables" with several tabs: "Failure & Repair Distributions", "Sparing", "Preventive Maintenance", "Maintenance Delays", "Costs", "Dependency", and "Advanced". The "Failure & Repair Distributions" tab is active, displaying a table with the following data:

Block Name	Failure Distro	Param1	Param2	Param3	Repair Distro	Param1	Param2	Param3
Attitude	Exponential	4520.0	0.0		Lognormal	10.0	2.0	
BatteriesNiCd	Exponential	670.0	0.0		Lognormal	50.0	10.0	
BatteriesNiCd	Exponential	670.0	0.0		Lognormal	50.0	10.0	
BatteriesNiCd	Exponential	670.0	0.0		Lognormal	50.0	10.0	
Bus_UR	Exponential	850.0	0.0		Lognormal	42.0	6.50	
Bus_UR	Exponential	850.0	0.0		Lognormal	42.0	6.50	
Bus_UR	Exponential	850.0	0.0		Lognormal	42.0	6.50	
HighGain	Exponential	350.0	0.0		Lognormal	72.0	12.0	
LowGain	Exponential	12350.0	0.0		Lognormal	72.0	12.0	
SolarP_500W	Exponential	820.0	0.0		Lognormal	23.0	1.0	
SolarP_500W	Exponential	820.0	0.0		Lognormal	23.0	1.0	

Below the table, there is a "Hierarchy Filter" dropdown menu set to "All", and buttons for "Apply", "Print", "Help", "Cancel", and "OK".

Figure 2.18 Failure and Repair Distributions for Chapter Two

This simple RBD could be solved using advanced mathematical (i.e., non-simulation) techniques, but as we shall soon see, it takes only a small change to a component's failure distribution, repair distribution, or repair cycle to cause this RBD to be mathematically intractable to all techniques except simulation.

• • •

Finally, Raptor is a unit-less tool, enabling the user to define the simulation unit. As far as a Raptor simulation is concerned, 350 hours is equivalent to 350 kilometers or 350 cycles. Raptor allows the user to define what 350 means by providing a list of standard units in the *Options – System Settings* menu. The System Settings dialog box General tab allows users to choose a unit of simulation as shown in Figure 2.19. The chosen unit will be displayed on any relevant dialog boxes (e.g., see Figures 2.4 and 2.13) so that the user is reminded of the units being used within a particular RBD. Changing units in Raptor, however, does not modify any entered values (e.g., 350 hours does not become 21,000 minutes). System settings changes are associated with a particular RBD; global changes can be made in the Preferences dialog box.

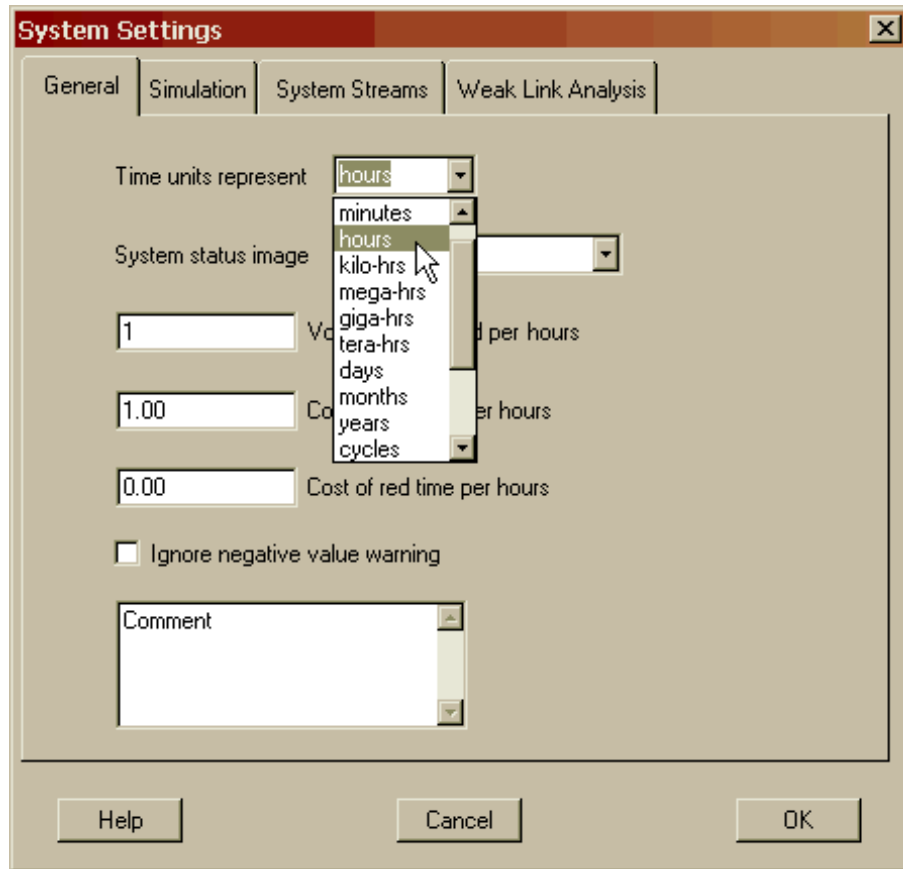
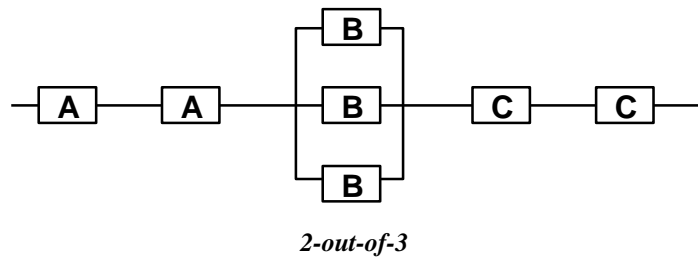


Figure 2.19 Setting Units on the System Settings Dialog Box

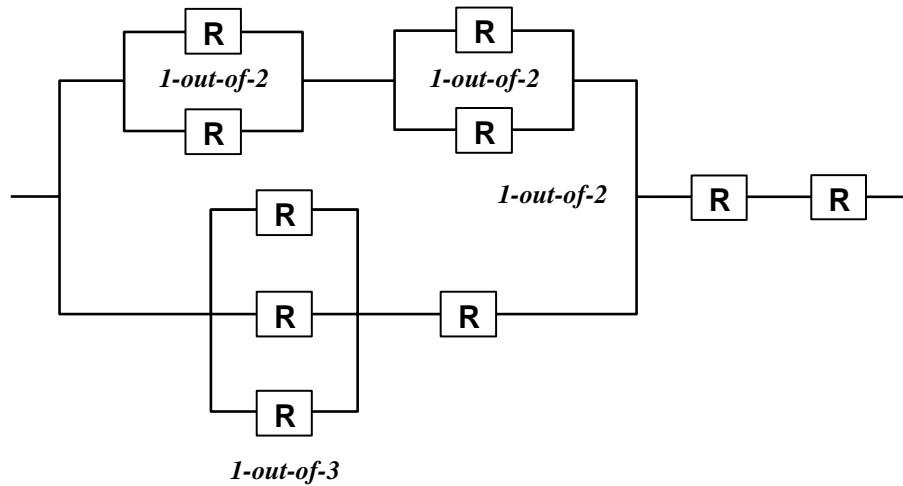
Problems

1. What two inputs must be specified for each component to be placed on the screen?
2. Since Raptor is a unitless tool, what controls the consistency of the inputs?
3. How do you stop adding blocks, events, links, nodes or hierarchies?
4. Can spaces be used in specifying an element's name?
5. What is the maximum number of characters that can be used as an element's name?
6. If the k -information for a node is not set while in the Workspace View, when will Raptor prompt the user for this information?
7. What is the most significant difference between time-truncated and failure-truncated simulations?
8. When might a cycle-termination simulation be conducted?

9. Are the random numbers drawn for each trial different?
10. Explain the Minimum, Mean, Maximum, Standard Dev and SEM for the MDT results in Figure 2.16.
11. Why does the A_0 parameter have more precision than the other parameters shown in Figure 2.16?
12. How long do simulation results remain within Raptor's memory?
13. Determine the mission reliability at 50 hours for the system shown below. The MTBF for components A, B and C are 2,000 hours, 500 hours and 1,000 hours, respectively. All these components fail exponentially and are not repaired.



14. If the MTBF of the exponential components is 75 hours, what is the system reliability at 75 hours if none of the components can be repaired?



15. If the components in Problem 13 are repaired according to a lognormal distribution with a mean of 10 hours and a standard deviation of 1 hour, what is the availability of the system at 75 hours?

Chapter 3



Objectives

1. Understand what failure and repair probability density functions represent.
2. Understand how means and standard deviations are determined for Raptor's probability density functions.
3. Understand which distributions are best for different types of data.
4. Understand how to use the empirical distribution.

Recommended Problems

4, 5, 6, 8, 9, 10, 11 and 12

Recommended Readings

The Exponential Distribution: the Good, the Bad and the Ugly (Murphy, Carter & Brown)

The Exponential Repair Assumption: Practical Impacts (Carter & Malerich)

Statistical Distributions (Evans, Hastings & Peacock), ISBN 0-471-55951-2

Distributions

Open the file labeled **Distributions1.rbd** and notice that the solar panels, batteries, and electrical buses are no longer identical as in Chapter 2. Although the RBD in Figure 3.1 possesses the same topology as the previous chapter's RBD, the final results produced by these two RBDs are radically different.

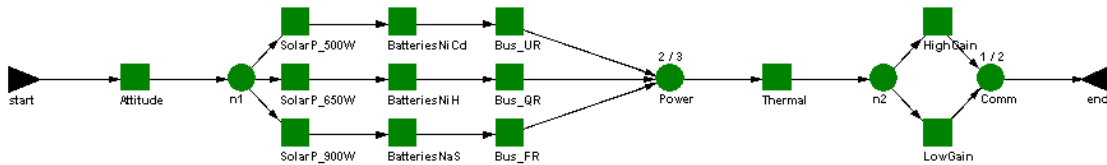


Figure 3.1 Initial RBD for Chapter 3

To understand the differences let us first view the failure and repair distributions of the blocks via the *Options – Tables – View Block Inputs* (shown in Figure 3.2).

Block Name	Failure Distro	Param1	Param2	Param3	Repair Distro	Param1	Param2	Param3
Attitude	Weibull	1.20	4520.0	0.0	Lognormal	10.0	2.0	
BatteriesNaS	Pearson 5	1.70	825.0	0.0	Uniform Real	25.0	45.0	
BatteriesNiCd	Pearson 5	1.70	625.0	0.0	Uniform Real	20.0	80.0	
BatteriesNiH	Pearson 5	1.70	725.0	0.0	Uniform Real	30.0	65.0	
Bus_FR	Exponential	1050.0	0.0		Lognormal	24.0	2.50	
Bus_QR	Exponential	950.0	0.0		Lognormal	36.0	3.50	
Bus_UR	Exponential	850.0	0.0		Lognormal	42.0	6.50	
HighGain	Exponential	350.0	0.0		Uniform Real	96.0	126.0	
LowGain	Exponential	12350.0	0.0		Lognormal	72.0	12.0	
SolarP_500W	Gamma	24.0	34.30	0.0	Normal	23.0	0.870	
SolarP_650W	Gamma	24.0	34.30	0.0	Normal	23.0	0.870	

Fail Distro: Weibull Repair Distro: Lognormal

Shape: 1.200000 Mean: 10.000000

Scale: 4520.000000 Standard Dev: 2.000000

Location: 0.000000

All Hierarchy Filter

Buttons: Help, Cancel, OK, Apply, Print

Figure 3.2 Failure and Repair Distributions for Chapter 3

Notice that this RBD is noticeably more complex, because it has more than the traditional exponential and lognormal failure and repair distributions. In fact, these distributions are sufficiently complex to cause this RBD to be extremely difficult if not implausible to analyze without simulation.

Failure and repair distributions are set in the Block Properties dialog box under the Failure and Repair Distribution tab. As a user selects different distributions, the failure or repair parameter value boxes change to reflect the required input attributes of a particular distribution as shown in Figure 3.3.

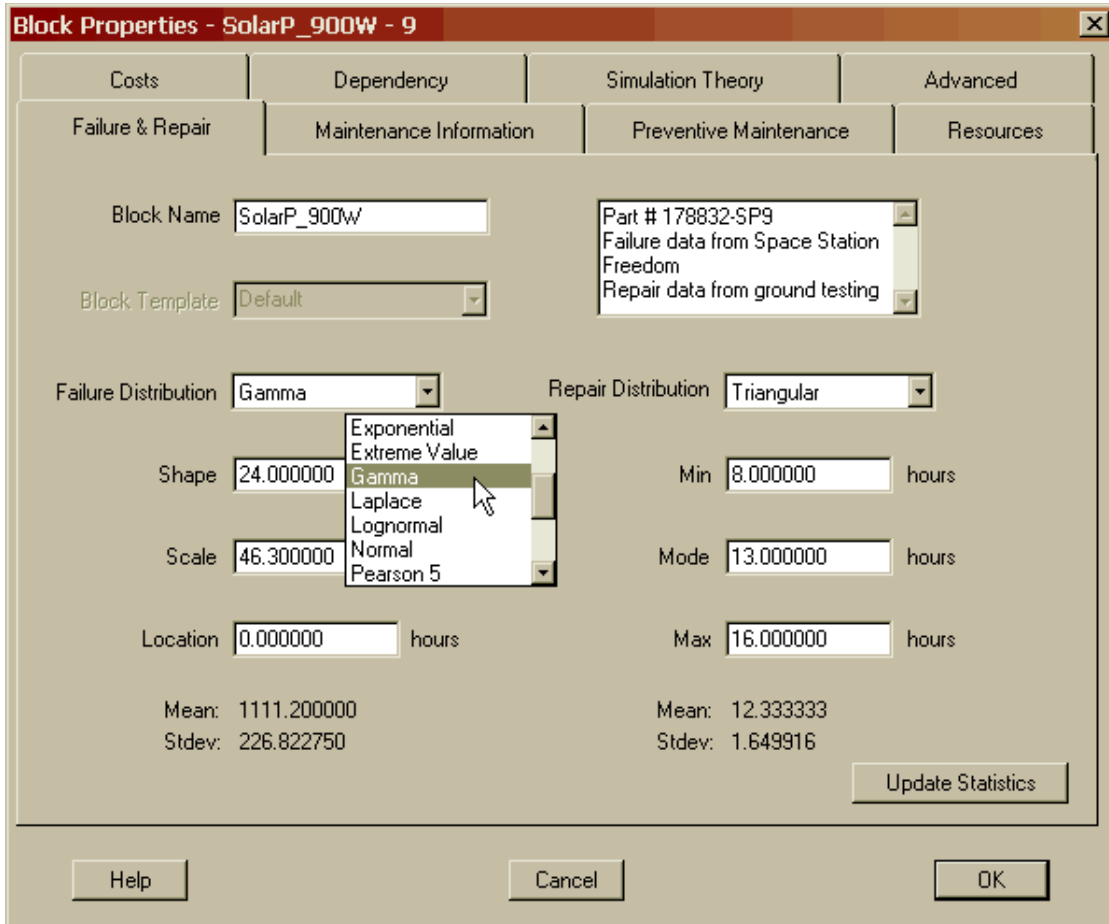


Figure 3.3 Choosing a Failure Distribution

Raptor allows users to choose from eighteen statistical distributions to accurately match data obtained from the field, testing or operations. Additionally, by choosing the None for the repair distribution as shown in Figure 3.4, repairs of a component are prohibited from occurring. Choosing the Fixed distribution (Figure 3.5) for repairs allows a set repair time to be implemented each time a block fails. This distribution is often used when the repair of a block has no variance.

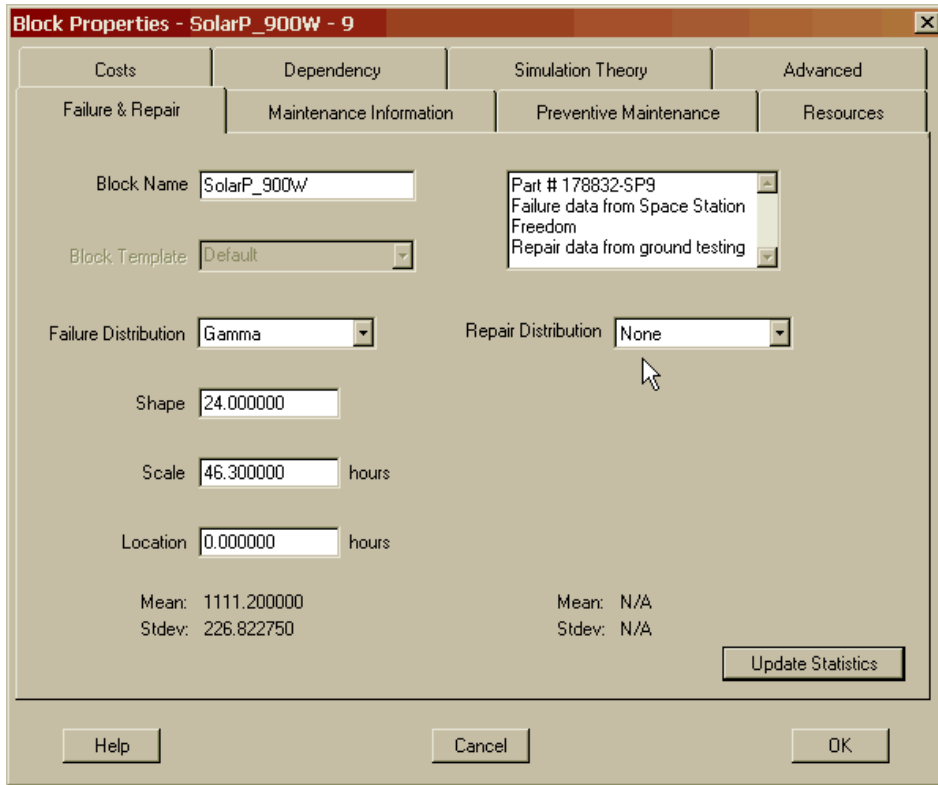


Figure 3.4 Choosing the None Distribution

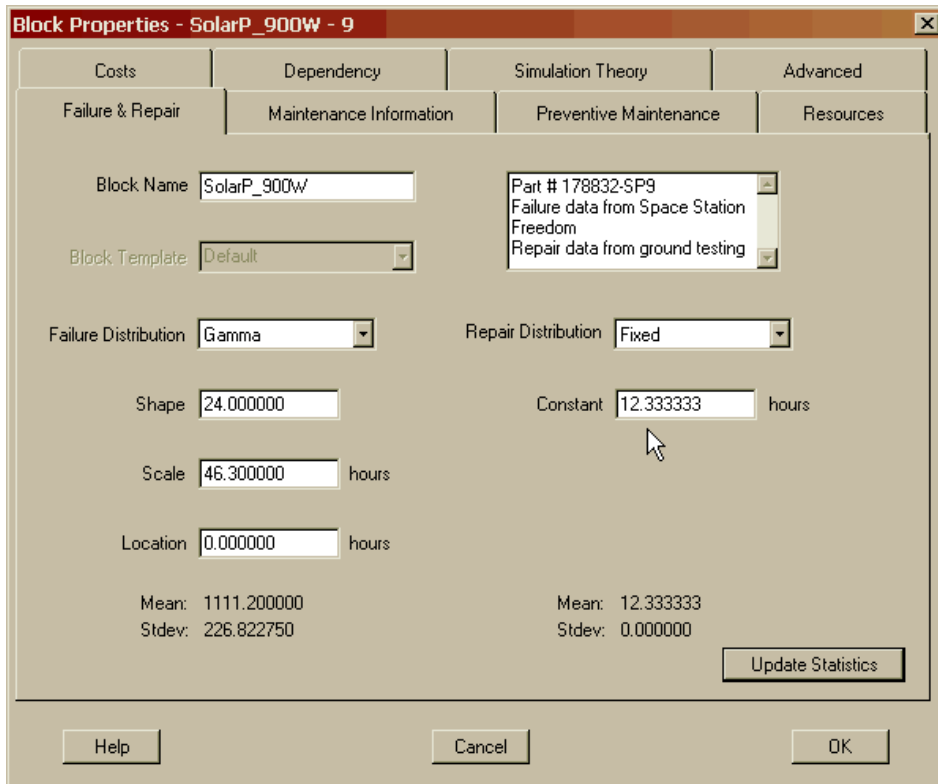


Figure 3.5 Choosing the Fixed Distribution

Ultimately, to design a realistic reliability block diagram, a user must determine which failure and repair distribution is appropriate for each component. There are many statistics books that help facilitate the choice of distribution, but the one most often used by the authors of this workbook is entitled *Statistical Distributions* by Evans, Hastings and Peacock. Unfortunately, even with the availability of such textbooks, choosing a suitable distribution is somewhat of an art that can take years of experience to perform well. A structured approach to this art, however, can be developed by understanding how failure and repair data exists in the reliability community. Data will generally be available from one of three sources as shown in Figure 3.6.

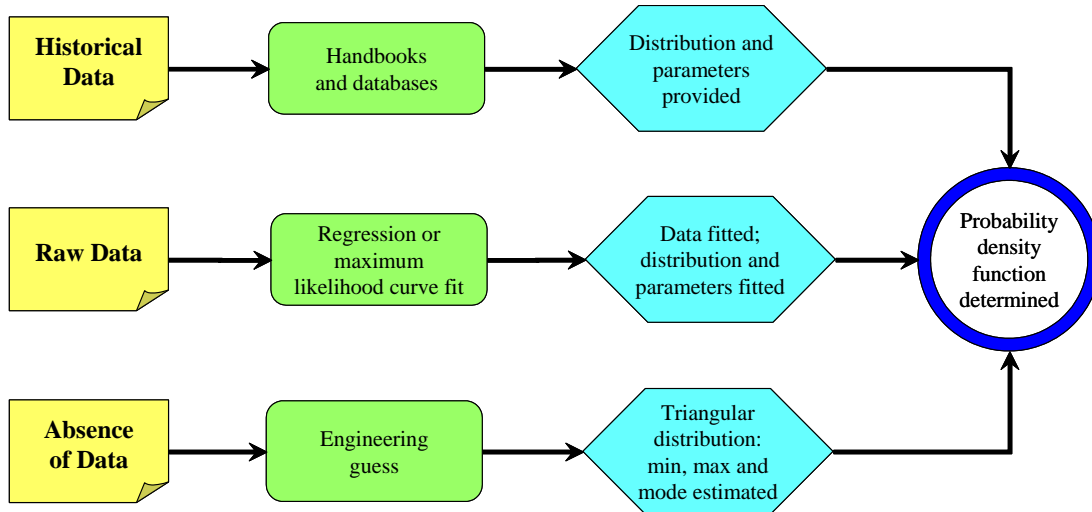


Figure 3.6 Data Sources for Failure and Repair Distributions

If data comes from historical references, the distribution and its key descriptive parameters (e.g., mean, variance, shape, etc.) will be provided. Several organizations collect and then resell databases of failure and repair data to interested parties. These databases are often partitioned by industry. For example, electronic industries will often use the U.S. Government’s Military Handbook 217 while the petrochemical industry will often use their own specific proprietary guide. No matter which guide is used, analysts are using these databases to find similar equipment to their application so that they have a basis for choosing a particular Raptor distribution. A common oversight in using these guides is the failure to consider matching the equipment’s operating environment (e.g., ground benign, uninhabited cargo bay, desert, low Earth orbit, habited-fighter aircraft, cycle of usage, etc.).

If raw data from life-testing or maintenance demonstrations is available, it must first be converted into a special type of histogram in which the area under the histogram sums to unity before it can be implemented within Raptor. Thus, a histogram that has been normalized to reflect the probability of an event (failure or repair) must be established. This special type of histogram is known as a “probability density function”. All of Raptor’s distributions are predefined to be probability density functions that represent the likelihood intensity of a particular failure or repair event occurring. The construction of probability density functions can be accomplished by

hand (if one possesses robust statistical skills) or by curve-fitting software that embeds statistical fitting techniques. These tools use goodness-of-fit tests to guide the choice of a distribution based on supplied data. In general, these curve-fitting tools either use regression or maximum likelihood techniques to determine the best fit. The pertinent output of these curve-fitting tools is to specify a best fit distribution with associated descriptive parameters.

If data is unavailable, choosing a distribution becomes speculative. When historical or actual data does not exist from which to form a basis to choose a failure or repair distribution, the triangular distribution should be considered. Since equipment designers can usually bound the values likely to be exhibited by their equipment, the triangular distribution is a natural choice. That is, a minimum, maximum and most likely value can be established for any failure or repair situation by those individuals who either have built or maintained the equipment. This method is often known as implementing the best *engineering guess*. If a particular engineering guess is considered questionable, Raptor can be used to iterate to a more accurate guess by varying the inputs of the triangular distribution and then analyzing the impacts to the resulting outputs.

The nomenclature, equation for the mean, representative illustrations (Figures 3.7 through 3.24) and a brief description of Raptor’s eighteen built-in failure and repair probability density functions follows:

Beta (shape ν , shape2 ω)

$$\text{Mean: } \frac{\nu}{\nu + \omega}$$

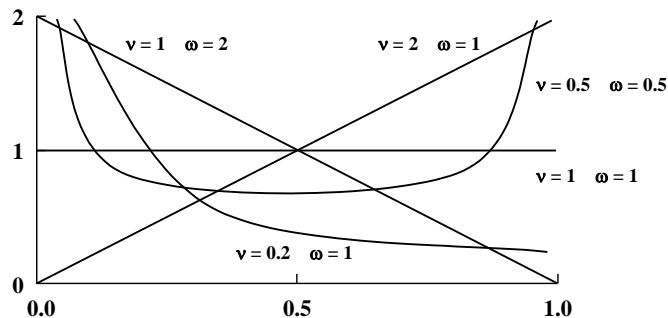


Figure 3.7 Beta Density Functions

- Continuous
- Bathtub curve shape when ν and ω equal one-half
- Data always range from zero to one
- Often used when data is comprised of p-values
- Example of use: probability of kill data; probability of success data

Binomial (trials n , probability p)

Mean: np

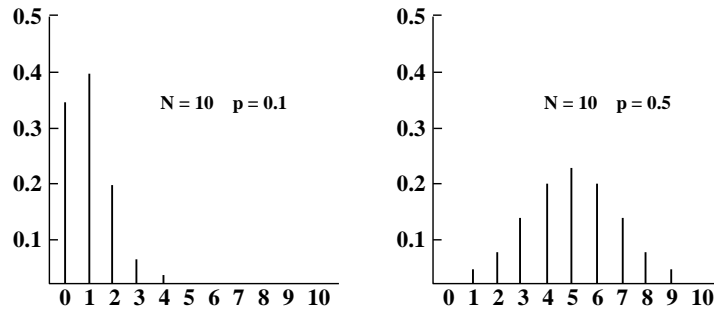


Figure 3.8 Binomial Density Functions

- Discrete
- Probability of number of successes occurring
- Often used in statistics in a cumulative fashion
- p -value must remain constant from trial to trial
- Varying p -value is an indication of the multinomial distribution
- Example of use: probability of x missiles hitting a target

Chi-squared (shape ν)

Mean: ν

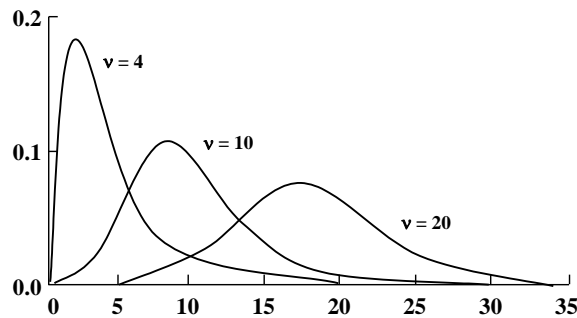


Figure 3.9 Chi-squared Density Functions

- Continuous
- Related to the gamma and normal distributions
- Often used in statistics in confidence bound determination
- Example of use: repair data

Empirical (n number of data points)

Mean: $\frac{\sum_{i=1}^n x_i}{n}$

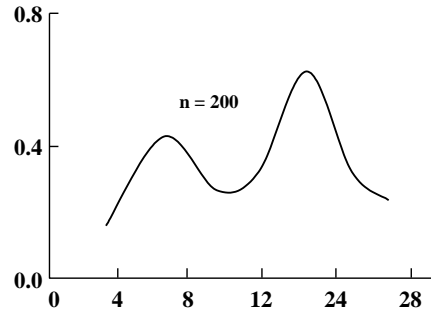


Figure 3.10 Empirical Density Function

- Continuous
- Raw data is discrete
- Information about the tails is often missing
- Example of use: multi-modal data; small data sample sizes

Erlang (shape c, scale b)

Mean: bc

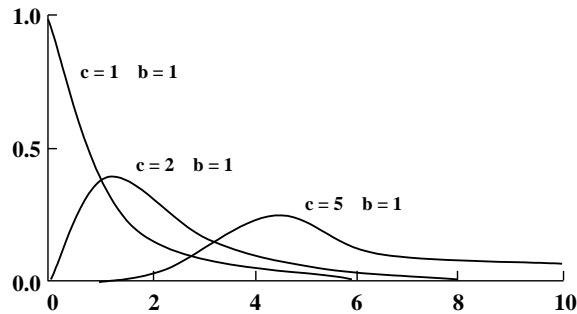


Figure 3.11 Erlang Density Functions

- Continuous
- Related to the gamma distribution when shape parameter is an integer
- Can be used to represent the sum of independent exponential variants
- Example of use: failure data; repair data

Exponential (mean θ , location π)

Mean: $\theta + \pi$

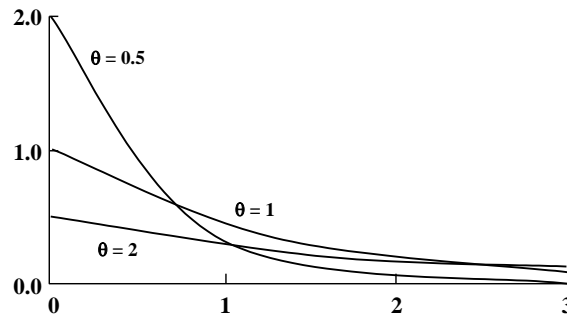


Figure 3.12 Exponential Density Functions

- Continuous
- Related to the Weibull and Poisson distributions
- Raptor failure distribution default
- Raptor allows the entering of exponential data in mean or lambda mode
- Most commonly used failure distribution in reliability industry
- Possesses the memoryless property
- Location parameter most commonly used in warranty analysis
- Example of use: failure data of electronic components

Extreme Value (location a , scale b)

Mean: $a - b\Gamma'(1)$

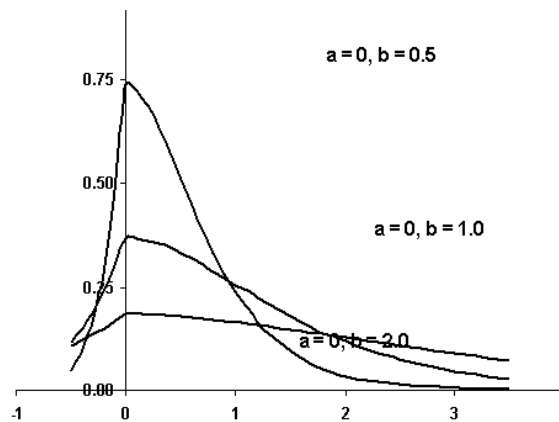


Figure 3.13 Extreme Value Density Functions

- Continuous
- Portion of curve lies within the negative domain; warrants caution
- Example of use: extreme data sources

Gamma (shape c , scale b , location π)

Mean: $bc + \pi$

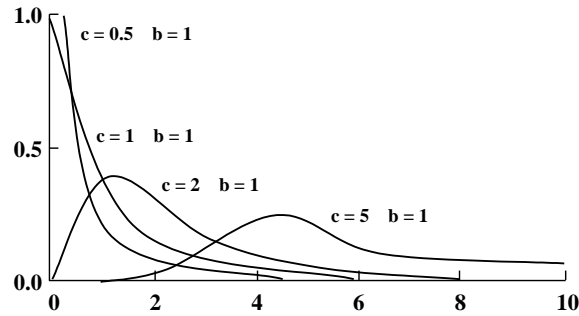


Figure 3.14 Gamma Density Functions

- Continuous
- Fundamental distribution that describes most shapes
- Location parameter most commonly used in warranty analysis
- Example of use: failure data; repair data

Laplace (location a , scale b)

Mean: a

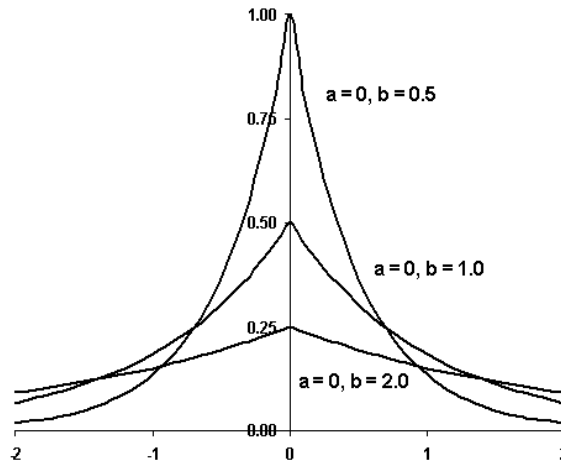


Figure 3.15 Laplace Density Functions

- Continuous
- Portion of curve lies within the negative domain; warrants caution
- Often known as the double exponential distribution
- Example of use: repair data and miss distances

Lognormal (mean m , standard deviation σ)

Mean: m

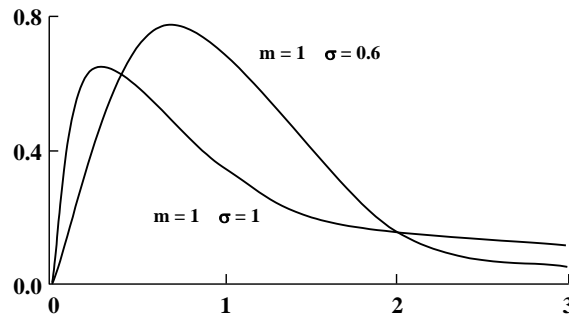


Figure 3.16 Lognormal Density Functions

- Continuous
- Related to the Normal distribution
- Raptor repair distribution default
- Raptor allows the entering of lognormal data in normal and log mode
- Extremely good fit for repair data of most systems
- Example of use: repair data

Normal (mean μ , standard deviation σ)

Mean: μ

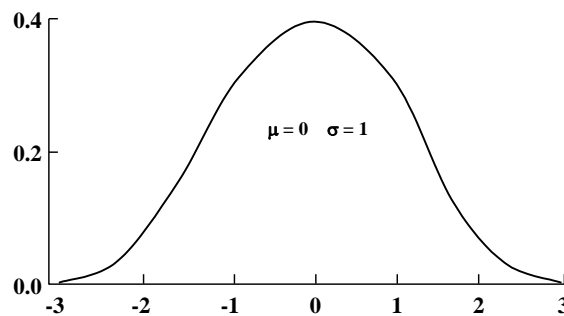


Figure 3.17 Normal Density Function

- Continuous
- Related to the Lognormal distribution
- Also known as the Bell Curve or Gaussian distributions
- Extremely good for data involving measurements
- Portion of curve lies within the negative domain; warrants caution
- Example of use: failure data; repair data

Pearson V (shape α , scale β , location π)

$$\text{Mean: } \frac{\beta}{\alpha - 1} + \pi$$

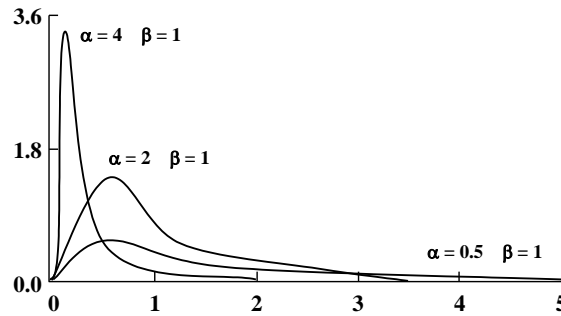


Figure 3.18 Pearson V Density Functions

- Continuous
- Similar to Lognormal distribution
- Used when small repair times dominate the repair data set
- Location parameter used when minimum repair is greater than zero
- Example of use: repair data

Pearson VI (shape α_1 , shape2 α_2 , scale β)

$$\text{Mean: } \frac{\beta \alpha_1}{\alpha_2 - 1}$$

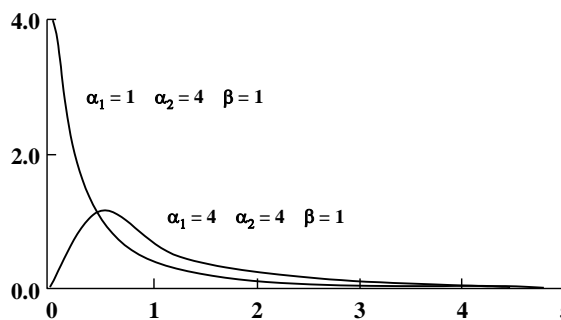


Figure 3.19 Pearson VI Density Functions

- Continuous
- Similar to Lognormal distribution
- Used when small repair times excessively dominate the repair data set
- Example of use: repair data; computer re-boots

Poisson (mean λ)

Mean: λ

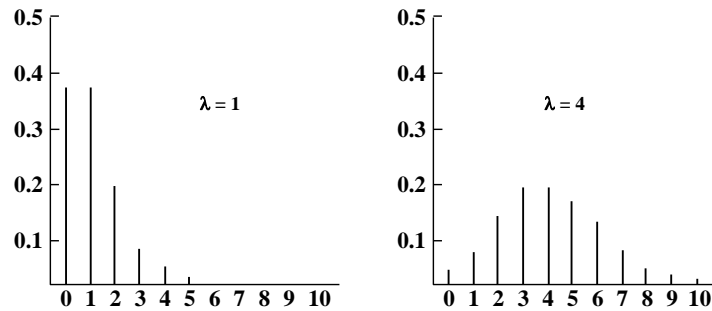


Figure 3.20 Poisson Density Functions

- Discrete
- Related to the exponential distribution
- Probability of x arrivals in given time interval
- Often used in queuing theory
- Often used in optimization of sparing levels
- Example of use: probability of x failures occurring in a Δt

Triangular (minimum a , mode c , maximum b)

Mean: $\frac{a + b + c}{3}$

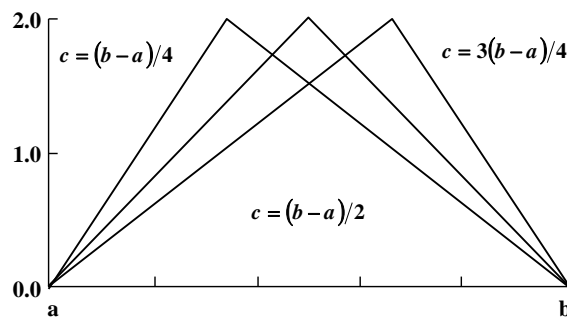


Figure 3.21 Triangular Density Functions

- Continuous
- Best distribution when information about a component does not exist
- Mode is the most frequently occurring value of the density function
- Example of use: failure data; repair data

Uniform Integer (minimum i, maximum j)

Mean: $\frac{i+j}{2}$

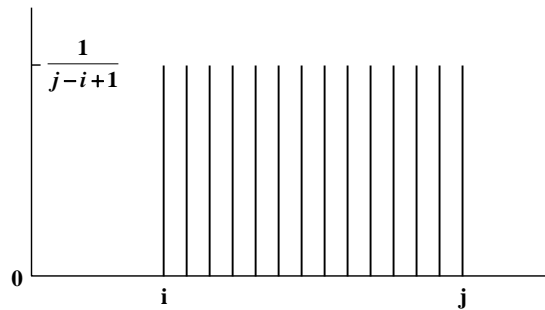


Figure 3.22 Uniform Integer Density Function

- Discrete
- Used when probability of occurrence is equally likely within a range
- Often used to model queuing lines and dice
- Example of use: repair data

Uniform Real (minimum i, maximum j)

Mean: $\frac{i+j}{2}$

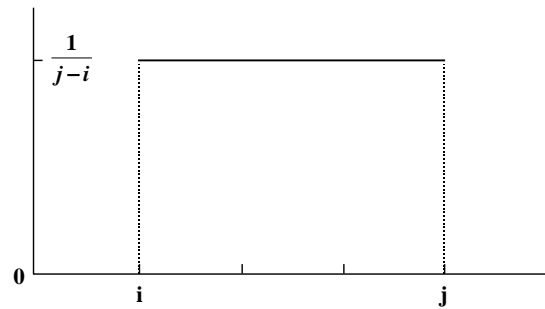


Figure 3.23 Uniform Real Density Function

- Continuous
- Used when probability of occurrence is equally likely within a range
- Fundamental distribution from which all other distributions are derived
- Often used to model queuing lines
- Example of use: repair data

Weibull (shape β , scale η , location π)

$$\text{Mean: } \eta \Gamma\left(\frac{\beta+1}{\beta}\right) + \pi$$

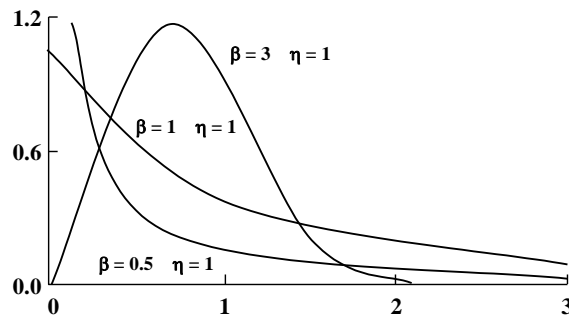


Figure 3.24 Weibull Density Functions

- Continuous
- Related to the exponential and Erlang distributions
- Fundamental distribution that describes most shapes
- Can take on all aspects of the bathtub curve but not simultaneously
- Location parameter most commonly used in warranty analysis
- Example of use: failure data; particularly wear-in and out; repair data

Sometimes the data does not fit any “standard” statistical distribution well. The frequency histogram of failure data for a hypothetical piece of equipment shown in Figure 3.25 is an example of this concern.

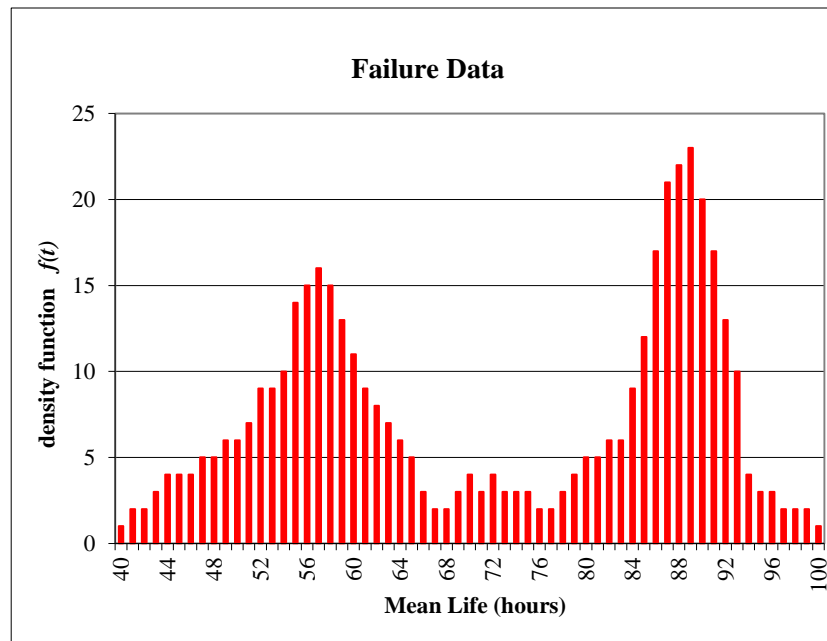


Figure 3.25 Histogram of Bimodal Data

This data is bimodal and does not fit any of the traditional statistical distributions used for reliability analysis (the reader might wish to verify this by reviewing Figures 3.7 through 3.24). By choosing Raptor's empirical distribution, a failure or repair density function will be generated based on user specified data. Data outside the range of values entered, however, cannot be represented by the empirical distribution. The data can be entered by hand, loaded from a file, or a combination of both methods may be used.

Change the repair distribution of the Thermal block using the following empirical data shown in Figure 3.26. When you choose the empirical distribution from within the Block Properties dialog box, another dialog box appears entitled Empirical Data. This box awaits data input from the user as shown in Figure 3.27. Enter each data point and click the accept button to load a data point into the empirical data table.

7.3	23.5	22.0	6.0
21.7	15.5	13.0	9.2
6.9	17.7	24.0	9.0

Figure 3.26 Empirical Repair Data

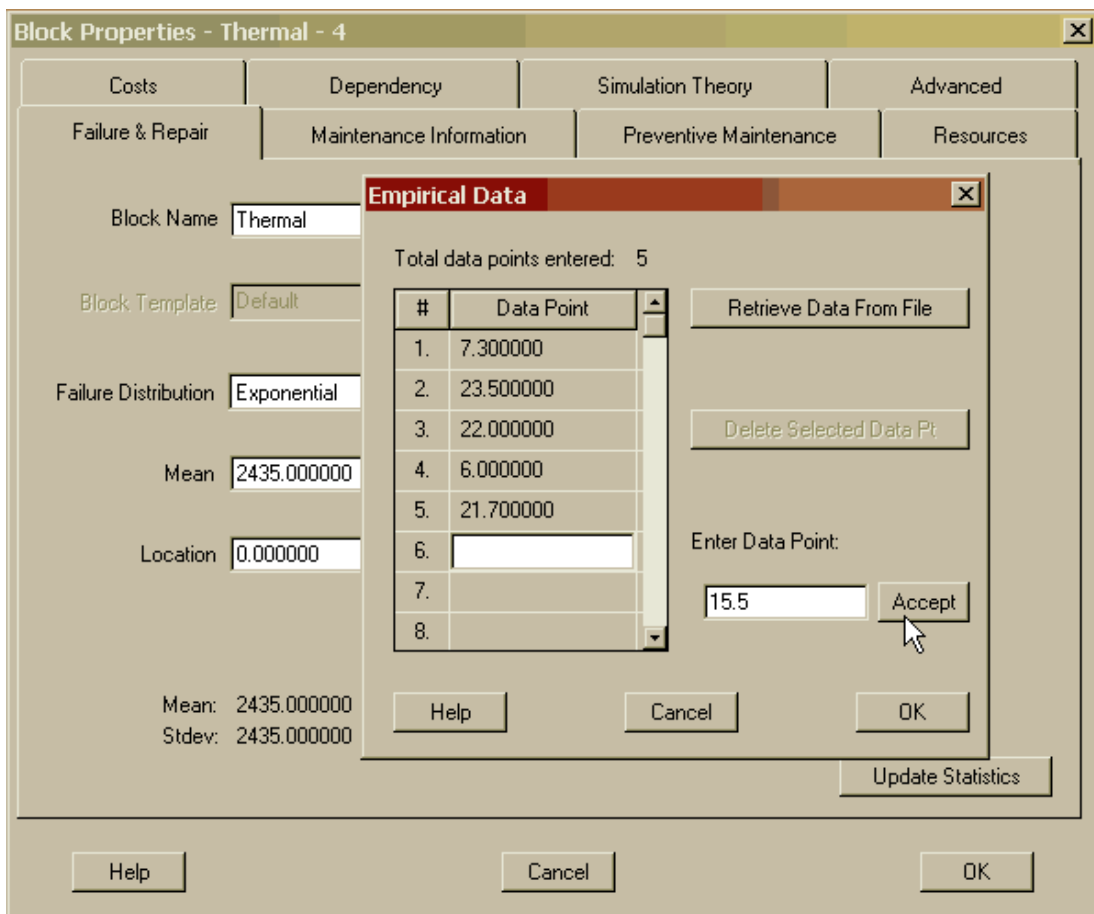


Figure 3.27 Empirical Data Entry

After entering the twelve repair data points, click the OK button of the Empirical Data dialog box and then the Update Statistics button of the Block Properties dialog box. If all data was entered correctly, the mean repair time for the Thermal block should be as shown in Figure 3.28.

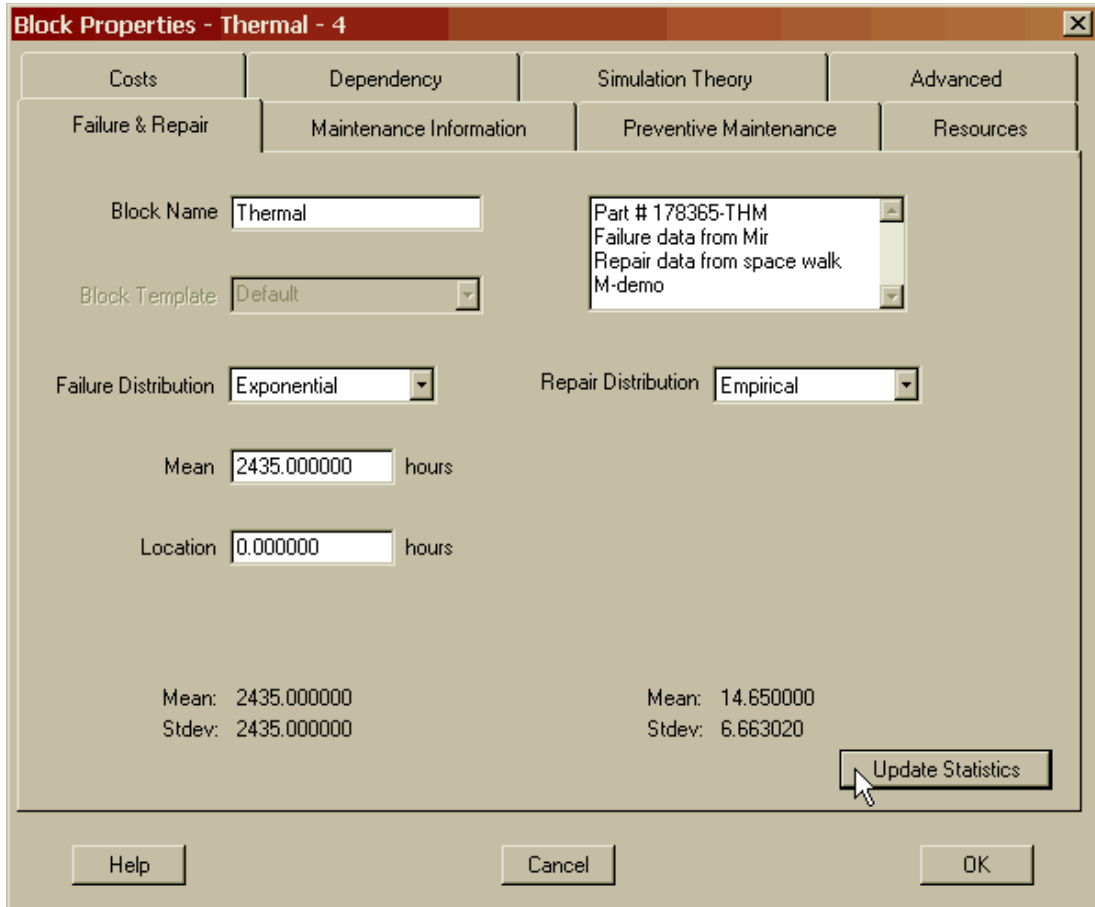


Figure 3.28 Empirical Repair Mean

If a file is used to load empirical data for a component, it must be in the standard “txt” format. It must be comprised of a single column of data without any delimiters or end-of-file markers. A single column of data brought into Excel®, Access®, or any other database-type tool that can convert data into a “txt” file, will produce a file that can be read by Raptor’s empirical algorithm. The maximum number of data points that can be read into Raptor for *each* empirical failure or repair distribution is 999.

Open the Block Input Tables dialog box either by the View Block Inputs button or via the *Options – Tables – View Block Inputs* menu item. Figure 3.29 displays the Thermal block with an empirical repair distribution. Notice that the empirical distribution does not display any parameter values like the other distributions. Conduct a time-truncated simulation (ten trials of 730.5 hours each) and note the results shown in Figure 3.30.

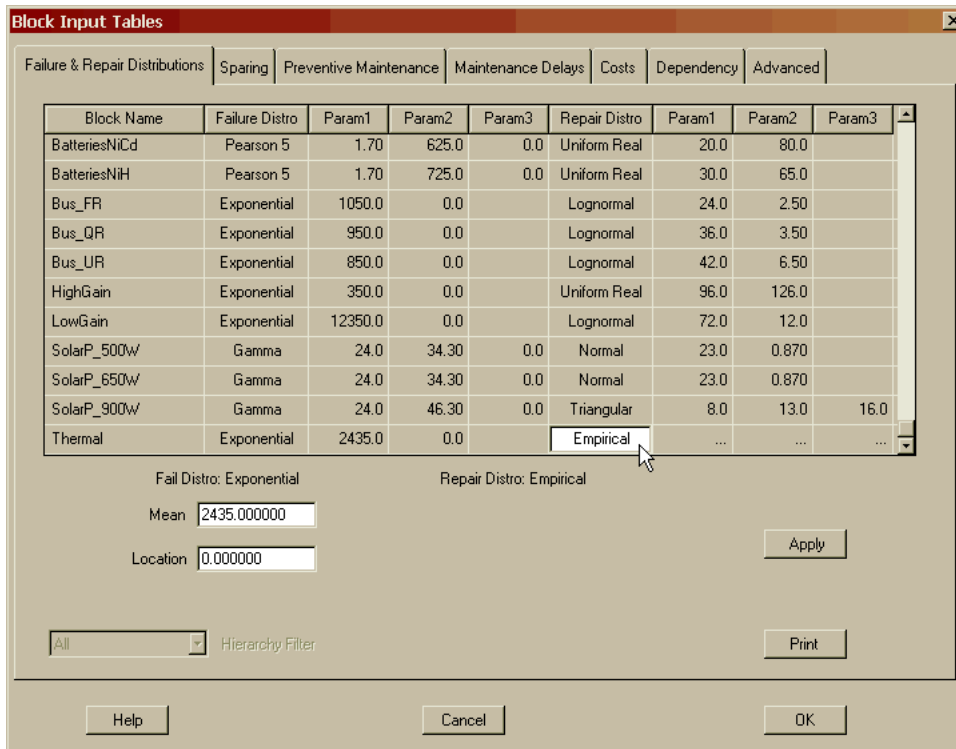


Figure 3.29 Empirical Distribution Displayed in Input Tables

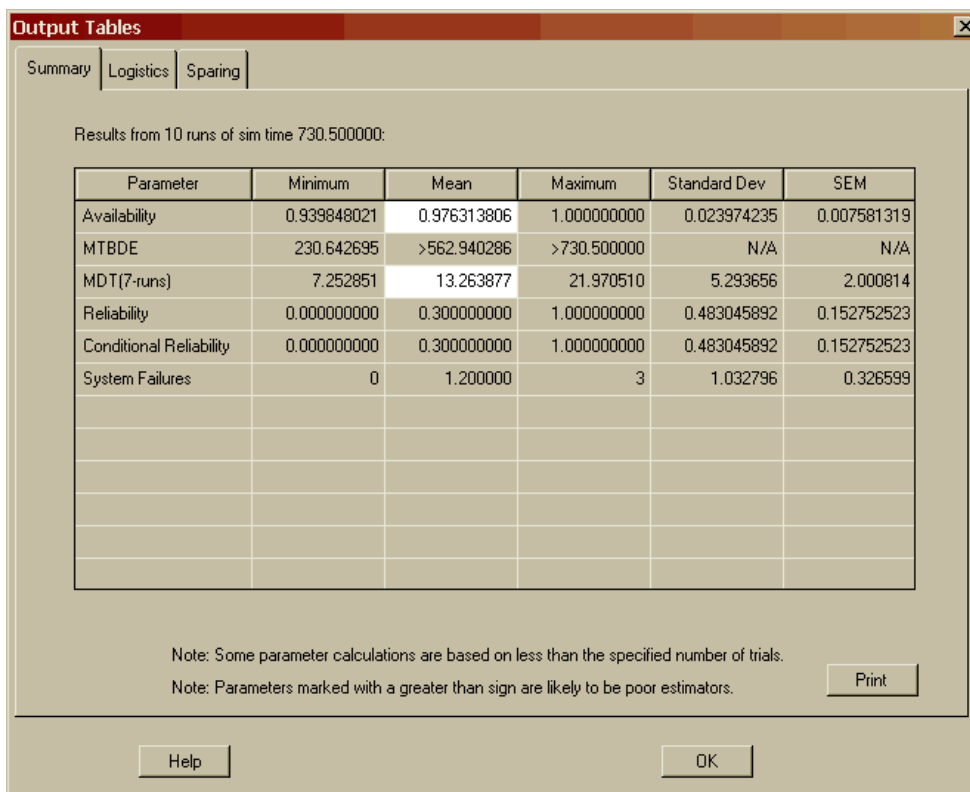
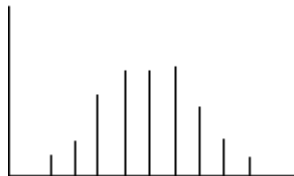


Figure 3.30 Results for Chapter 3

Problems

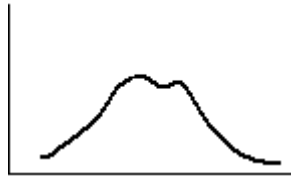
1. Can the failure or repair distributions of a component be changed within the Fail & Repair Distributions table of the Input Tables dialog box?
2. List and describe two types of systems that might have components that use the None repair distribution.
3. What is the difference in a component having the None repair distribution opposed to having the Fixed 0.0 repair distribution?
4. How many Raptor distributions have a failure parameter that equals the mean of the probability density function?
5. Name a major drawback for using the empirical function.
6. Why is the MDT parameter displayed in Figure 3.30 only based on seven trials when ten were conducted?
7. Why does the MTBDE parameter have a greater-than sign in front of the mean value of 562.94 in Figure 3.30?
8. Why is the MTBDE's standard deviation listed as "N/A" in Figure 3.30?
9. A histogram of data with a mean of 75.4 and a standard deviation 74.7 is shown below. Which distribution might be the best to describe this data?



10. A histogram of 30 data points is shown below. The data is a collection of target miss-distances. Which distribution might be the best to describe this data?



11. A histogram of 300 data points is shown below. The data is a collection of target miss-distances. Which distribution might be the best to describe this data?



12. Take the data displayed below and make a txt file (Excel, NotePad, etc.) and load it into the repair distribution of the Thermal component for the **Distribution1.rbd**. Conduct a simulation and confirm that the A_o is 0.976734136 and the MDT is 13.019976.

7.3	23.5	22.0	6.0
21.7	15.5	13.0	9.2
6.9	17.7	24.0	9.0
23.5	19.5	20	13.8
6.2	12	6.9	12.4
8	10.1	6.1	22

Chapter 4



Objectives

1. Distinguish between the different types of replacement strategies.
2. Create a spare and resource pool.
3. Understand the logistical information in the Sparing tab of the Block Input Tables dialog box.
4. Understand the logistical information in the Maintenance Delays tab of the Block Input Tables dialog box.
5. Understand the information in the Logistics tab of the Output Tables dialog box.
6. Understand the information in the Sparing tab of the Output Tables dialog box.
7. Understand the Raptor repair cycle.

Recommended Problems

1, 2, 3, 4, 5, 6, 7, 9 and 11

Logistics Constraints

Logistics, or lack thereof, can often cause a system to be unavailable. Disregarding logistics in a simulation model can produce low fidelity results. This chapter will focus on how to implement logistic constraints within Raptor. Open **Logistics1.rbd**, look at the Block Input Tables dialog box and focus on the second tab, labeled "Sparing", shown in Figure 4.1.

Block Name	Source of Spares	Stock	New	Arrive at	Order	# of	Arrive at	Emergency
Attitude	Infinite	N/A	N/A	N/A	N/A	N/A	N/A	N/A
BatteriesNaS	Cells	3	1	730.0	0	1	360.0	168.0
BatteriesNiCd	Cells	3	1	730.0	0	1	360.0	168.0
BatteriesNiH	Cells	3	1	730.0	0	1	360.0	168.0
Bus_FR	Electronics	2	N/A	N/A	N/A	N/A	N/A	24.0
Bus_QR	Electronics	2	N/A	N/A	N/A	N/A	N/A	24.0
Bus_UR	Electronics	2	N/A	N/A	N/A	N/A	N/A	24.0
HighGain	Infinite	N/A	N/A	N/A	N/A	N/A	N/A	N/A
LowGain	Infinite	N/A	N/A	N/A	N/A	N/A	N/A	N/A
SolarP_500W	Infinite	N/A	N/A	N/A	N/A	N/A	N/A	N/A
SolarP_650W	Infinite	N/A	N/A	N/A	N/A	N/A	N/A	N/A
SolarP_900W	Infinite	N/A	N/A	N/A	N/A	N/A	N/A	N/A
Thermal	Custom	1	1	1460.0	N/A	N/A	N/A	48.0

Figure 4.1 Spare or Replacement Strategies

Only blocks possess sparing or replacement strategies and the three types of replacement strategies are Infinite, Pool and Custom sparing. The words "Infinite" and "Custom" appear in the second column of the Sparing tab, but a block belonging to a spare pool displays the name of the pool in the second column.

A replacement strategy is an algorithm that specifies the type of spare that will be used to fix a broken component. A fundamental Raptor assumption is that a spare is always an identical replacement of the particular component that failed. A component can be assigned to only one of the three replacement strategies. First, components that implement an infinite replacement strategy will never run out of spares, since their stockpile is by definition infinitely large. When a component utilizing infinite sparing fails, a spare is immediately available to begin the repair process, barring any other logistics constraints (e.g., labor, repair equipment, etc). Second, components that belong to a spare pool are in competition with other components trying to acquire

spares from the very same pool. This competition may delay a component from beginning its hands-on maintenance portion of its repair cycle. Lastly, components designated with the custom sparing replacement strategy are guaranteed spares (again, barring other logistical constraints) designated for particular components. None of the replacement strategies attempt to fix the broken components; they are simply discarded after replacement. It is important to understand that custom sparing is different from spare pools in that the former applies to a single component, while the latter applies to multiple components. This may seem like a trivial difference, but it is rather important since a component using custom sparing will never be subject to competition with other components for its spares.

Logistical constraints other than the lack of spares that might interfere with the completion of a component’s repair cycle include lack of resources, logistic delays, preventive maintenance and dependency. This chapter will cover resources and logistical delays while preventive maintenance and dependency will be discussed separately in Chapters 6 and 7 respectively, due to the complexity of these two subjects. For this particular RBD, a summary of the resource and logistical delay constraints are listed in the Maintenance Delays tab of the Block Input Tables dialog box shown in Figure 4.2.

Block Name	Pre-Logistics	Post-Logistics	Resource	# for Mx	# for PM
Attitude	Fixd (2.000000)	Fixd (1.000000)	Astronauts	3	1
BatteriesNaS	Fixd (0.150000)	Fixd (0.150000)	Astronauts	1	1
BatteriesNiCd	Fixd (0.250000)	Fixd (0.250000)	Astronauts	3	1
BatteriesNiH	Fixd (0.250000)	Fixd (0.250000)	Astronauts	2	1
Bus_FR	Fixd (2.500000)	Fixd (0.000000)	Astronauts	3	1
Bus_QR	Fixd (2.000000)	Fixd (0.000000)	Astronauts	4	1
Bus_UR	Fixd (3.000000)	Fixd (0.000000)	Astronauts	5	1
HighGain	Fixd (2.000000)	Fixd (1.000000)	Astronauts	1	1
LowGain	Fixd (0.500000)	Fixd (0.500000)	Astronauts	1	1
SolarP_500W	Fixd (2.000000)	Fixd (0.000000)	Astronauts	3	1
SolarP_650W	Fixd (2.000000)	Fixd (0.000000)	Astronauts	3	1
SolarP_900W	Fixd (1.000000)	Fixd (0.000000)	Astronauts	2	1
Thermal	Fixd (0.000000)	Fixd (0.000000)	Astronauts	2	1

Figure 4.2 Maintenance Delays Properties

Notice that the pre and post repair logistics delays have been specified as fixed lengths of time. Both can be specified as random variables, however, using any of Raptor's eighteen built-in distributions. Notice also that Figure 4.2 indicates that resources for each component are needed to conduct a repair. Double-click on the Thermal block and select the second tab, labeled Maintenance Information, as shown in Figure 4.3.

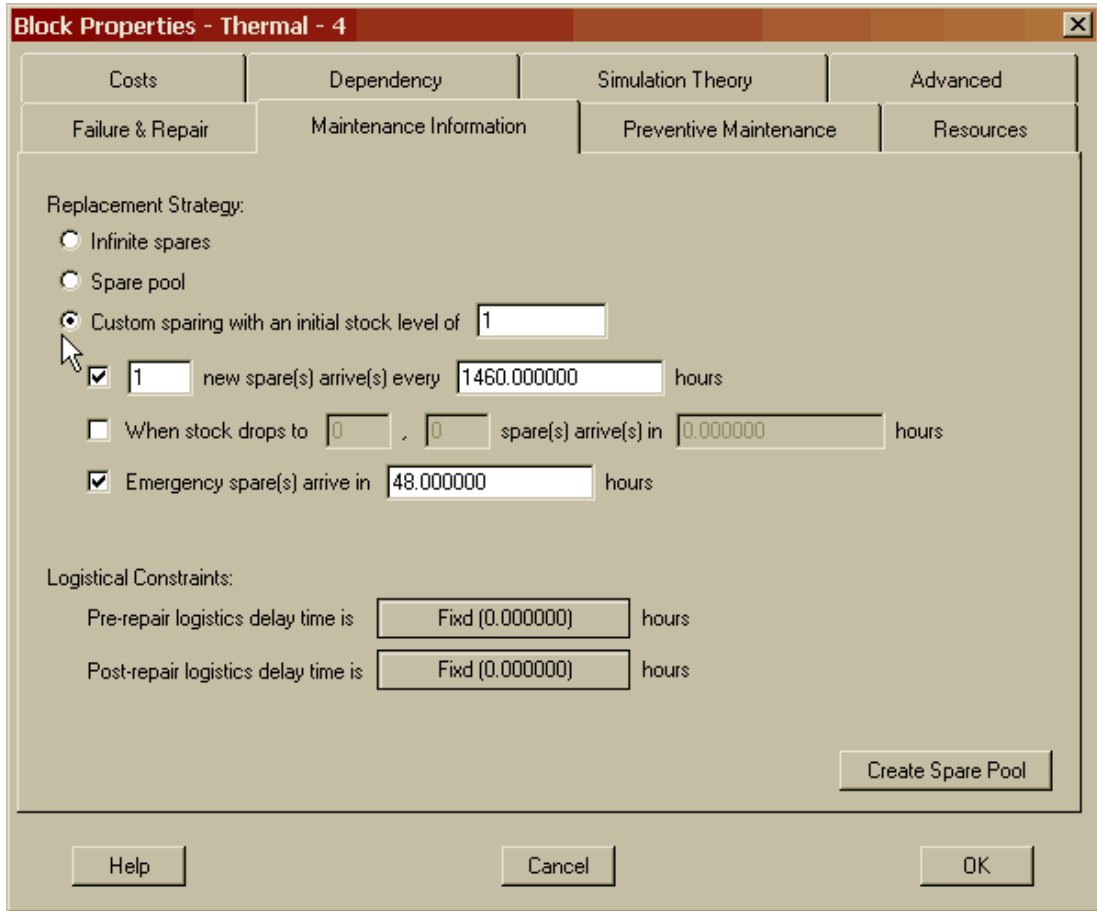


Figure 4.3 Thermal's Maintenance Information

This tab allows a component to be assigned to one of three types of replacement strategies as well as two fixed or random logistical delays. Figure 4.3 indicates that the Thermal component is implementing custom sparing with two of three options utilized; although any combination of these three options can be specified. When custom sparing is selected, an initial stockpile that is available at the beginning of a simulation trial must be chosen even if it is initially zero. The first option allows for a number of new spares to arrive at a fixed frequency. The second option allows for a number of new spares to arrive after a component's stock level has reached a given level and a specified time has passed. The last option allows a spare to arrive after a specified time when the component's stock level is emptied and the demand for another spare has been made by a component.

Observe that the default for pre and post logistics delays is the fixed distribution with a mean of zero. Thus, the Thermal component's repair cycle is only composed of hands-on maintenance whose length was specified on the Failure & Repair tab of the Block Properties dialog box. Let us change the pre-repair logistic delay time to be represented by the triangular distribution as shown in Figure 4.4. By selecting the pre repair logistics delay button, the Define Distribution dialog box appears on top of the Block Properties dialog box. Throughout Raptor, if a distribution is displayed on a button, it can be modified by simply clicking on the button and displaying the Define Distribution dialog box. Additionally, whenever one dialog box appears on top of another, the one on top has *focus* and must be resolved (i.e., selecting the OK or Cancel buttons) before the dialog box on the bottom can be manipulated.

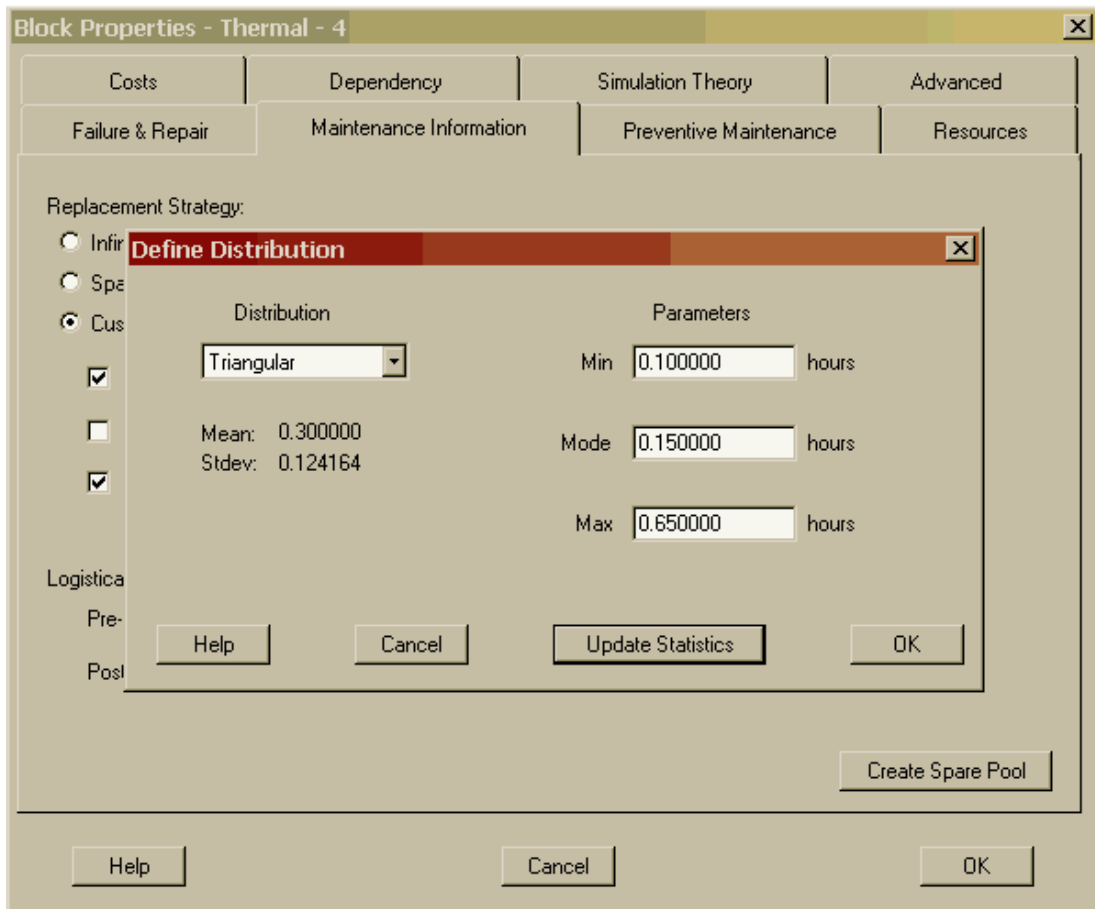


Figure 4.4 Setting the Pre Logistics Delay to be Random

We have selected a triangular distribution with a minimum of 0.10 hours, a maximum of 0.65 hours and a mode (or most likely value) of 0.15 hours. While the Define Distribution dialog box is displayed, select the Update Statistics button to determine the mean and standard deviation of this particular triangular distribution. Thus, on average eighteen minutes (0.30 hours) of pre logistic delay time is needed before hands-on maintenance can be started (i.e., assuming spare and resource constraints are met). Modify the properties of the post-repair logistics delay time in the same manner

as the pre-repair logistic delay time such that the Thermal's Block Properties dialog box appears as shown in Figure 4.5.

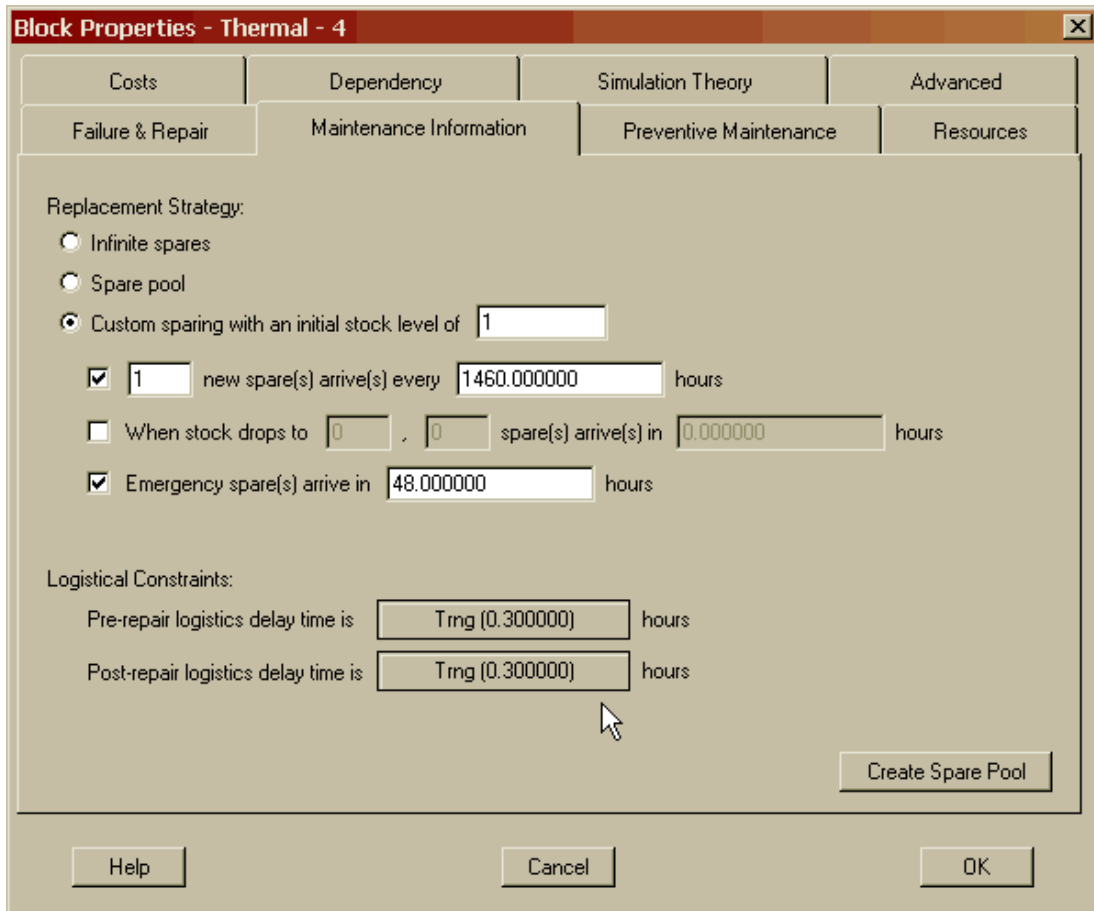


Figure 4.5 Random Pre and Post Logistics Delays

On average, eighteen minutes (0.30 hours) of post logistic delay time is needed after hands-on maintenance is completed before the component can be considered repaired. Until the post-repair logistics delay has expired, a component cannot begin to operate. Pre and post logistical delays are common elements of a component's repair cycle and examples of these types of delays include: acquiring repair personnel or tools, driving to or from a repair location, safety inspection of a repaired component, functional testing of a component and paperwork required to be filled out after a repair is completed.

Double-click on the nickel-cadmium batteries (BatteriesNiCd) component and select the second tab labeled Maintenance Information. Figure 4.6 displays an example of a component that uses the spare pool replacement strategy. Notice that a component can be assigned to any of the spare pools listed in the spare pools drop down list box. An example of infinite sparing can be observed by double-clicking on the 500-watt solar panel (SolarP_500W) component and selecting the second tab as shown in Figure 4.7.

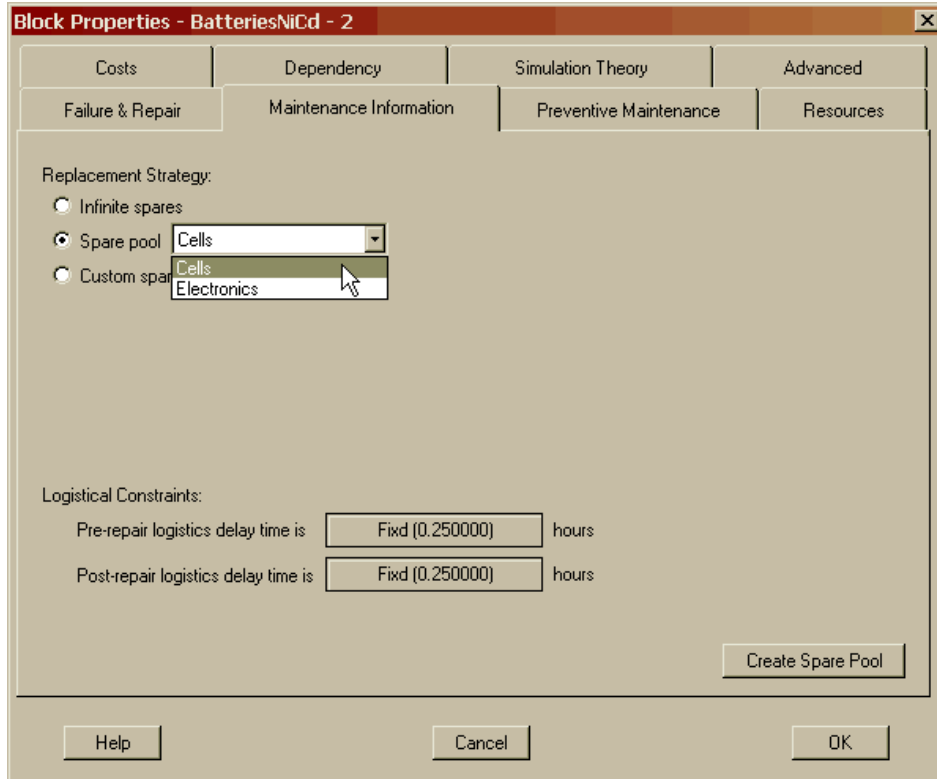


Figure 4.6 Nickel-Cadmium's Maintenance Information

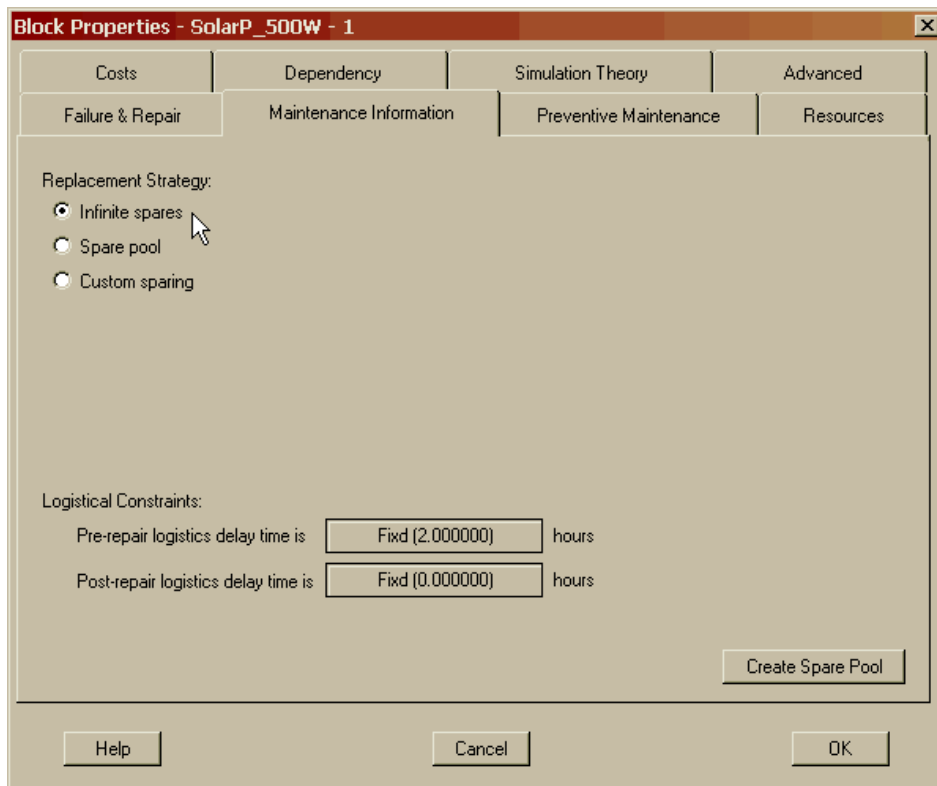


Figure 4.7 500 Watt Solar Panel's Maintenance Information

Let us change this component's replacement strategy from infinite to spare pool by clicking on the Create Spare Pool button, which results in the Spare Pools dialog box being displayed on top of the 500-watt solar panel's Block Properties dialog box. Create a spare pool called PhotoVPanels (i.e., photovoltaic panels) with the attributes shown in Figure 4.8. There will be an initial stock pile of two spares and one new spare will arrive approximately every six months from a shuttle re-supply mission. If needed, an emergency spare can be acquired in three days from a dedicated launch of a Spaceship One type vehicle. The stock level ordering sparing strategy was determined to be infeasible for this component and therefore was not engaged.

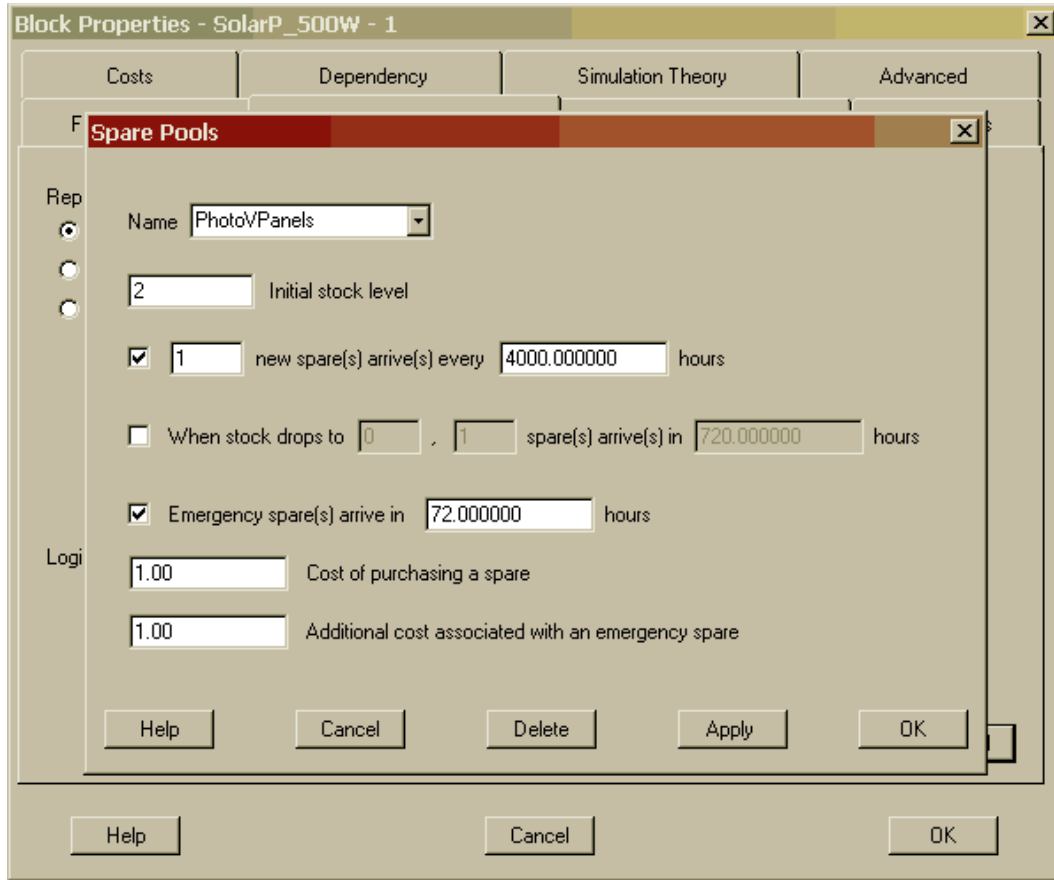


Figure 4.8 Spare Pools Dialog Box via Create Spare Pool Button

You must click the Apply button to register the creation of a spare pool each time a new one is generated. Once you have clicked the Apply button, the Spare Pools dialog box is reset to allow the development of more pools if desired. After creating the photovoltaic spare pool cancel from this dialog box and change this component from infinite sparing to the photovoltaic spare pool as shown in Figure 4.9. Set both of the other solar panels (i.e., the 650 and 900 watt components) to acquire spares from the photovoltaic spare pool as well. Before continuing on in this chapter, view the Sparing tab from the Block Input Tables dialog box and confirm that your table looks exactly like the one displayed in Figure 4.10 (specifically all three solar panel components have the same spare pool attributes).

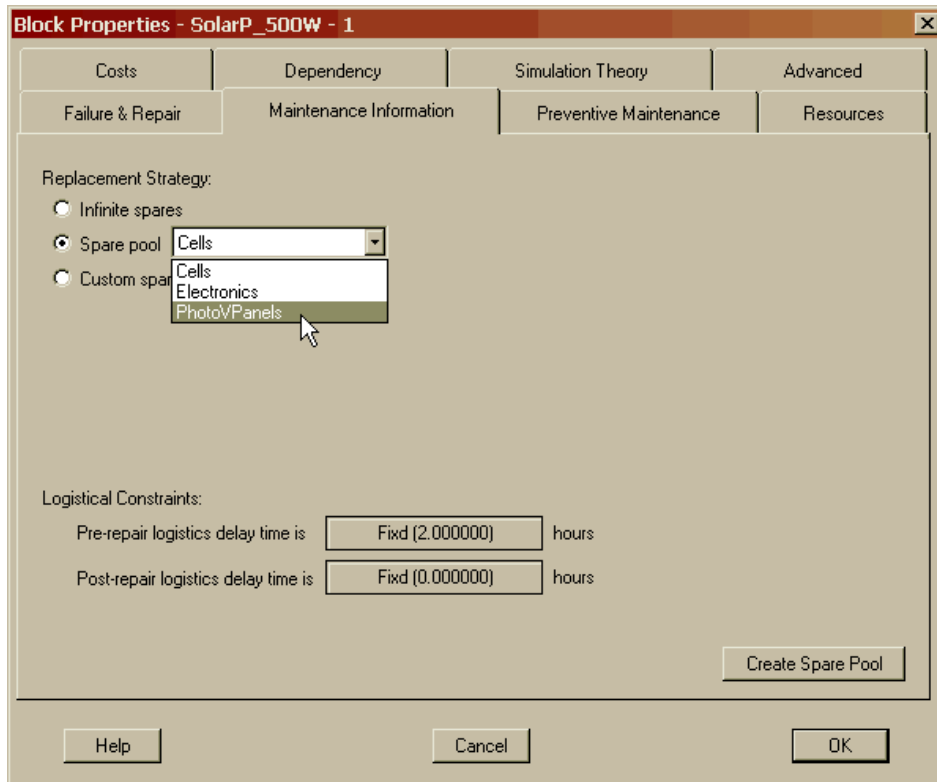


Figure 4.9 Changing from Infinite to Spare Pool Replacement Strategy

Block Name	Source of Spares	Stock	New	Arrive at	Order	# of	Arrive at	Emergency
Attitude	Infinite	N/A	N/A	N/A	N/A	N/A	N/A	N/A
BatteriesNaS	Cells	3	1	730.0	0	1	360.0	168.0
BatteriesNiCd	Cells	3	1	730.0	0	1	360.0	168.0
BatteriesNiH	Cells	3	1	730.0	0	1	360.0	168.0
Bus_FR	Electronics	2	N/A	N/A	N/A	N/A	N/A	24.0
Bus_QR	Electronics	2	N/A	N/A	N/A	N/A	N/A	24.0
Bus_UR	Electronics	2	N/A	N/A	N/A	N/A	N/A	24.0
HighGain	Infinite	N/A	N/A	N/A	N/A	N/A	N/A	N/A
LowGain	Infinite	N/A	N/A	N/A	N/A	N/A	N/A	N/A
SolarP_500W	PhotoVPanels	2	1	4000.0	N/A	N/A	N/A	72.0
SolarP_650W	PhotoVPanels	2	1	4000.0	N/A	N/A	N/A	72.0
SolarP_900W	PhotoVPanels	2	1	4000.0	N/A	N/A	N/A	72.0
Thermal	Custom	1	1	1460.0	N/A	N/A	N/A	48.0

Figure 4.10 Completed Replacement Strategies

Select the fourth tab of the Attitude component’s Block Properties dialog box and notice that this component requires three astronauts to begin its hands-on repair (Figure 4.11). Resources concerning preventive maintenance will be discussed in Chapter 6 when the subject is introduced.

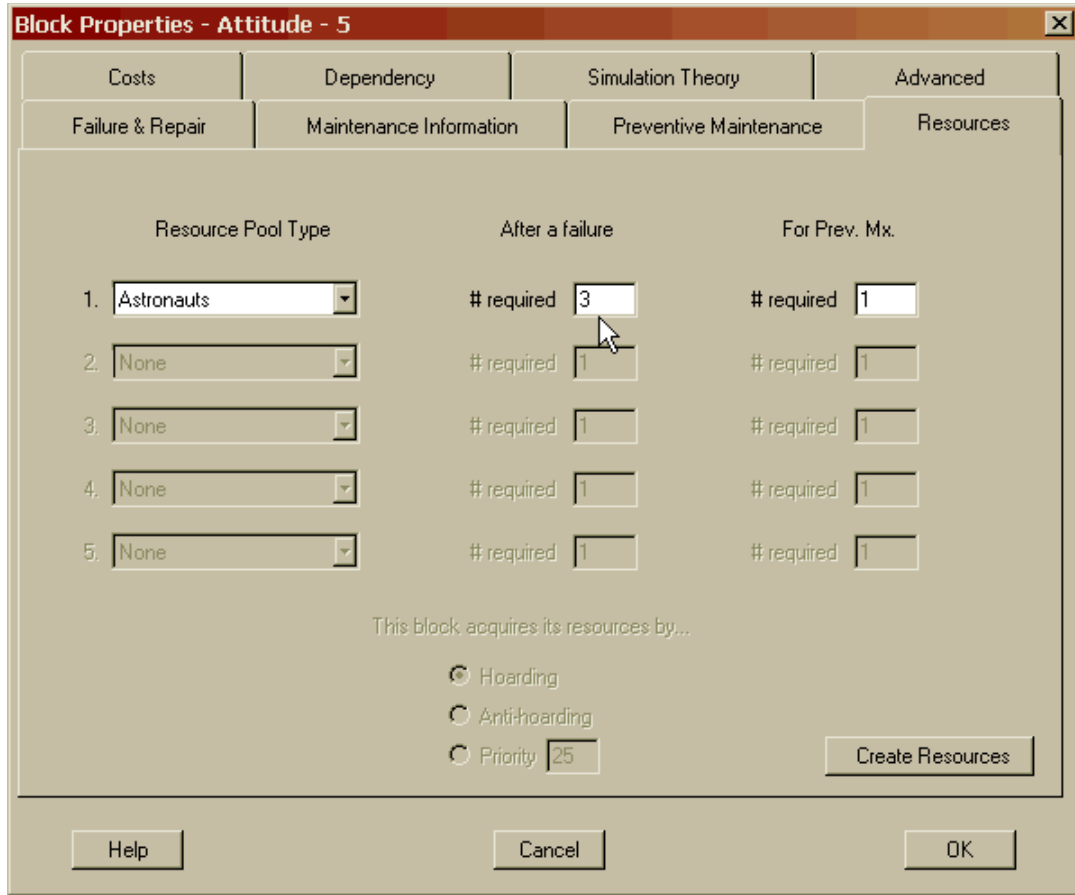


Figure 4.11 Resources for the Attitude Component

These three astronauts are drawn from a resource pool that has already been created for this RBD. The creation of resource pools is conducted in the same manner as spare pools by simply clicking on the Create Resources button of the Resources tab. Resources differ from spares in that they are not discarded but are entities that are held by components until their repairs are completed. Even if a spare is available, repair of a component cannot begin until all resources requested by a component are also available. In fact, resources are not requested until a spare has first been obtained by a component.

Click the Create Resources button and select Astronauts from the dropdown list box. As shown in Figure 4.12, there are only five astronauts available to conduct repair operations for all components assigned to this resource pool. See Figure 4.2 to view a global perspective of the component demand for the five astronauts. Note: While an RBD can possess numerous resource pools, a component can only draw from one of these pools.

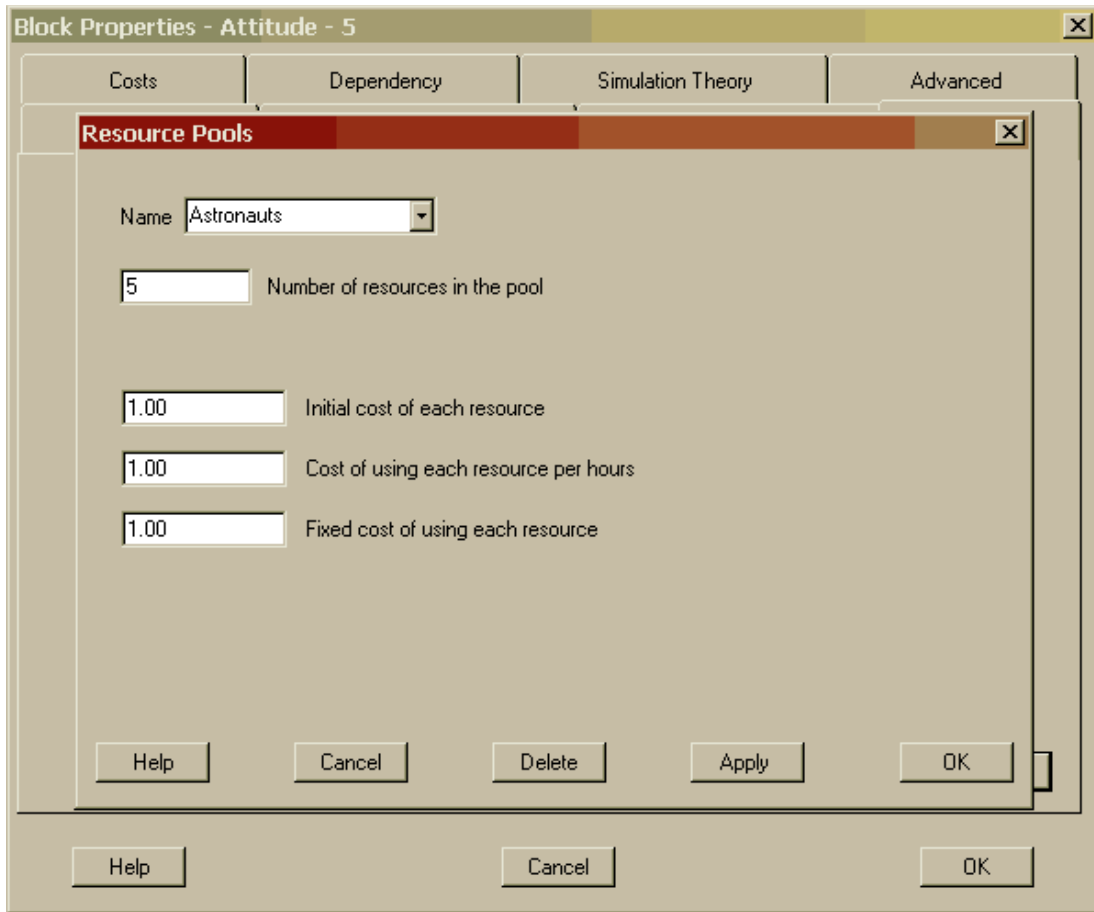


Figure 4.12 Resources for the Attitude Component

You can create spare and resource pools without having to be within the Block Properties dialog box of a particular component; this is merely a convenience. The Spare Pools and Resource Pools dialog boxes can be displayed via the *Options – Spares* menu item and the *Options – Resources* menu item, respectively.

For the RBD we have just finished building, conduct a time-truncated simulation for ten trials. Set the simulation length to 730.5 hours. Your results should look identical to those shown in Figures 4.13 through 4.15. Specifically note that the availability and MDT results of Chapter 3 (0.976313806 and 13.263877, respectively) are significantly different from those highlighted in Figure 4.13. Since the RBDs from Chapters 3 and 4 are so different, a direct comparison of results is not appropriate. For the purposes of this workbook however, we shall say that the results from this chapter are more accurate representations of the true RAM characteristics for this system, since this model is of higher fidelity by simulating the effects of numerous logistical constraints. Recall that the Chapter 3 RBD possessed infinite spares and infinite resources since neither were ever specified (i.e., the Raptor default situation). In the long run, the action of adding logistics constraints can only degrade a system’s overall RAM performance since infinite spares and resources represents ideal conditions.

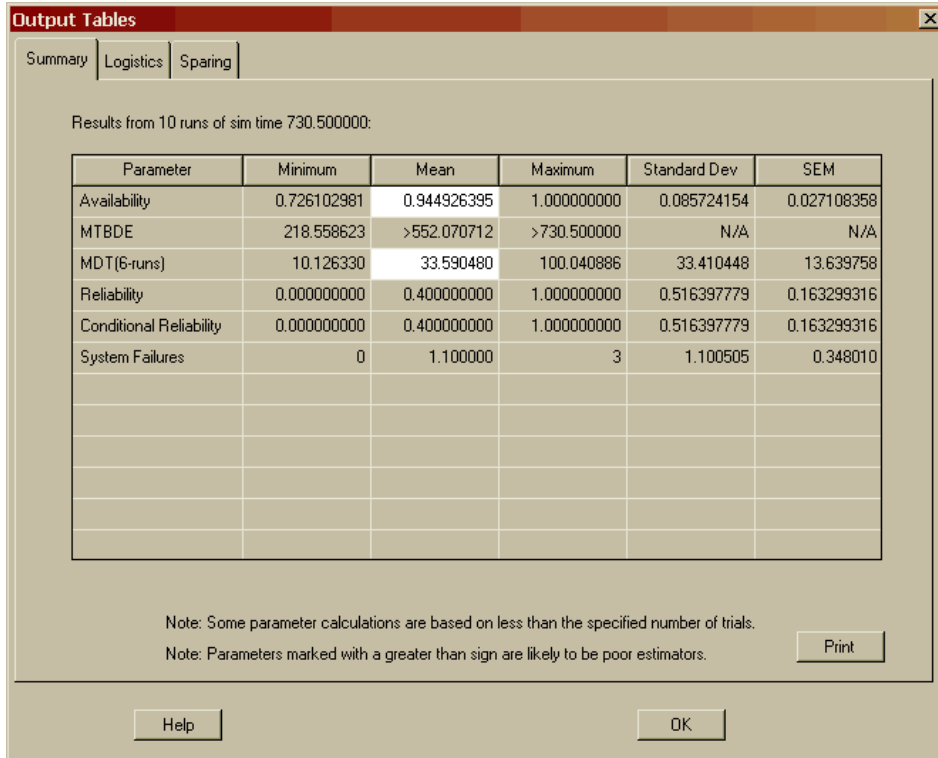


Figure 4.13 Summary Results from Chapter 4

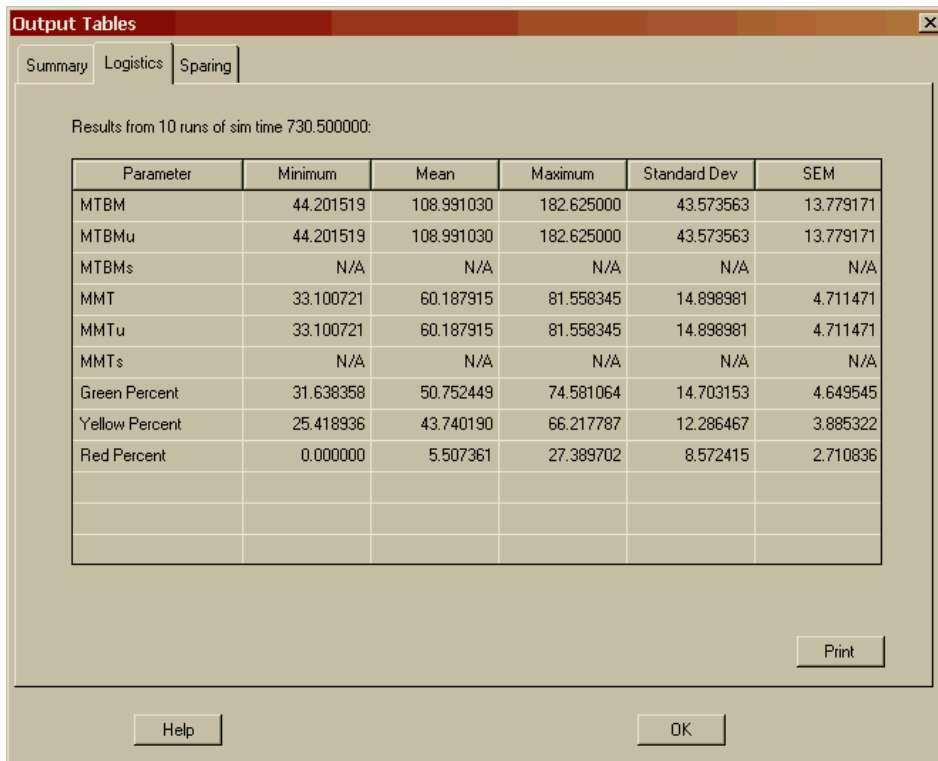


Figure 4.14 Logistics Results from Chapter 4

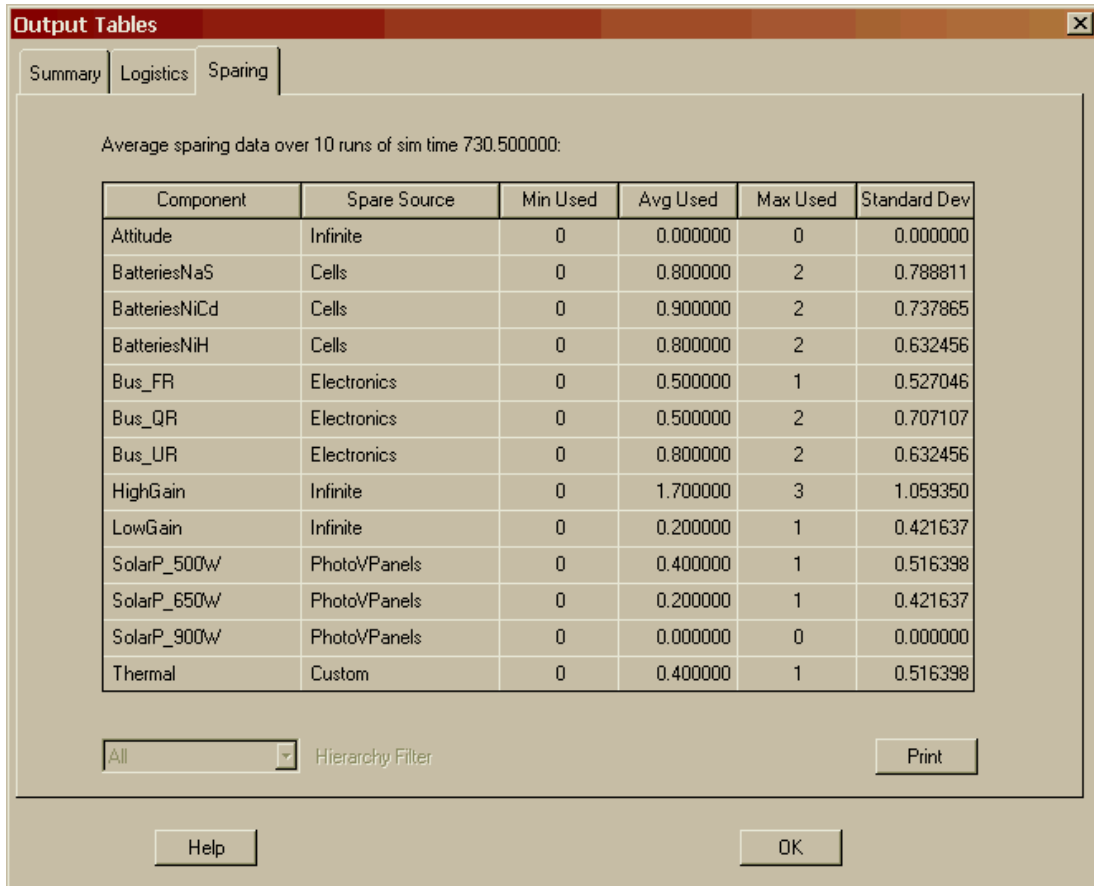


Figure 4.15 Sparing Results from Chapter 4

The total mean time between maintenance (MTBM) is nearly 109 hours, which indicates how often the system requires some kind of maintenance that may or may not be associated with a system-level failure. The unscheduled mean time between maintenance (MTBMu) is identical to the total MTBM since preventive maintenance is not specified in this model. The total mean maintenance time (MMT) is just over 60 hours, which indicates the average hands-on maintenance time for all components that failed during the simulation. If preventive maintenance is not included in a particular model, the MMT parameter is commonly denoted as the Mean Repair Time (MRT). Figure 4.14 also indicates the percentage of time the system is within a red (failed), yellow (weakened) or green (fully functional) state.

Notice that minimum, maximum, standard deviation and standard error of the mean (SEM) are given for each reliability, availability or maintainability parameter. The minimum is the smallest value observed over the number of trials simulated, while the maximum is the largest. The standard deviation (sample not population) is a measure of the spread of the results across all relevant trials. The SEM is a measure of the spread about the mean value and is determined by dividing the standard deviation by the square root of the number of runs (or trials) that were simulated. The SEM is often used when attempting to construct confidence bounds about the mean of a particular parameter. For example, the 90% confidence interval about the mean availability,

MDT, MTBM and MMT is approximately equivalent to the mean plus or minus two SEMs and are shown in Figure 4.16.

	Lower Bound	Mean	Upper Bound
Availability	0.895	0.945	0.995
MDT	6.106	33.590	61.075
MTBM	83.732	108.991	134.250
MMT	51.551	60.188	68.825

Figure 4.16 Confidence Bounds for some Chapter 4 Results

Notice that Figure 4.15, the Sparing tab of the Output tables, displays the number of spares that have been consumed during the simulation. The HighGain component consumed on average 1.7 spares across the ten trials of simulation. At least one trial needed three spares for the HighGain component and at least one trial did not need any spares. Again, it can be shown that a 90% confidence bound interval for the number of spares needed for the HighGain component ranges from 1.086 to 2.314. Thus, providing 2.3 spares would ensure high confidence that the HighGain component will not run out of spares during this length of simulation. Since non-integer spares often cannot be acquired, a provisioning of three spares might be considered appropriate for this component. (Note: This results in a confidence of 99.63%).

The Attitude component had an infinite supply of spares but did not require any during the ten trials of simulation. If the simulation length is representative of the operational time of interest and enough trials are conducted, the Sparing tab of the Output Tables could be used to determine the provisioning levels for each block. Notice that the solar panels collectively consumed only 0.6 spares (the sum of 0.4, 0.2 and 0.0 average spares consumed for the 500, 650 and 900-watt solar panels, respectively) across the ten trials. Recall that the photovoltaic spare pool has an initial stockpile of one, which for this simulation seems to be adequate. While Raptor's logistics capabilities are designed to allow an assessment of a particular sparing or resource strategy, Raptor cannot directly determine an optimal strategy. Raptor can be used, however, as a tool that allows a user to iterate towards an optimal solution.

• • •

To fully grasp the logistics simulated by Raptor, you must understand the repair cycle that components undergo once they have failed. Figure 4.17 summarizes the repair cycle that a component undergoes each time it fails unless the None distribution is specified. The reader should note that some portions of a repair cycle always occur instantaneously since they are logic features of the Raptor code (e.g., check system status, tally statistics, etc.) and thus are not traditional aspects of a logistic repair cycle. The most important aspect of this diagram is to realize that there are a number of events (resources, spare availability, etc.) that can delay the start or conclusion of a component's repair, which then directly affects a system's RAM characteristics. If a

component uses the None repair distribution, it will not initiate a repair cycle. Instead, a component is permanently failed without any possibility of being repaired within a particular trial of a simulation.

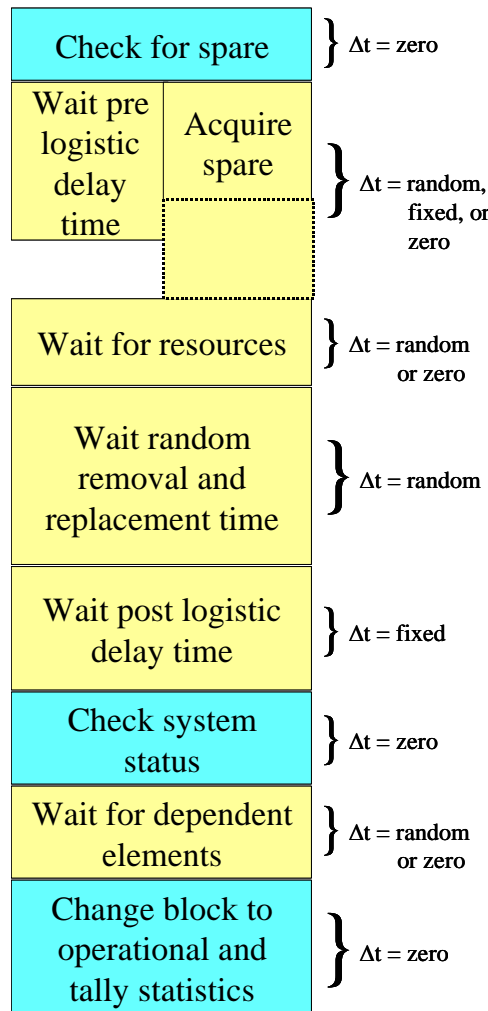


Figure 4.17 Component Repair Cycle

Problems

1. When is it appropriate to use the unrealistic replacement scenario of infinite sparing?
2. What happens to a component that tries to obtain a spare from an empty pool?
3. If two components with different repair distributions are in the same spare pool, and one of them fails, what repair rate is used to fix the broken component?
4. Does a custom sparing of 10 spares arriving every 1000 hours equal 1 spare arriving every 100 hours?

5. Is there any priority for assigning resources to competing components?
6. Can a component be assigned to a resource pool *and* a spare pool?
7. If component A needs three repair people to start its repair cycle but only one is available, then component B fails and needs only one repair person from that same pool, can the repair of component B begin?
8. Name a system that might use a fixed length logistics time delay and one that might use a random delay length.
9. Determine the 90% confidence interval for the mean number of system failures as shown in Figure 4.13.
10. Determine the 99% confidence interval for the mean system green time as shown in Figure 4.14.
11. What is the probability that more than one spare would be needed for the LowGain component for a month of operation?
12. Is the length of any repair cycle fixed, random, or either? Explain your reasoning.

Chapter 5



Objectives

1. Be able to build an RBD.
2. Be able to extract relevant attributes of an RBD from a physical description of a system.
3. Be able to set up a spare pool.
4. Understand the best way to determine a system's MTBDE.
5. Understand the importance of consistent units in a simulation.
6. Understand the difference between no repair and zero length repairs.

Recommended Problems

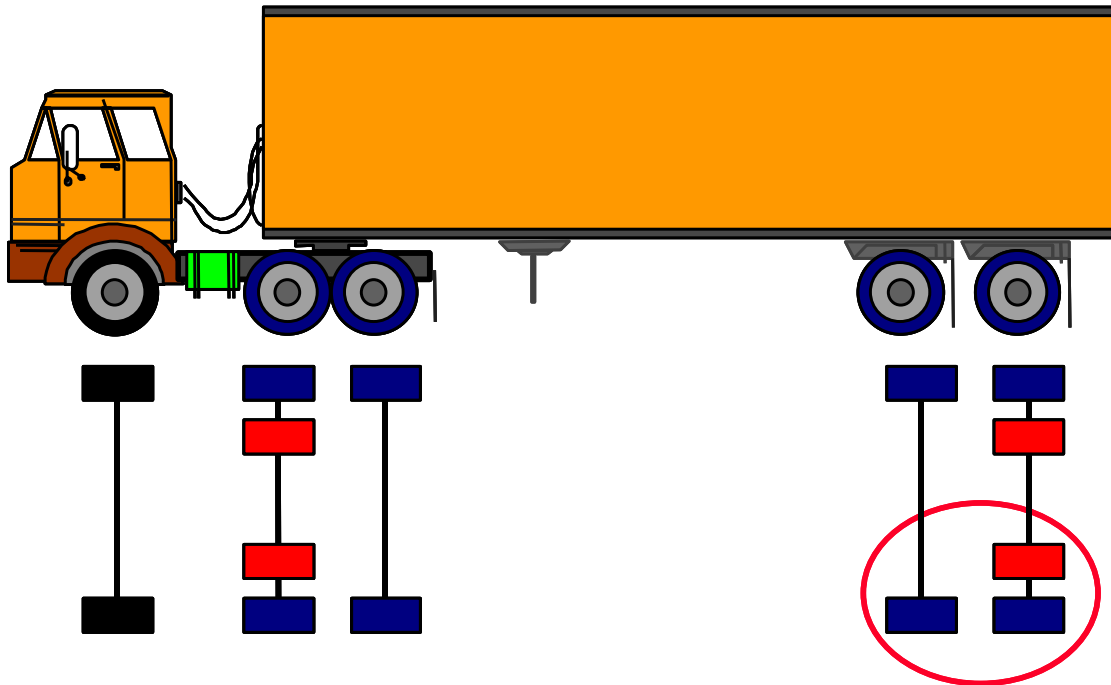
2, 3, 5 and 6

Hands On Example

The best way to understand how to apply and appreciate any software application is to actually use the software. A problem is presented below that should reinforce your understanding of the previous lessons. It begins with the actual development of a system-level RBD (sometimes the most difficult task in constructing reliability simulations), and ends with the generation of statistically meaningful results.

System Description:

A new 14-wheel truck is being designed by WorldFleet. The tractor consists of the following subsystems: engine, electronics, fuel, cooling, transfer-drive, brakes, tires, structure and a gearbox. The fuel subsystem is composed of two tanks and one pump. There are two braking systems, either of which is capable of stopping the truck. The trailer is comprised of the following subsystems: tires, electronics and structure. The configuration of the tires is shown in Figure 5.1.



10,000 pounds max per corner

Figure 5.1 Tire Configuration of a WorldFleet Truck

The tires beneath the trailer primarily support the truck's load. Each tire can support a 5,000-pound load. The truck will continue to function if each of the four corners (not including the front axle) can sustain a 10,000 pound load. The truck carries two different spares; one spare is for the four interior tires and the other spare is for the remaining tires not including the front axle.

A. Describe a downing event for the truck system.

B. Draw an RBD of the truck system.

C. Determine replacement strategies for each element.

D. Open **Truck1.rbd** and use the inside tire information in Figure 5.2 (all values are in miles) and the replacement strategies you just determined to complete the RBD. The inside tires, which have not yet been added to the RBD take 60 minutes to repair while the outside tires take 30 minutes to repair. Conduct a 25-trial simulation to determine the mission reliability of the truck on a 1200-mile trip and the system's mean time to failure.

Block Name	Order	Failure	Par1	Par2	Par3
Engine	1	Normal	12,000	1,000	
Tractor_Ele	2	Exponential	60,000	0	
Fuel_Tank1	3	Gamma	1.4	57,000	0
Fuel_Tank2	4	Gamma	1.4	57,000	0
Fuel_Pump	5	Weibull	1.25	57,000	0
Cooling	6	Exponential	15,000	0	
Transfer_Drive	7	Lognormal	45,000	1,200	
Brakes_1	8	Weibull	1.3	15,000	0
Brakes_2	9	Weibull	1.3	15,000	0
Gearbox	10	Weibull	2.3	36,000	0
Tractor_Structure	11	Weibull	1.5	900,000	0
LeftFront_Tire	12	Weibull	1.4	60,000	0
RightFront_Tire	13	Normal	1.4	60,000	0
Trailer_Structure	14	Weibull	1.5	720,000	0
Trailer_Ele	15	Exponential	84,000	0	
Tire_Outside_1	16	Weibull	1.2	28,000	0
Tire_Outside_2	17	Weibull	1.2	28,000	0
Tire_Outside_3	18	Weibull	1.2	28,000	0
Tire_Outside_4	19	Weibull	1.2	28,000	0
Tire_Outside_5	20	Weibull	1.2	28,000	0
Tire_Outside_6	21	Weibull	1.2	28,000	0
Tire_Outside_7	22	Weibull	1.2	28,000	0
Tire_Outside_8	23	Weibull	1.2	28,000	0
Tire_Inside_1	24	Weibull	1.2	36,000	0
Tire_Inside_2	25	Weibull	1.2	36,000	0
Tire_Inside_3	26	Weibull	1.2	36,000	0
Tire_Inside_4	27	Weibull	1.2	36,000	0

Figure 5.2 Truck Failure Distributions

R(1200) = _____

MTBDE = _____

A longer simulation time could have been chosen such that the system was guaranteed to fail in each trial. This approach would ensure that the greater than sign would not be displayed in front of the MTBDE value. An obstacle with this approach is that the user would have to guess the ending simulation time such that all trials would see at least one system failure. This type of simulation could be slow since those trials that would have early system failures would continue until the ending simulation time is reached. A method for overcoming this obstacle is to conduct what are known as failure-truncated simulations. Thus, the best way to achieve a good indication of the system's mean time to first downing event would be to conduct a one-failure simulation for a number of trials. This type of simulation is often called an endurability simulation since you are attempting to determine how long a system can endure before failing. Figure 5.4 displays how to set up a failure-truncated simulation.

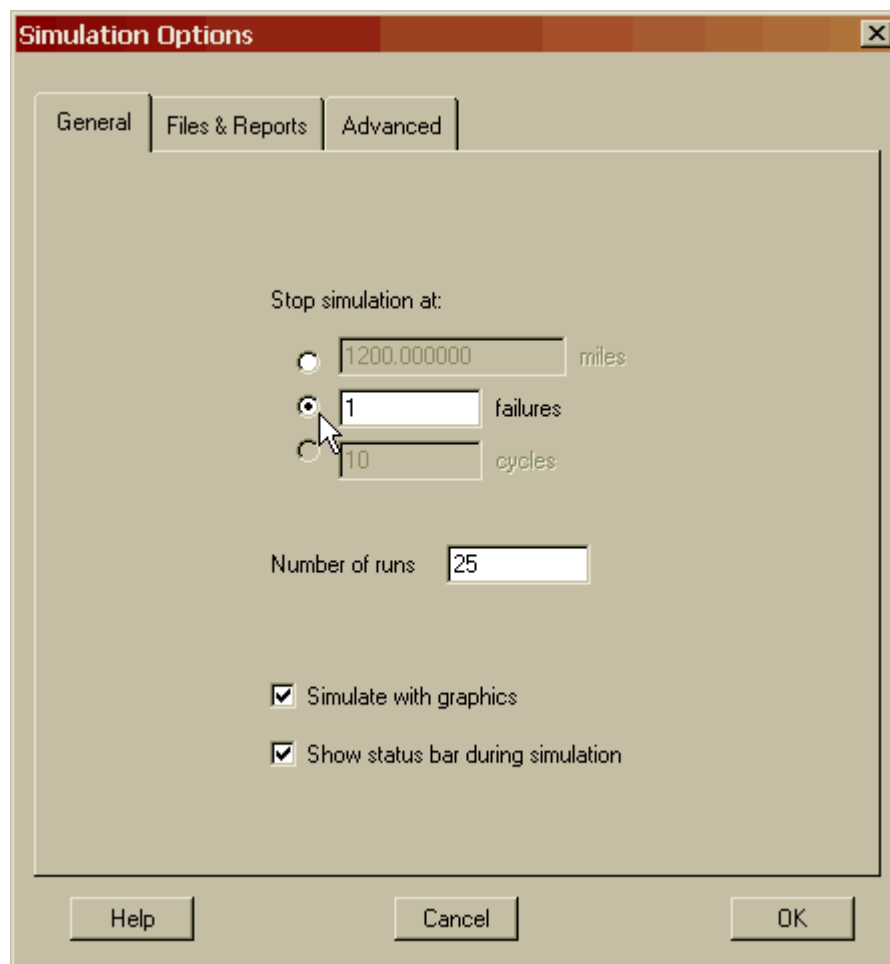


Figure 5.4 Setting up a Failure-Truncated Simulation

Use your existing RBD or open **Truck3.rbd** and conduct a failure-truncated simulation. The results of such a simulation are shown in Figure 5.5.

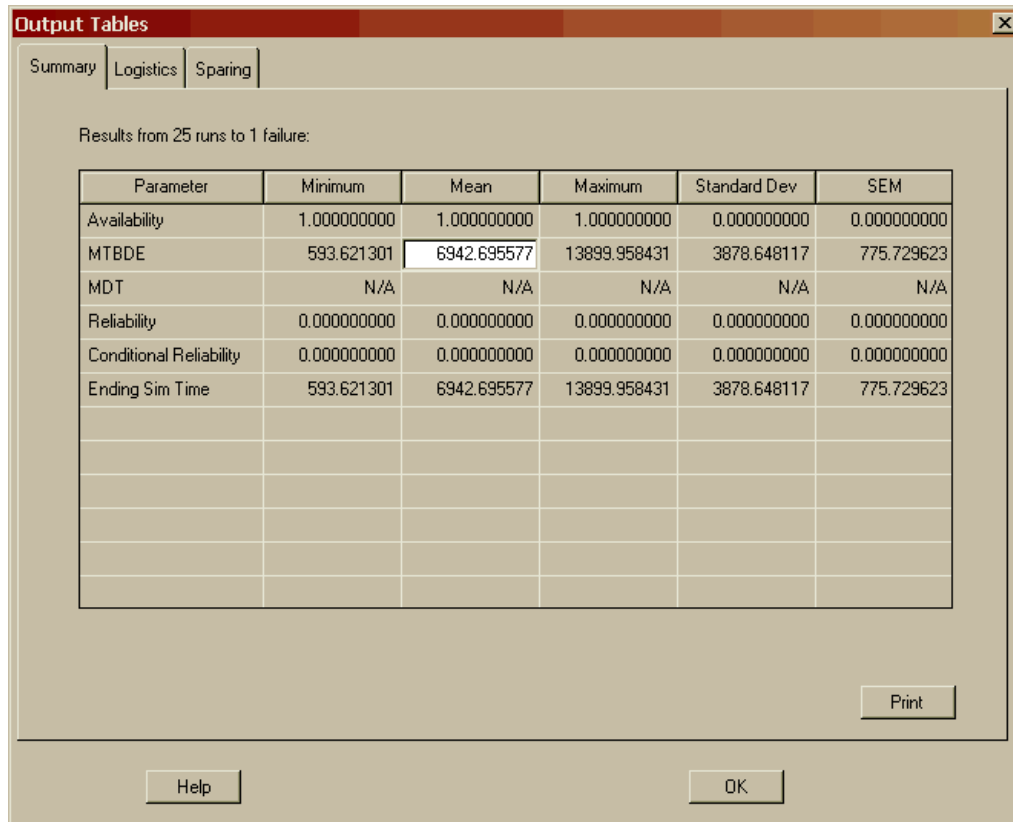


Figure 5.5 Summary Results for Failure-Truncated Simulation

Of the 25 trials conducted, the average trip duration achieved was a little more than 6,942 miles; this is enough for a cross-country round trip of the United States. An MTBDE of 6,942 miles is a much better estimate of the truck's mean life than the previous simulation result of a little more than 1,101 miles. Perhaps of more interest is the minimum value obtained, which in this case was a little more than 593 miles. Thus we could be relatively confident that our truck could consistently accomplish a 600-mile trip based on a sample size of 25 trials.

Notice in Figure 5.5 that the mission reliability parameter is zero when conducting failure-truncated simulations. The reason for this is that we specified the system to fail once, thus we know the mission reliability by definition will equal zero for every trial.

• • •

Readers should not be surprised that they must modify a simulation to extract relevant data from the summary results tab of the Output Tables dialog box. That is, one simulation does not necessarily yield all the information desired about the system's RAM characteristics. The previous simulations were structured to yield relevant mission reliability values in one case, and system MTBDE in another.

The reader should note that during these simulations, all non-tire components turned an Indian red color (i.e., done status) upon failing and this color indicates that there is no possibility of repairing this component (e.g., none repair distribution selected, spare stockpile is exhausted or replacement spares are not scheduled to arrive). Also, notice in Figure 5.5 that the last row's title has changed from "System Failures" to "Ending Sim Time" since we already know the number of system failures that will occur. We do not know, however, when each simulation trial will terminate since this is a random variable for failure-truncated simulations.

Problems

1. Why are the MDT results in Figure 5.3 based on only five of the 25 trials conducted?
2. What do the MDT values from Figure 5.3 represent with respect to miles?
3. Why are the availability numbers in Figure 5.3 bad estimators of the truck's true availability?
4. Explain when you would conduct a mission reliability simulation and when you would conduct an endurance simulation.
5. How would you properly determine this system's MDT?
6. List some ambiguous aspects of the truck examples.
7. Why are the MMT results from the **Truck3.rbd** based on only eight of the 25 trials conducted?

Chapter 6



Objectives

1. Understand the preventive maintenance options.
2. Be able to distinguish between individual and collective preventive maintenance.
3. Be able to create a preventive maintenance trigger.
4. Be able to assign preventive maintenance resources.
5. Understand the preventive maintenance specific output parameters.
6. Understand the issue with scheduling preventive maintenance activities for exponentially distributed components.

Recommended Problems

1, 3, 4, 8 and 10

Preventive Maintenance

Thus far, Raptor has only sent a component to its repair cycle if a failure occurs based on the component's probability density failure function. This is often called a *hard failure* and occurs randomly, which implies that the failures are unscheduled. This chapter's focus is on Raptor's abilities to model scheduled maintenance activities for a single component and for a group of components. Often scheduled maintenance is referred to as preventive maintenance and is traditionally implemented to lengthen the mean time between failures of a component. The goal of preventive maintenance is to time a scheduled maintenance action before the component was due to fail and thus avoid an unpredictable failure. By timing the frequency of preventive maintenance correctly, users can manage the impacts of component failures on a system.

Open the **PreventMx1.rbd** file and then display the Block Properties dialog box for the Attitude component. As shown in Figure 6.1, select the third tab labeled Preventive Maintenance and then engage the checkbox at the top of the tab. Once this checkbox is marked, the preventive maintenance activities of this component can be defined.

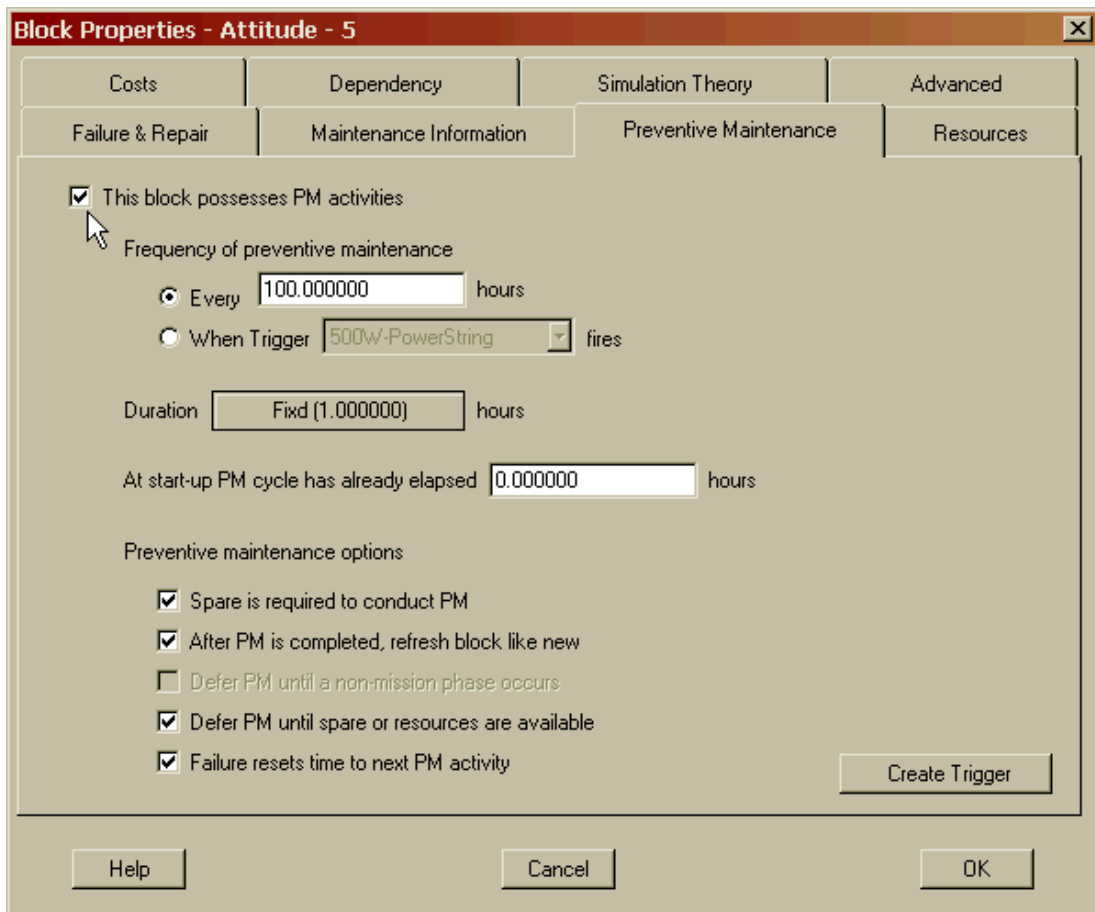


Figure 6.1 Preventive Maintenance Tab of Block Properties Dialog Box

The first task that must be specified before any preventive maintenance can transpire for a particular component is to determine how often this scheduled maintenance will occur. Raptor allows the frequency of preventive maintenance to be defined in one of two forms. First, a component can have a fixed frequency that is entirely associated with only that component. Second, a component can belong to a group of components that all have preventive maintenance occurring at the same time. The latter is a common maintenance strategy for many systems since it coordinates maintenance activities on groups of components that possess the same preventive maintenance frequency. For example, after a fighter aircraft has flown for 500 hours, it is sent to its preventive maintenance inspection where numerous components are scrutinized, replaced or possibly lubricated. Another example is a car that has a preventive maintenance cycle of 3,000 miles in which the oil, the oil filter and the air filter are all replaced. It is possible to define all three of the previously mentioned preventive maintenance actions individually by component but since they all occur at the same frequency, it is usually better to define a preventive maintenance group. A group is defined by specifying a *trigger* and then attaching the desired components to that trigger. We will discuss trigger-based preventive maintenance and its advantages later in this chapter. Set the Attitude's frequency of preventive maintenance to be a fixed 1,500 hours as shown in Figure 6.2.

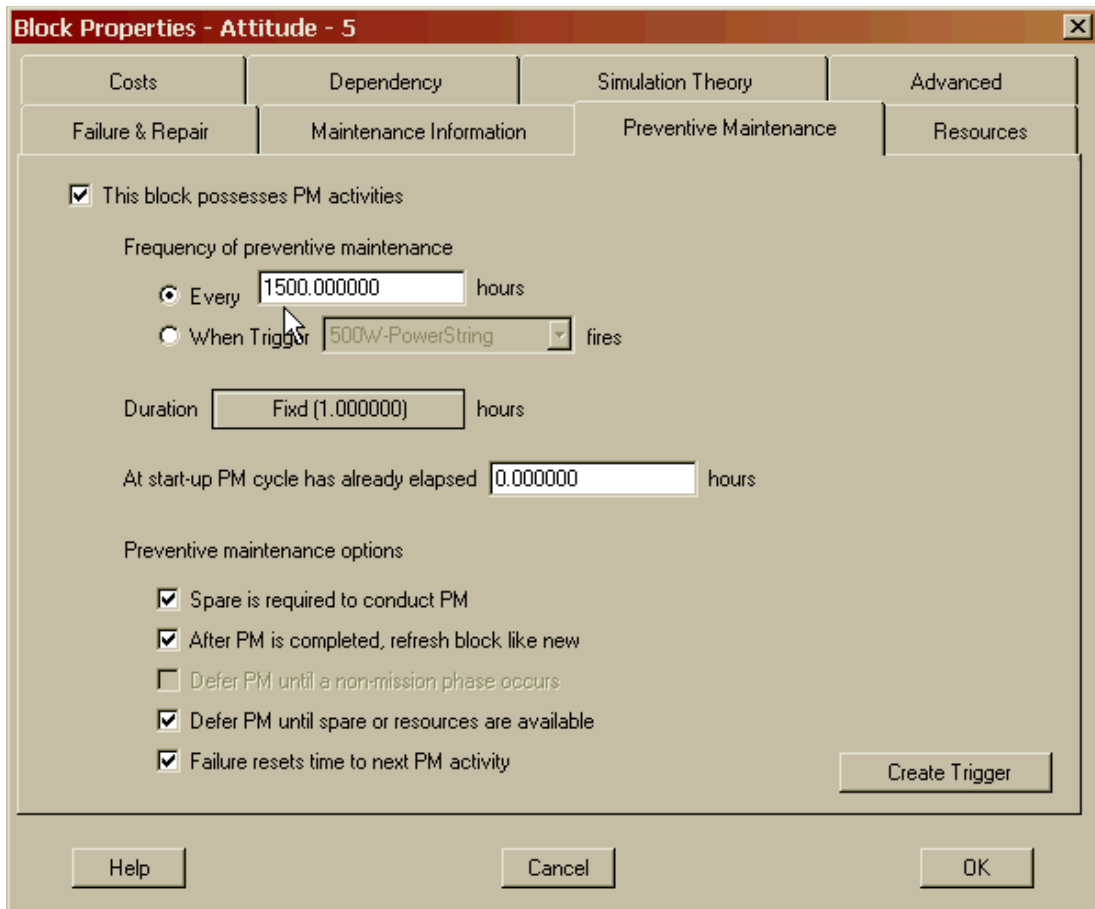


Figure 6.2 Setting the Frequency of Preventive Maintenance

Now that the frequency has been specified, we need to define the duration of the preventive maintenance repair action. Select the Duration button and set the length of the preventive maintenance to be triangularly distributed as shown in Figure 6.3. Return to the component's preventive maintenance tab (Figure 6.4) and note that the Duration button now displays a mean of 2.167.

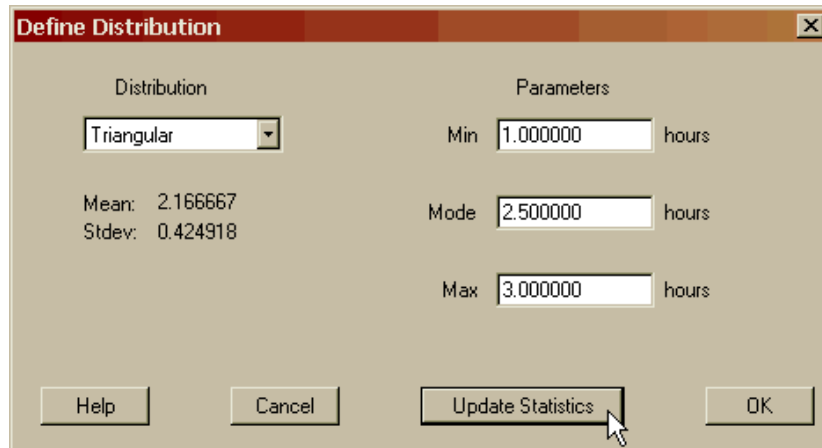


Figure 6.3 Setting the Duration of Preventive Maintenance

At this point, the basic preventive maintenance functions for this component have been defined. There remains an advanced feature that is commonly specified when multiple components have the same preventive maintenance frequency. An elapsed time can be delineated that can shorten the first occurrence of a component's preventive maintenance. This is often used when a user does not want an entire group to arrive for preventive maintenance at the same time and thus overwhelm the maintenance personnel, unnecessarily weakening the system. For example, four machines may need to be lubricated every 100 hours. The first machine could have an elapsed time of 80 hours and thus its first preventive maintenance action would occur at a simulation time of 20 hours. The time to the first preventive maintenance action is determined by subtracting the elapsed time from the preventive maintenance frequency. The elapsed times for the other three machines could be specified as 60, 40 and 20 hours and thus their first preventive maintenance actions will occur at 40, 60 and 80 hours, respectively. This type of offset strategy attempts to ensure that the four-machine system does not inundate the maintenance personnel and that the system never has more than one of four machines engaged in preventive maintenance activities.

There are five checkbox options that can be specified for a component if further fidelity is needed to stipulate a component's preventive maintenance activities. The first checkbox specifies whether or not a spare is needed to engage the preventive maintenance action. For example, an inspection type of preventive maintenance often does not require a spare. The second checkbox states that after a preventive maintenance action is completed, a component is either refreshed as good as new or operates with the amount of life exhausted that it possessed when it started the preventive maintenance action. The third checkbox, which is currently deactivated,

states whether a preventive maintenance action is delayed until a system is in a non-mission mode. Missions will be defined in Chapter 17. The essence of this option is that you might not want to perform preventive maintenance on an aircraft engine while the aircraft is in flight. The fourth checkbox allows a user to specify whether or not a component will defer its preventive maintenance until the required spare and resources are available to begin the repair action. When this checkbox is unmarked, a component may have a longer effective preventive maintenance time than specified since it could be waiting to start its scheduled maintenance action due to logistical competition for spares and resources. The last option is a checkbox that states whether a hard failure (i.e., an unscheduled maintenance event) resets the time to the next preventive maintenance activity or allows it to occur as previously scheduled. For example, if a component with a preventive maintenance frequency of 100 hours fails after 99 hours of operation, should it conduct its preventive maintenance action one-hour later or 100 hours after its repair is completed?

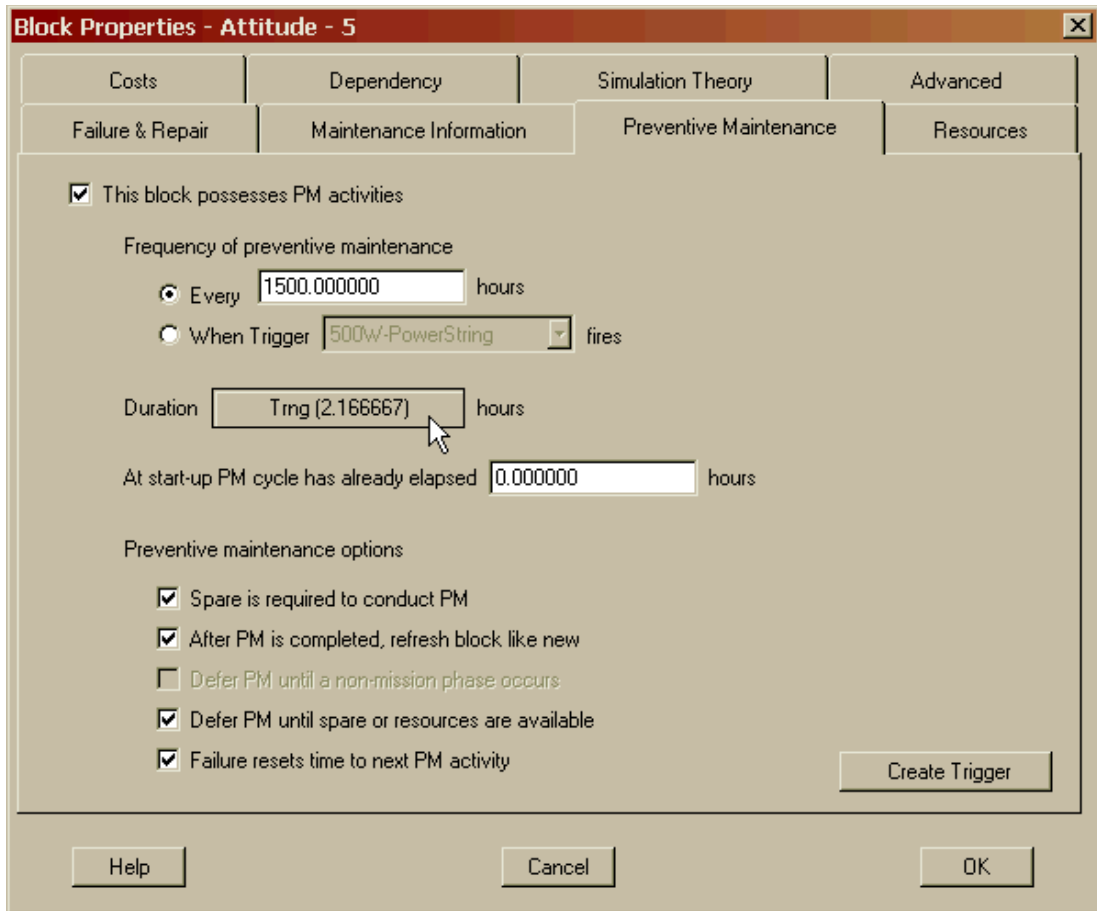


Figure 6.4 Updating the Preventive Maintenance Duration for the Attitude Component

As shown in Figure 6.4, the Attitude component needs a spare to perform its preventive maintenance action, will be refreshed to a new component status after this action, will wait for the appropriate number of Astronauts and a spare are available to start this action, and a hard failure will reset its time for its next preventive maintenance action.

Now that all of the preventive maintenance functions are delineated for the Attitude component, the number of resources needed to conduct the preventive maintenance action needs to be specified. For this component, we will need two Astronaut resources to perform the preventive maintenance action. Recall from Chapter 4 that a component's resource information is located on the fourth tab of the Block Properties dialog box. After being modified as described above, Figure 6.5 shows that three Astronaut resources are needed to repair the component as a result of an unscheduled maintenance event while only two are needed for a scheduled maintenance event.

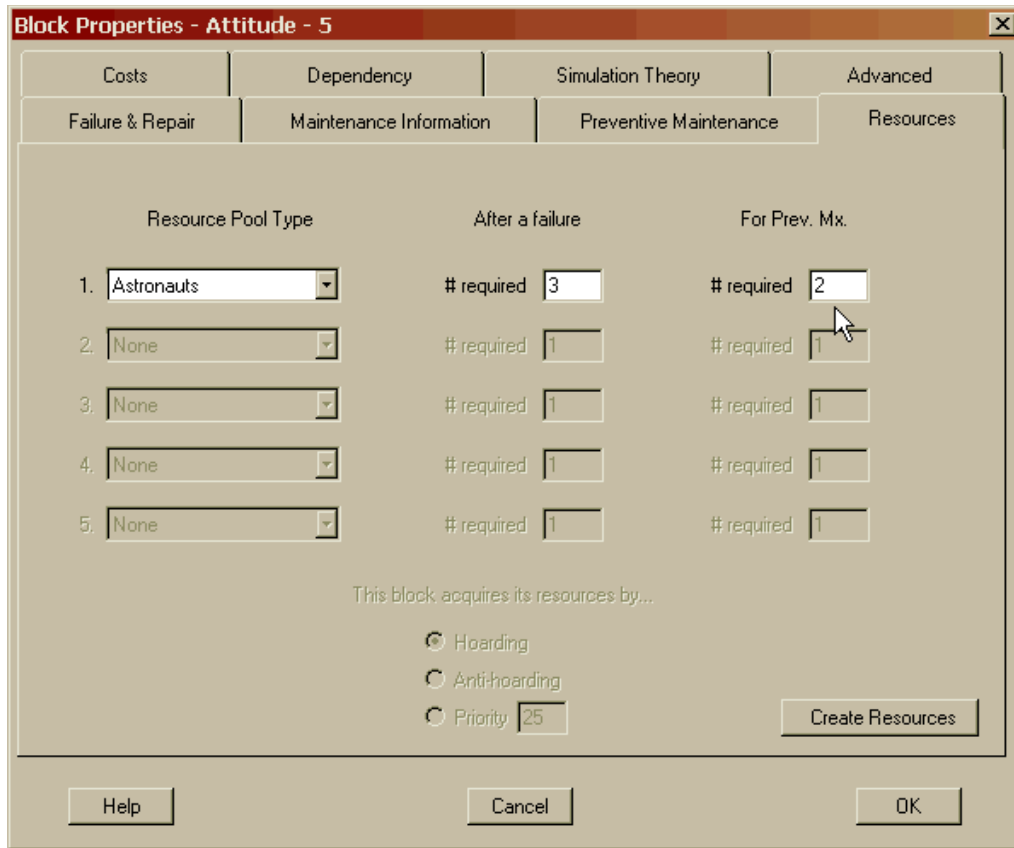


Figure 6.5 Setting the Number of Preventive Maintenance Resources

Thus far we have only defined preventive maintenance for a single component. Next, we want to define a few preventive maintenance groups. By defining groups we ensure that the preventive maintenance actions are performed at regular intervals, which might not be the case for a single component. The repair of a component, its dependency relationship or its standby status can all shift a single component's preventive maintenance schedule.

We want to perform preventive maintenance at regular intervals on each of the solar panels and their corresponding batteries but in such a manner that we do not needlessly take the entire system down. We will define a trigger in conjunction with two others previously created such that they occur approximately every twelve days and each trigger is offset by twenty-four hours. Select the *Triggers* menu item from the *Options* menu and note that the Define Triggers dialog appears as shown in Figure 6.6. This dialog box allows numerous triggers to be defined in the same manner as the Spare Pools and Resource Pools dialog boxes.

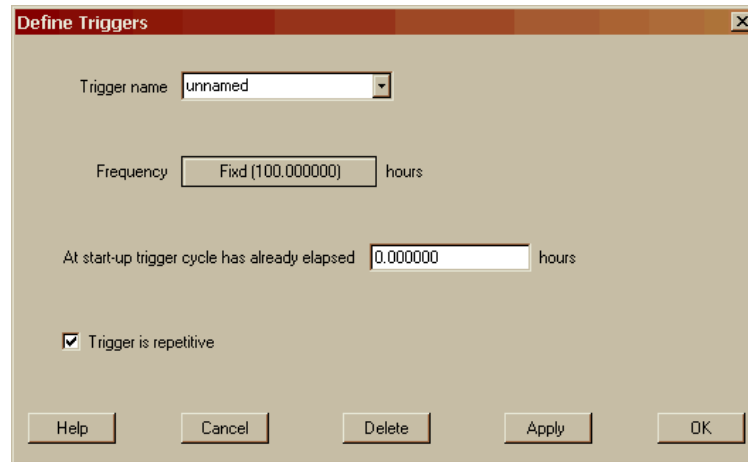


Figure 6.6 Triggers Dialog Box

Triggers must have a unique name, can possess a fixed or random frequency and can be offset by an elapsed time. Triggers can also be defined to occur just once or repeatedly during a simulation trial. By contrast, a non-trigger based preventive maintenance frequency is by definition repetitive. Build a 900W-PowerString trigger using a frequency that is normally distributed (with a mean of 288 hours and a standard deviation of one hour) and an elapsed time of 216 hours as shown in Figure 6.7.

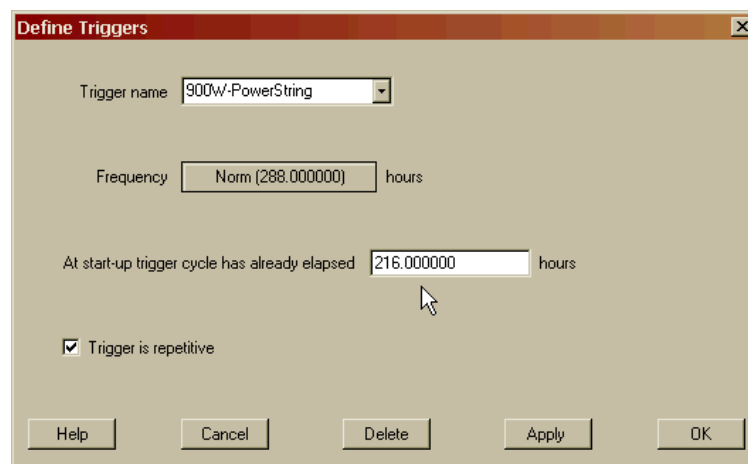


Figure 6.7 900W-PowerString Trigger Defined

Be sure to establish the creation of a trigger by clicking the Apply button. The 500W-PowerString and 650W-PowerString triggers have already been created and their attributes are shown in Figures 6.8 and 6.9, respectively. Both of these triggers are normally distributed with a mean of 288 hours and a standard deviation of one hour.

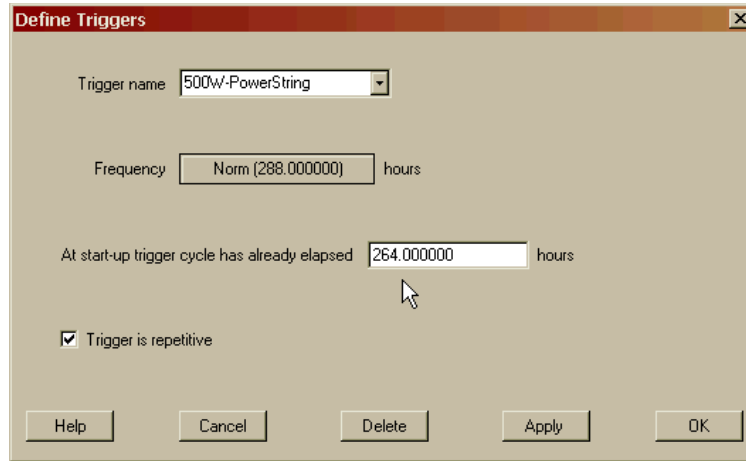


Figure 6.8 500W-PowerString Trigger Defined

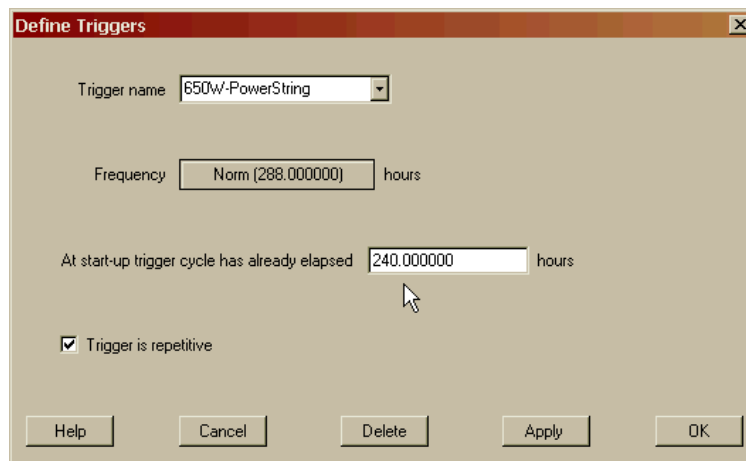


Figure 6.9 650W-PowerString Trigger Defined

Notice that each trigger differs from at least one other by twenty-four hours (i.e., subtract their elapsed times from each other). These triggers are set up such that the 500W-PowerString will occur first, after about twenty-fours of simulation, followed by the 650W-PowerString approximately twenty-fours later. Then the 900W-PowerString trigger will occur about twenty-four hours after the 650W-PowerString trigger. This cycle will repeat approximately every twelve days. A clearer explanation of the manner in which the triggers will operate during a simulation trial of length 730.5 hours is shown in Figure 6.10.

	Day																													
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
500W-PowerString		x												x												x				
650W-PowerString			x												x												x			
900W-PowerString				x												x												x		

Figure 6.10 Preventive Maintenance Schedule Design

Now that we have defined a trigger, we need to group components that will be controlled by that trigger. Select the 900 Watt solar panel component, assign it to the 900W-PowerString trigger, and set its remaining preventive maintenance attributes as shown in Figure 6.11.

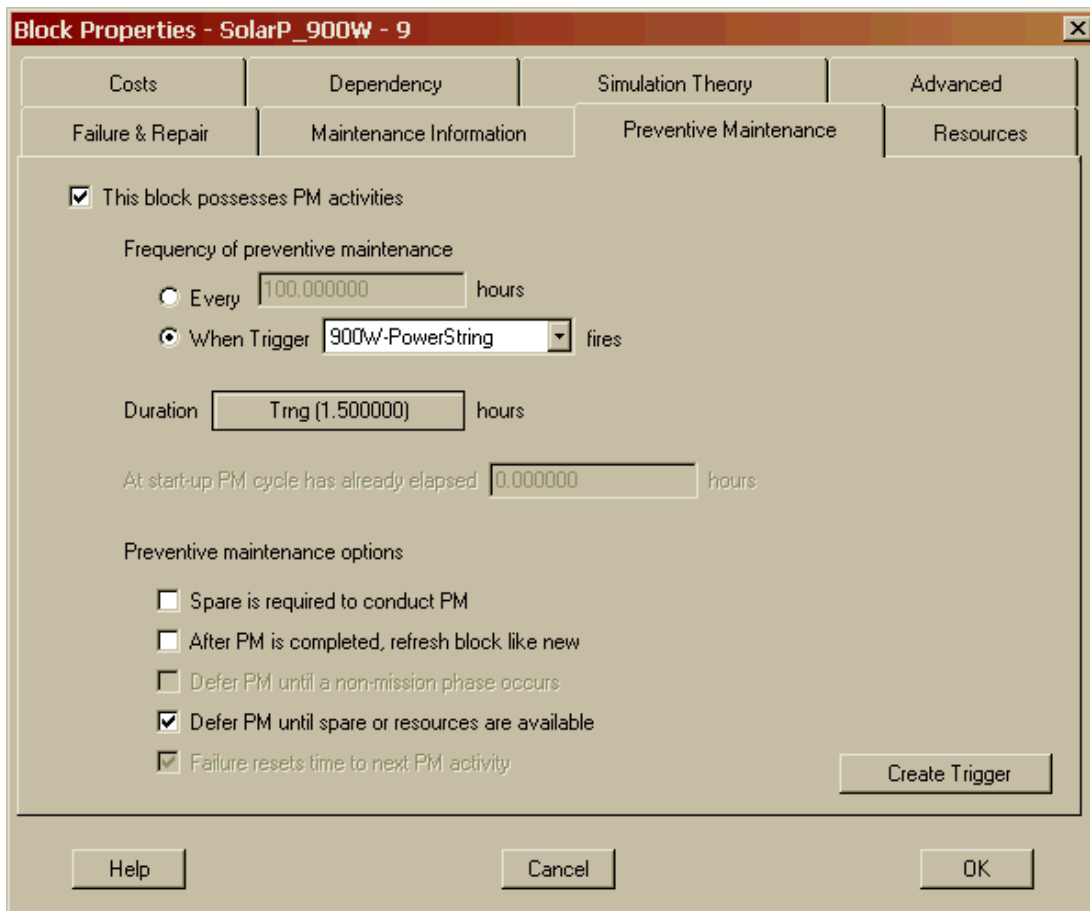


Figure 6.11 900 Watt Component Preventive Maintenance Attributes

Ensure to set the duration to be triangularly distributed with a minimum of one, a maximum of two and the most likely value of 1.5. Uncheck the first two preventive maintenance options checkboxes and note that the last checkbox is deactivated. If a component belongs to a trigger it cannot alter the timing of a trigger's next event since a trigger is a community event associated with multiple components and not an individual component event. Select the Sodium-Sulfate battery component and set its preventive maintenance attributes as shown in Figure 6.12.

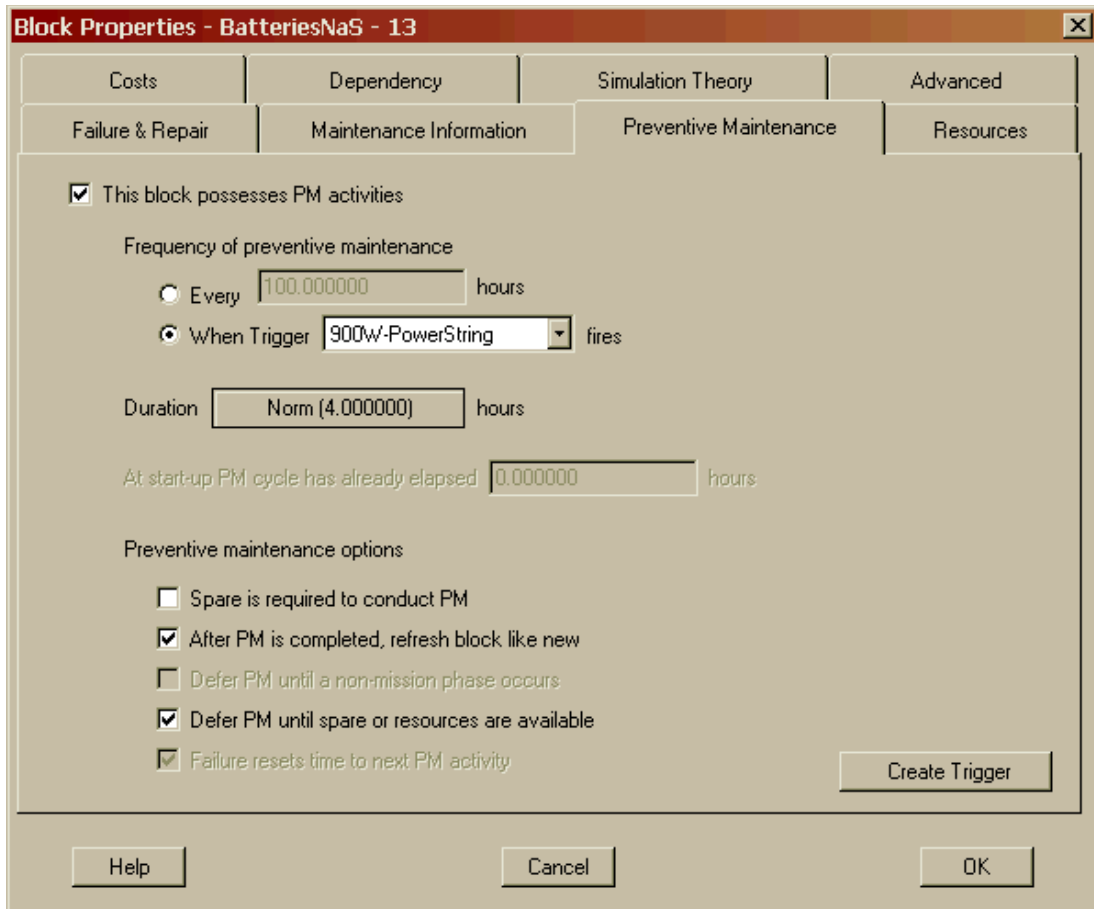


Figure 6.12 Sodium-Sulfate Battery Component Preventive Maintenance Attributes

Set the duration to be normally distributed with a mean of 4 hours and standard deviation of 0.10 hours. Uncheck only the first preventive maintenance option. Thus, the batteries of a power string are refreshed as a new component while their corresponding solar panels are merely inspected and not refreshed. This represents a scenario where by the batteries can be refreshed with new catalysts but the space-based solar panels cannot due to the constant degradation of solar radiation.

The 500 Watt solar panel component has already been assigned to the 500W-PowerString trigger and the 650 Watt solar panel component to the 650W-PowerString trigger. Both utilized the same triangular duration and checkboxes as they were defined for the 900 Watt solar panel component. The nickel-cadmium and nickel-hydrogen battery components have also been previously assigned to their respective preventive maintenance triggers with their attributes shown in Figures 6.13 and 6.14. Both have normally distributed preventive maintenance durations with the former possessing a mean duration of 8 hours with a standard deviation of 0.50 hours and the latter has a mean of 6 hours with a standard deviation of 0.25 hours.

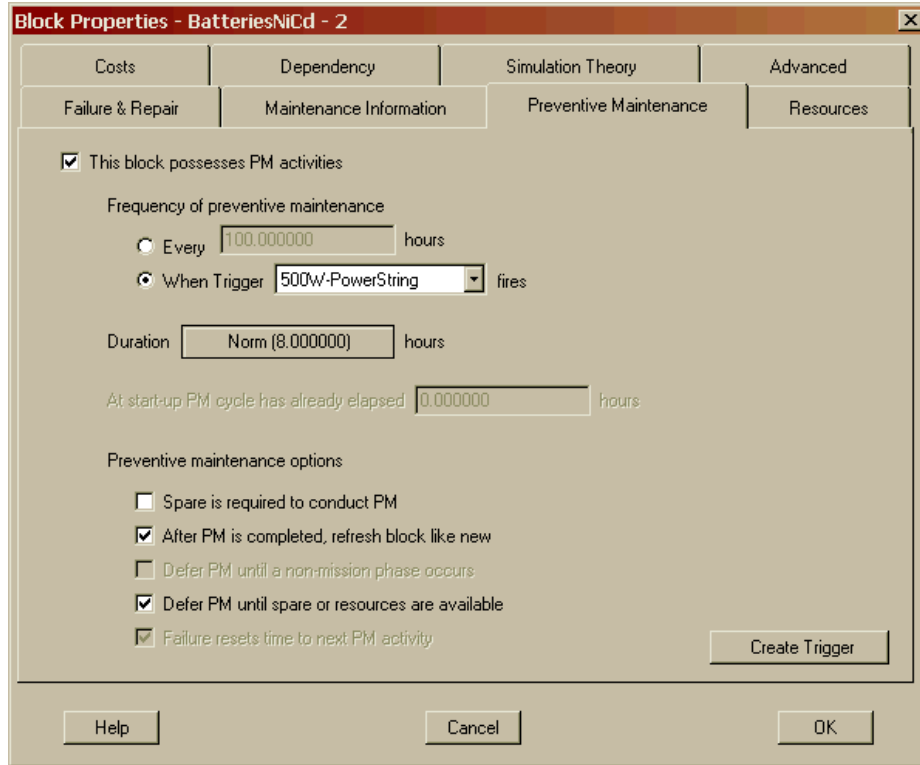


Figure 6.13 Nickel-Cadmium Battery Component Preventive Maintenance Attributes

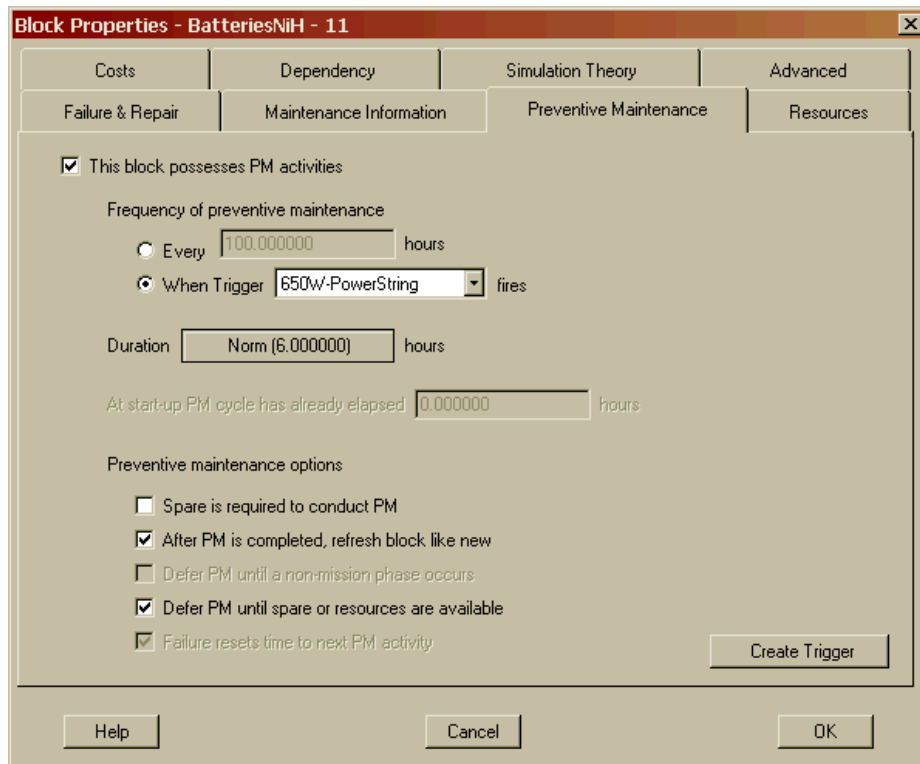


Figure 6.14 Nickel-Hydrogen Battery Component Preventive Maintenance Attributes

Open the Block Input Tables dialog box and select the Preventive Maintenance tab. If all changes were conducted properly, the tab should appear as shown in Figure 6.15. Notice that only the seven components for which we checked the preventive maintenance checkbox are listed on this tab. All other components possess only unscheduled maintenance activities. The last five columns of Figure 6.15 represent the five preventive maintenance check boxes previously described.

Block Name	Frequency	Duration	Startup	SRB	RNB	DMB	DLB	FRB
Attitude	1500.000000	Trng (2.166667)	0.000000	True	True	False	True	True
BatteriesNaS	900W-PowerString	Norm (4.000000)	N/A	False	True	False	True	N/A
BatteriesNiCd	500W-PowerString	Norm (8.000000)	N/A	False	True	False	True	N/A
BatteriesNiH	650W-PowerString	Norm (6.000000)	N/A	False	True	False	True	N/A
SolarP_500W	500W-PowerString	Trng (1.500000)	N/A	False	False	False	True	N/A
SolarP_650W	650W-PowerString	Trng (1.500000)	N/A	False	False	False	True	N/A
SolarP_900W	900W-PowerString	Trng (1.500000)	N/A	False	False	False	True	N/A

Figure 6.15 Preventive Maintenance Tab of the Block Input Tables Dialog Box

A global view of the triggers can be obtained from the *Options – Tables – View System Inputs* menu item. This will bring up the System Settings Input Tables dialog box, which provides an overview of the system topology on the first tab. The last tab displays all the triggers contained within a model and their attributes as shown in Figure 6.16. Additionally, this dialog box provides a global view of the spare pools and resources associated with a particular model.

Open the Simulation Options dialog box and select the Advanced tab as displayed in Figure 6.17. Check the “Start the simulation in step mode” checkbox, which will allow us to view the effects of the preventive maintenance that we specified in this chapter. This simulation will start with the simulation halted at time zero and will only advance when the Step button is clicked. Each time the Step button is clicked, the simulation will advance to the next event on the simulation calendar. Click the OK button to start the simulation.

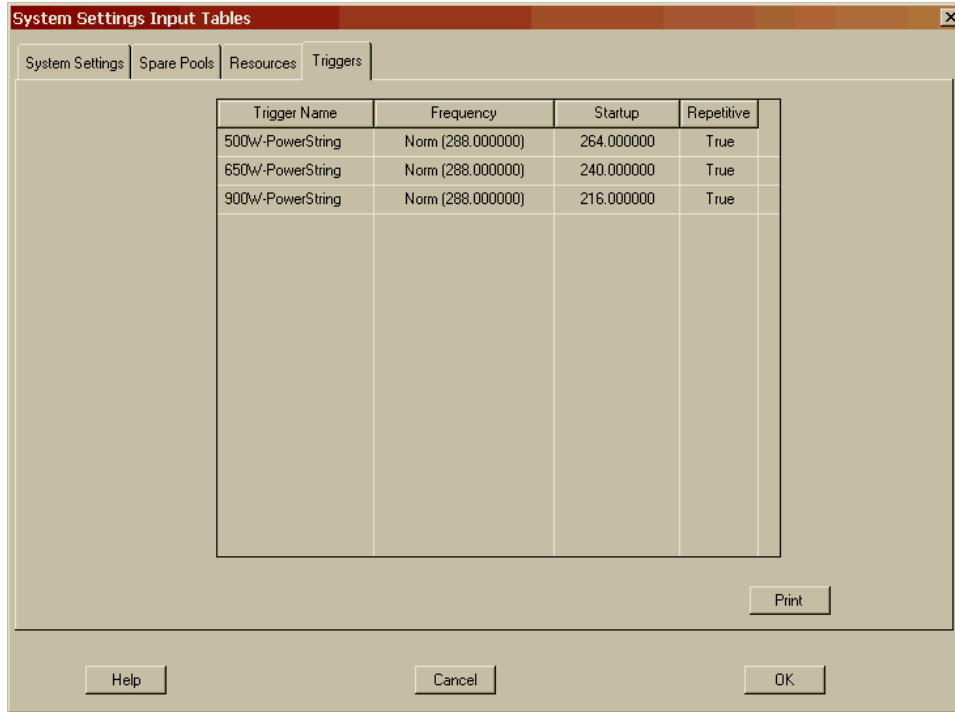


Figure 6.16 Global View of All Triggers

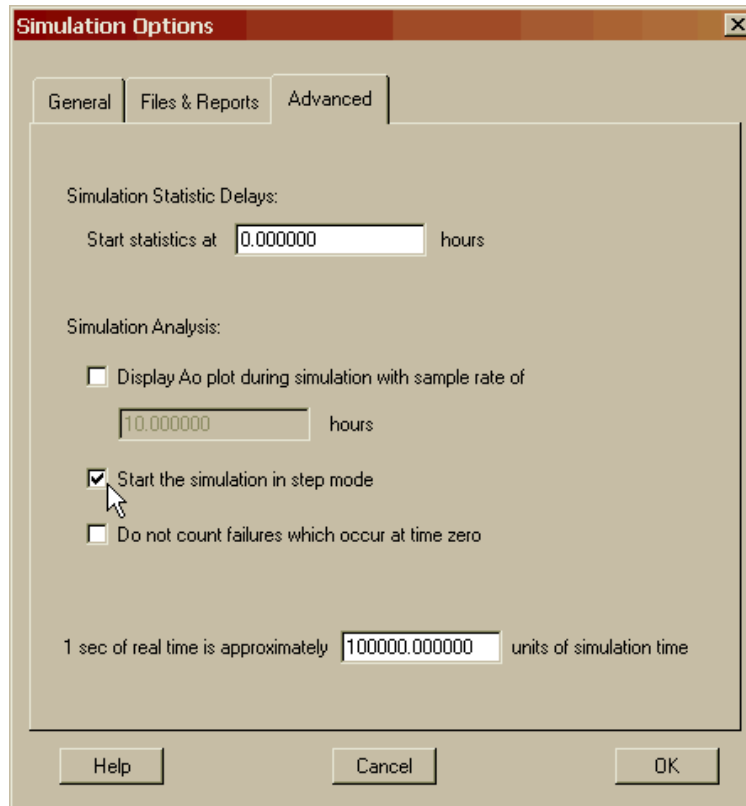


Figure 6.17 Advanced Tab of the Simulation Options Dialog Box

It should be noted that the Step mode function is often used to facilitate the detailed validation of numerous simulation effects. Clicking the depressed Pause button will place the simulation back into normal operation mode and thus progress without stopping through all of the events of a simulation. Clicking on the Jump button will change the simulation back to normal simulation speed and then stop at the next trial. This button is often used to jump to the n^{th} trial instead of clicking through each and every event contained with the first (n-1) trials. Figure 6.18 shows the Pause, Step and Jump buttons of the Simulation toolbar.



Figure 6.18 The Pause, Step and Jump Buttons of the Simulation Toolbar

Step once and notice that the 500 Watt solar panel and the Nickel-Cadmium batteries have turned to an orange color (Figure 6.19), which is the color that indicates a component is undergoing preventive maintenance activities.

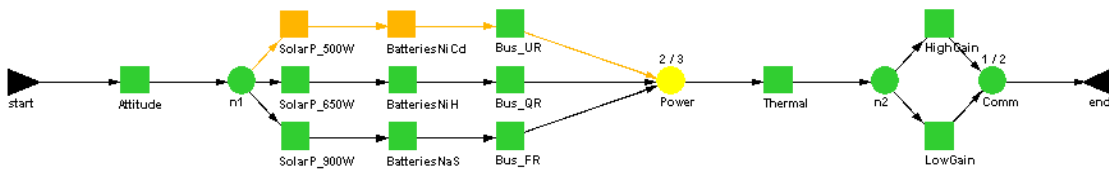


Figure 6.19 Color Status of Preventive Maintenance Activities

Take fifteen more steps and notice that the 500 Watt solar panel in Figure 6.20 is undergoing preventive maintenance but its trigger companion, the Nickel-Cadmium battery component, is not. Recall that we specified that a component must wait to engage its preventive maintenance action until enough resources are available. The failure of the Bus_QR component takes four of the five Astronauts and does not leave enough resources to conduct preventive maintenance on both components of the 500W-PowerString trigger group.

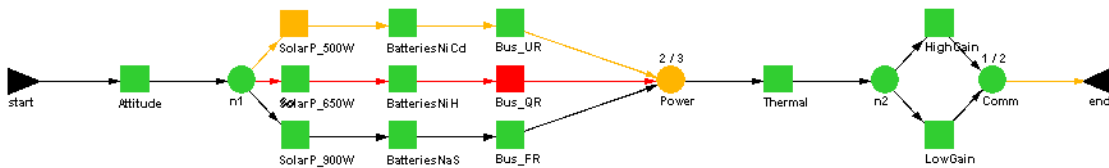


Figure 6.20 Effects of Deferring Preventive Maintenance Due to Lack of Resources

Click on the depressed Pause button and allow the simulation to complete. You should obtain results as shown in Figures 6.21 and 6.22

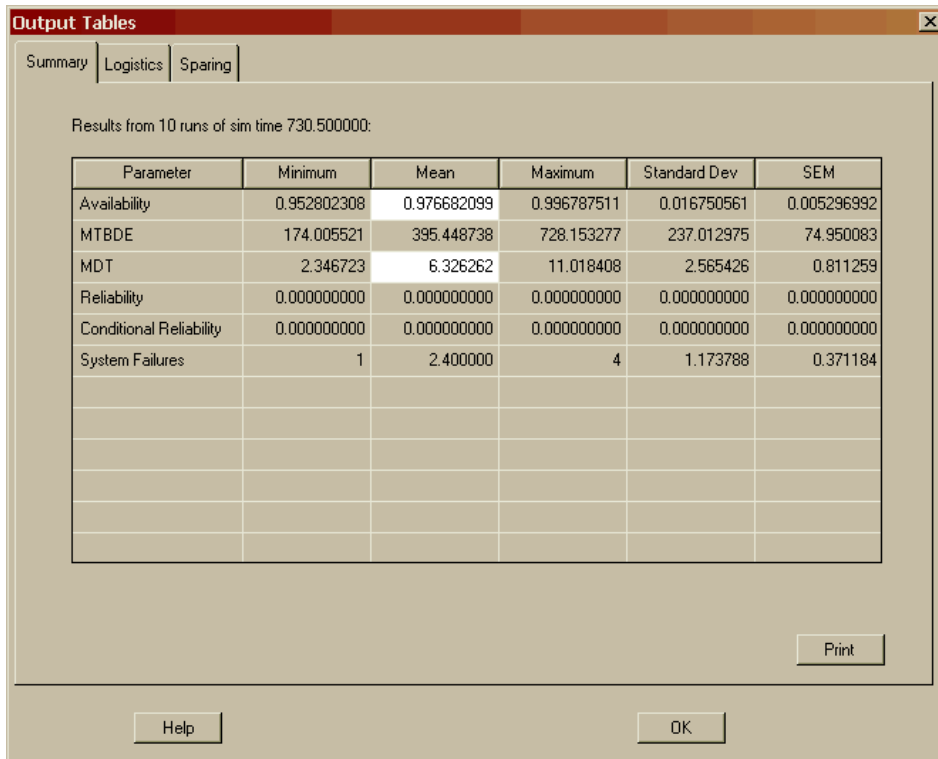


Figure 6.21 Summary Results for Chapter 6

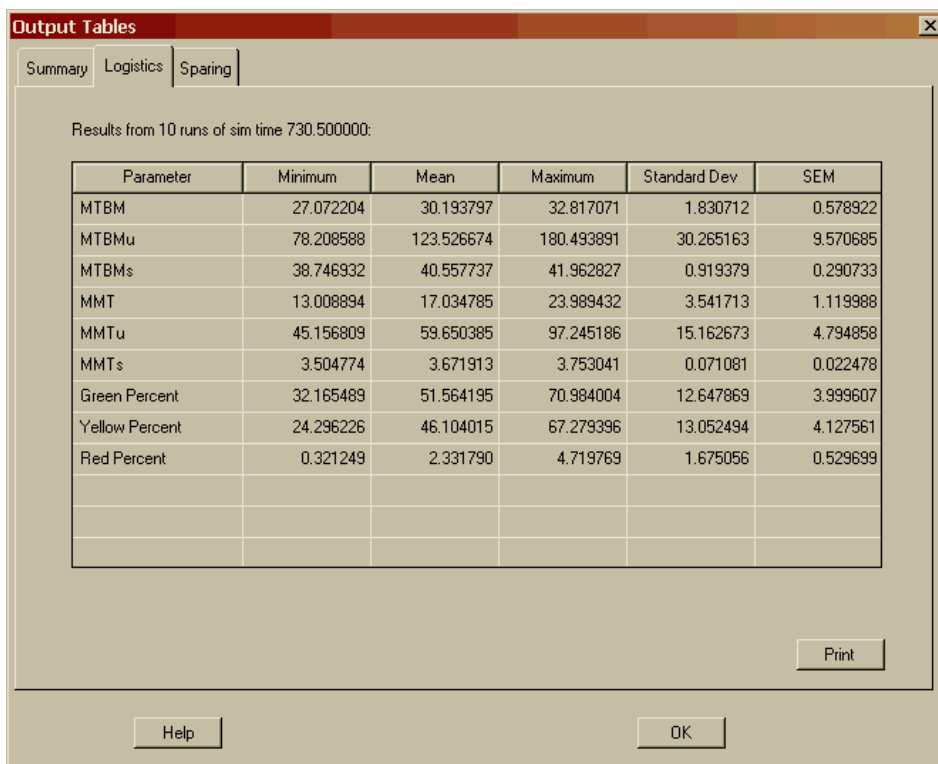


Figure 6.22 Logistics Results for Chapter 6

Both the availability and MDT improved for this model as compared to the results of Chapter 4 (0.944926395 and 33.590480, respectively) as would be expected since we occasionally avoid hard failures as a result of our preventive maintenance strategies. Figure 6.22 is showing for the first time results for both the MTBMs and MMTs parameters. The total MTBM is 30.193797 hours while the unscheduled and scheduled MTBM are 123.526674 and 40.557737 hours, respectively. It should be obvious that the total MTBM is not a straight average of the MTBMu and MTBMs parameters but a weighted average based on the number of each type of maintenance activity. Likewise, the total MMT is 17.034785 hours while the unscheduled and scheduled MMT are 59.650385 and 3.671913 hours, respectively.

• • •

It should be stressed that if a component has an inherent exponential failure rate, conducting preventive maintenance on such a component would make little sense. An exponentially distributed component possesses the memory-less property, which states that no matter how much life is exhausted on a component, it has nothing to do with when it will fail in the future. In the long run, performing preventive maintenance on an exponentially distributed component will not extend the expected time to the component's next failure and hence should not be performed. Occasionally though, an item with a wear-out failure mode that utilizes preventive maintenance early in its life is modeled using an exponential distribution.

Problems

1. How would you set up a random preventive maintenance frequency for a single component?
2. Does it make sense to define the Attitude's preventive maintenance frequency to be greater than the simulation length? If not, when would it?
3. Assume a component has a preventive maintenance frequency of 100 hours, a PM duration of ten hours and an elapsed time of 50 hours. When would the first and second preventive maintenance actions occur?
4. Assume a component belongs to a trigger group such that it has a preventive maintenance frequency of 100 hours, a duration of ten hours and an elapsed time of 50 hours. When would the first and second preventive maintenance actions occur?
5. Could a component's first preventive maintenance action occur at a random time given that the elapsed time field takes on a fixed value? If yes, explain how this would be set up.
6. State a component that requires a spare to perform its preventive maintenance action and one that does not.

7. State a component that is refreshed like a new component after its preventive maintenance action and one that is not.
8. What risk is being taken if the defer preventive maintenance until a spare and resources are available checkbox is unmarked?
9. State a component for which a hard failure would reset the time to its next preventive maintenance action, and one for which it would not.
10. Why would one model a component that receives preventive maintenance but does not refresh the component as good as new?

Chapter 7



Objectives

1. Understand the general concept of dependence.
2. Understand the difference between system dependence, local dependence and elemental dependence.
3. Understand the different states a component that is dependent on another can be sent to.
4. Understand the role of nodes when they are used with dependence.
5. Be able to build or modify an RBD to properly model dependence.

Recommended Problems

3, 4, 5, 7 and 8

Dependence

One aspect of modeling a system that is important, but is nevertheless often completely ignored during RBD development, is dependence. At its core, dependence is completely embodied with the following question: “Do the components in my system continue to operate in their own failure-repair cycles regardless of what happens elsewhere within the system, or do these components stop operating based on the status of other components?”

Consider a simple dependent system consisting of a power supply and an electric motor. The major failure mode of the electric motor is its brushes, which exhibit wear and fail after approximately 100 hours of operation. The power supply is unreliable and only operates about 50% of the time. When the power supply fails and is being repaired, the motor stops rotating and wear on the brushes does not accumulate. Thus, the motor is dependent on the power supply to operate. When the motor is stopped from operating because of the failure of the power supply, the motor is said to be in an idle status. The motor will transition between operating and being placed in a dependent state until it has accumulated 100 hours of operating time. On average, the electric motor will fail every 200 hours due to the power supply's 50% availability to function and not 100 hours as might be expected.

Figure 7.1 displays the states that a component can be in when tasked to initiate its dependency rules and what states the component can ultimately reach and remain in until the dependency link is severed.

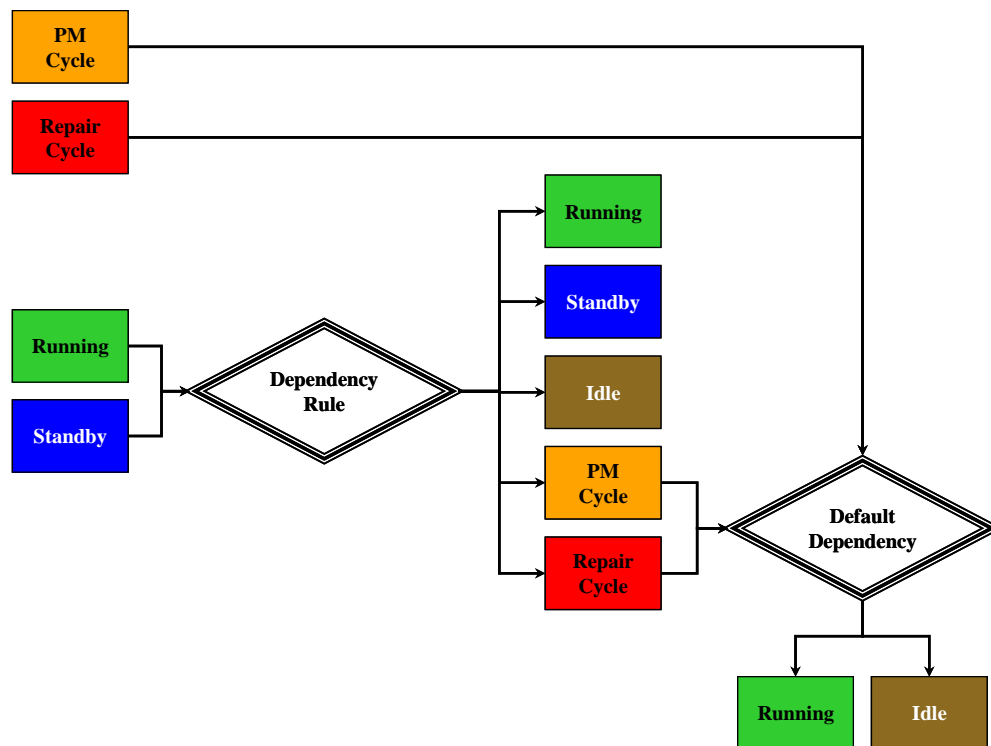


Figure 7.1 Dependency From- State-To- State Diagram

It is important to understand that only when a component fails can another component activate its own dependency rules. In the next chapter we will extend this linkage between components by introducing the combined effects of dependency and standby conditions. A component can possess one of four types of dependency. A component can be independent, system dependent, elementally dependent or locally dependent. A new component placed within the Workspace View has a default of independence unless changed via the *Block Defaults* menu option of the *Edit* menu. A component's dependence type is selected from the Dependency tab of the Block Properties dialog box as shown in Figure 7.2.

A component that is declared independent will fail and repair according to its failure and repair distributions without any regard to outside dependency influences. Even if the system itself is down, an independent component will continue to operate and thus accumulate hours towards its next failure. A component that is system dependent will be affected only by the status of the system. Every component that is dependent upon the system will initiate its own dependency rules when the system is down.

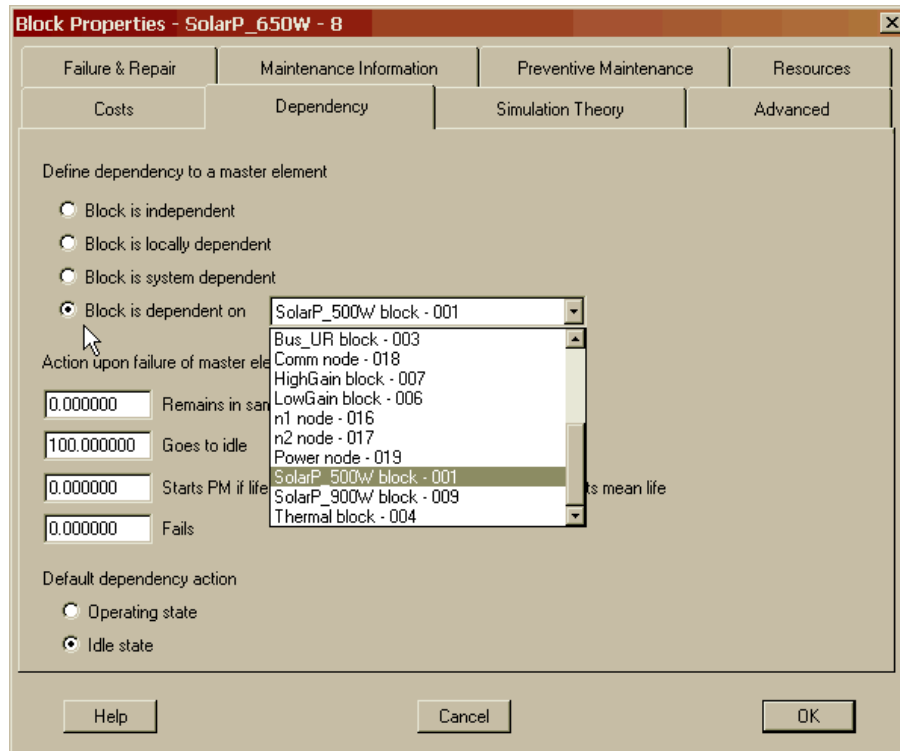


Figure 7.2 Dependency Tab of the Block Properties Dialog Box

A component that is elementally dependent will be affected only by the status of a particular element. A component can only be elementally dependent upon a single element. Local dependence is similar to system dependence, but on a subsystem-type level. For a block, being locally dependent is equivalent to being elementally dependent on the next downstream node. For a node or hierarchy, local dependence is equivalent to being elementally dependent on the first downstream node that is able to

withstand the failure of the node in question; that is, everything else being up, it is the first node downstream that would not go red with the failure of the dependent node. Local dependence is a good choice when the user wants components to go to an idle state when their impact on the system is not necessary. This option is also convenient for creating a subsystem with internal dependencies that is then copied and pasted multiple times. Thus, each copied subsystem will have the same internal dependencies defined.

Let us begin to manipulate the dependency aspects of an RBD to better understand the essence of the dependency concept. Open **Dependence1.rbd**, and then display the Dependency tab of the Block Input Tables dialog box as shown in Figure 7.3. As you can see, some dependency rules have already been employed in this model compared to the previous model (recall that all components in Chapter 6 were defaulted to be independent). We will implement some additional dependency rules for the Attitude, the 500 Watt solar panel and the 900 Watt solar panel components. In fact, we will build a system, local and elemental dependency relationship to ensure each type of dependency is clarified.

Block Name	Type	% Same	% Idle	% PM	If Exhaust	% Fail	Default
Attitude	Independent	N/A	N/A	N/A	N/A	N/A	N/A
BatteriesNaS	900path Node	20.000000	60.000000	10.000000	75.000000	10.000000	Idle
BatteriesNiCd	Locally Dependent	0.000000	90.000000	0.000000	75.000000	10.000000	Idle
BatteriesNiH	Locally Dependent	10.000000	80.000000	0.000000	75.000000	10.000000	Idle
Bus_FR	900path Node	20.000000	70.000000	0.000000	75.000000	10.000000	Idle
Bus_QR	Locally Dependent	10.000000	80.000000	0.000000	75.000000	10.000000	Idle
Bus_UR	Locally Dependent	0.000000	90.000000	0.000000	75.000000	10.000000	Idle
HighGain	System Dependent	0.000000	100.000000	0.000000	75.000000	0.000000	Idle
LowGain	System Dependent	0.000000	100.000000	0.000000	75.000000	0.000000	Idle
SolarP_500W	Independent	N/A	N/A	N/A	N/A	N/A	N/A
SolarP_650W	Locally Dependent	10.000000	80.000000	0.000000	75.000000	10.000000	Idle
SolarP_900W	Independent	N/A	N/A	N/A	N/A	N/A	N/A
Thermal	Independent	N/A	N/A	N/A	N/A	N/A	N/A

Figure 7.3 Dependence1.rbd

It is often difficult to understand dependency since its construction within Raptor is abstract with no visual feedback in the Workspace View. To alleviate this situation, it is recommended that users develop dependency diagrams to facilitate their model verification efforts. Figure 7.4 shows a dependency diagram of the **Dependence1.rbd**

model. Notice that the links that provide topology logic for the RBD have been removed and colored arcs have been added. The yellow arcs represent components that are system dependent. Thus, when the system goes down (i.e., a red status indicator is displayed) the high and low gain antennas will engage their dependency rules. The purple arcs represent components that are locally dependent and thus tied to special nodes that control the status of localized areas within an RBD topology. For example, if the 500path node was to go down, the nickel-cadmium battery and unregulated bus components would engage their dependency rules. The red arcs represent components that are elementally dependent and hence tied to a user defined element other than itself or the system.

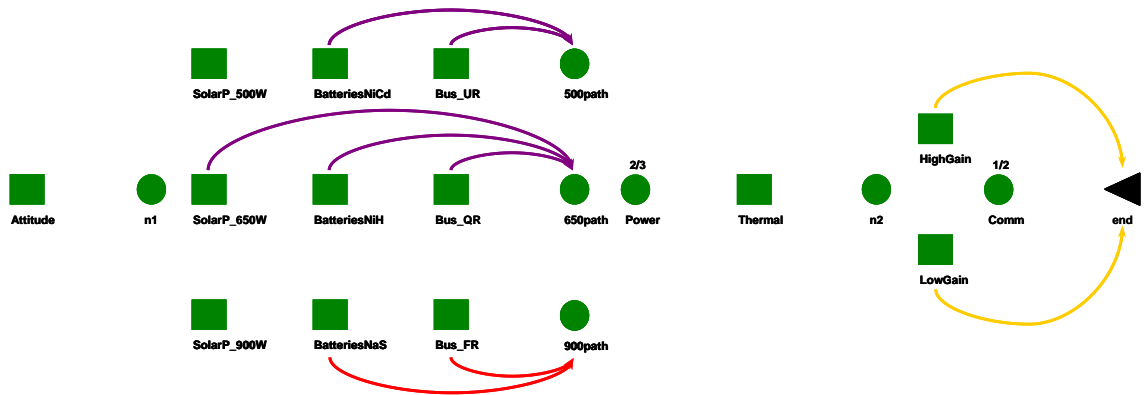


Figure 7.4 Dependence Diagram for Dependence1.rbd

Hence, Figure 7.4 displays all of the dependencies that have thus far been embedded into the **Dependence1.rbd** model. We will add three additional dependencies to this model as shown in Figure 7.5.

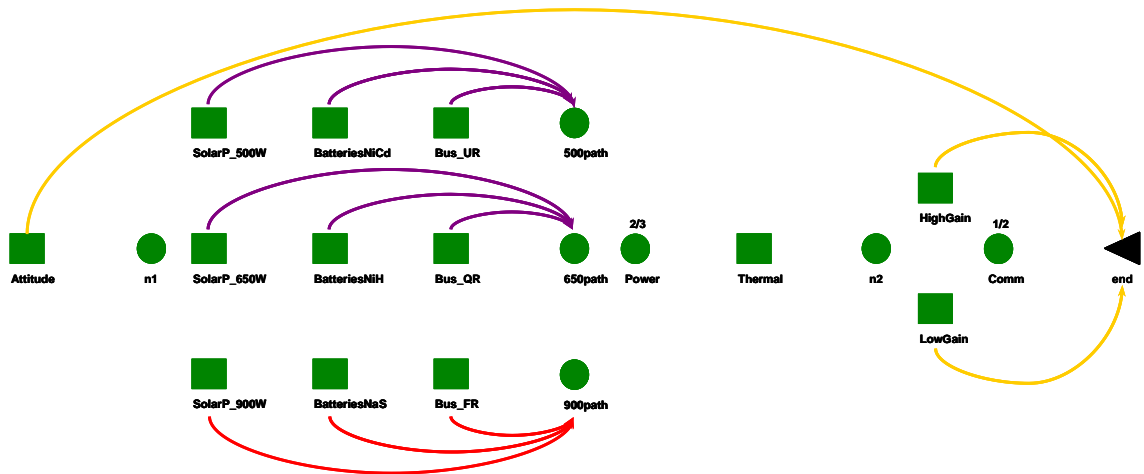


Figure 7.5 Added Dependencies to Dependence1.rbd

Double click on the Attitude component to open its Block Properties dialog box and select the Dependency tab. Currently, the Attitude component's dependency type has been designated as independent. Change the Attitude's dependency type to system dependent and modify its dependency rules as shown in Figure 7.6.

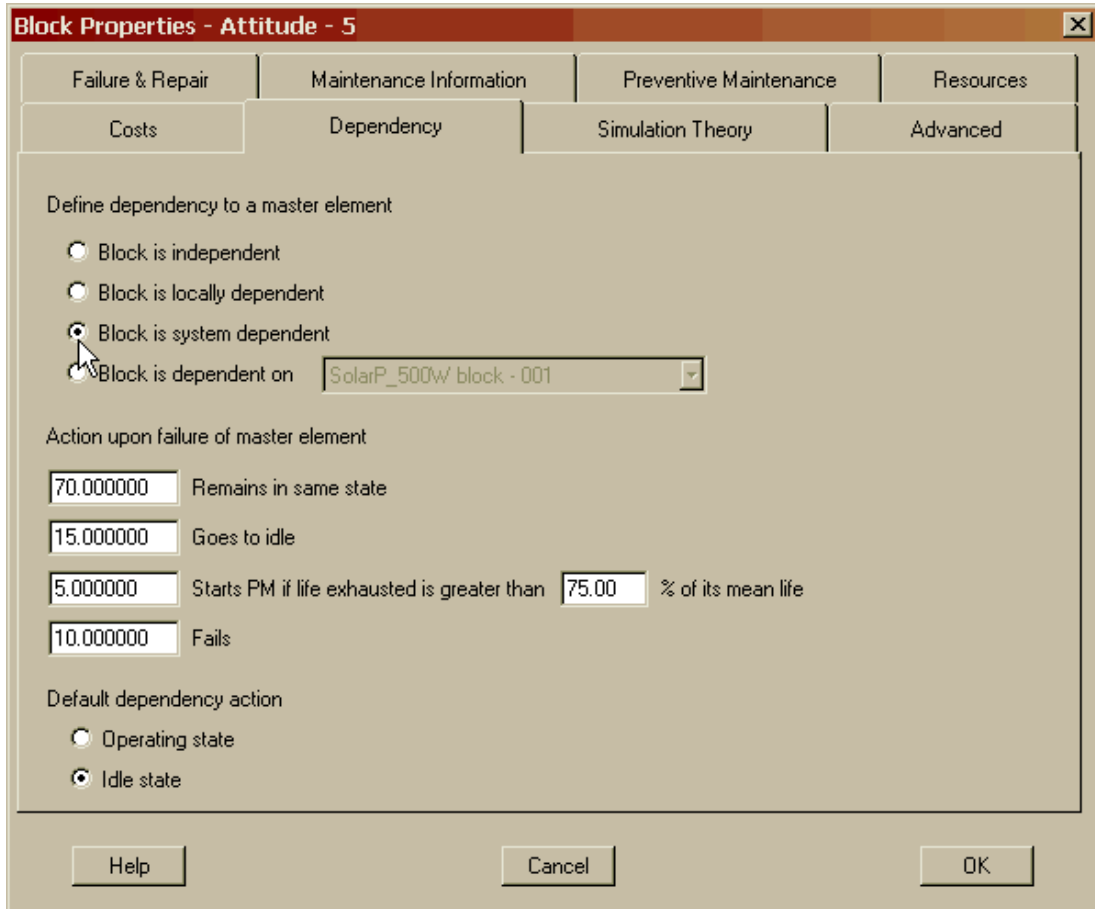


Figure 7.6 System Dependent Attitude Component

When the system goes down, the Attitude component will remain operating seventy percent of the time as if there were no dependency effects at all. Fifteen percent of the time the Attitude component will stop operating and be placed into an idle status awaiting its master element (i.e., the system in this case) to be returned to an operating status. Five percent of the time this component will engage its preventive maintenance activities as described in Chapter 6 provided that seventy-five percent or more of the Attitude's mean expected life is currently exhausted. If this last condition is not met, the component will engage its default dependency action (currently set to the idle state for this component), which will be discussed in greater detail later in this chapter. Sending a component into a preventive maintenance status when its master element has stopped operating is often called *opportunistic maintenance*. The remaining ten percent of the time, the Attitude component will fail as a result of its master element failing. This scenario is known as *induced* or *precipitated failures*.

Change the 500 Watt solar panel dependency type to locally dependent and modify its dependency rules as shown in Figure 7.7. The 500 Watt solar panel component is now dependent on the 500path node as are the other two components of the 500path string. If the 500 Watt solar panel, the nickel-cadmium batteries or the unregulated bus component fails, the other two components will employ their respective dependency rules. This will occur since the 500path node controls the dependency aspects of the localized series topology. While a node itself cannot go down, when any one of these series components fails, the 500path string has failed. Once this has occurred, the 500path node then instructs all elements dependent on it to engage their dependency rules.

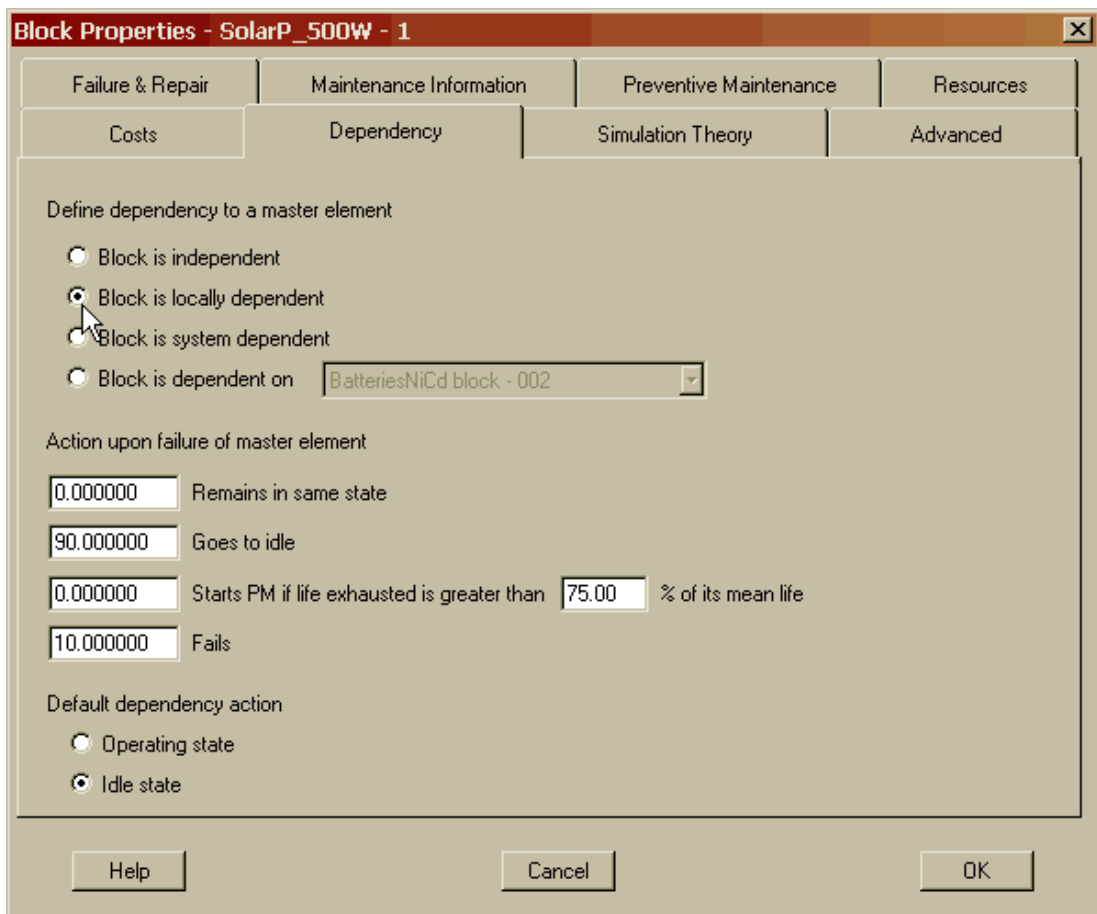


Figure 7.7 Locally Dependent 500 Watt Component

When instructed by the 500path node, the 500 Watt component will go into an idle status ninety percent of the time. A component that is idle does not acquire life and thus acts as if it is in suspended animation. Ten percent of the time, this component will be induced to fail itself as a result of either the nickel-cadmium batteries or the unregulated bus component failing. This component cannot remain operating or go into its preventive maintenance state as a result of a dependency event since both of these dependency rules have been set to zero percent.

Lastly, change the 900 Watt solar panel dependency type to elementally dependent and modify its dependency rules as shown in Figure 7.8. The 900 Watt solar panel component is now dependent on the 900path node as are the other two components of the 900path string. If the 900 Watt solar panel, the sodium-sulfate batteries or the fully-regulated bus component fails, the other two components will employ their respective dependency rules. Again, this will occur since the 900path node controls the dependency aspects of the localized series topology.

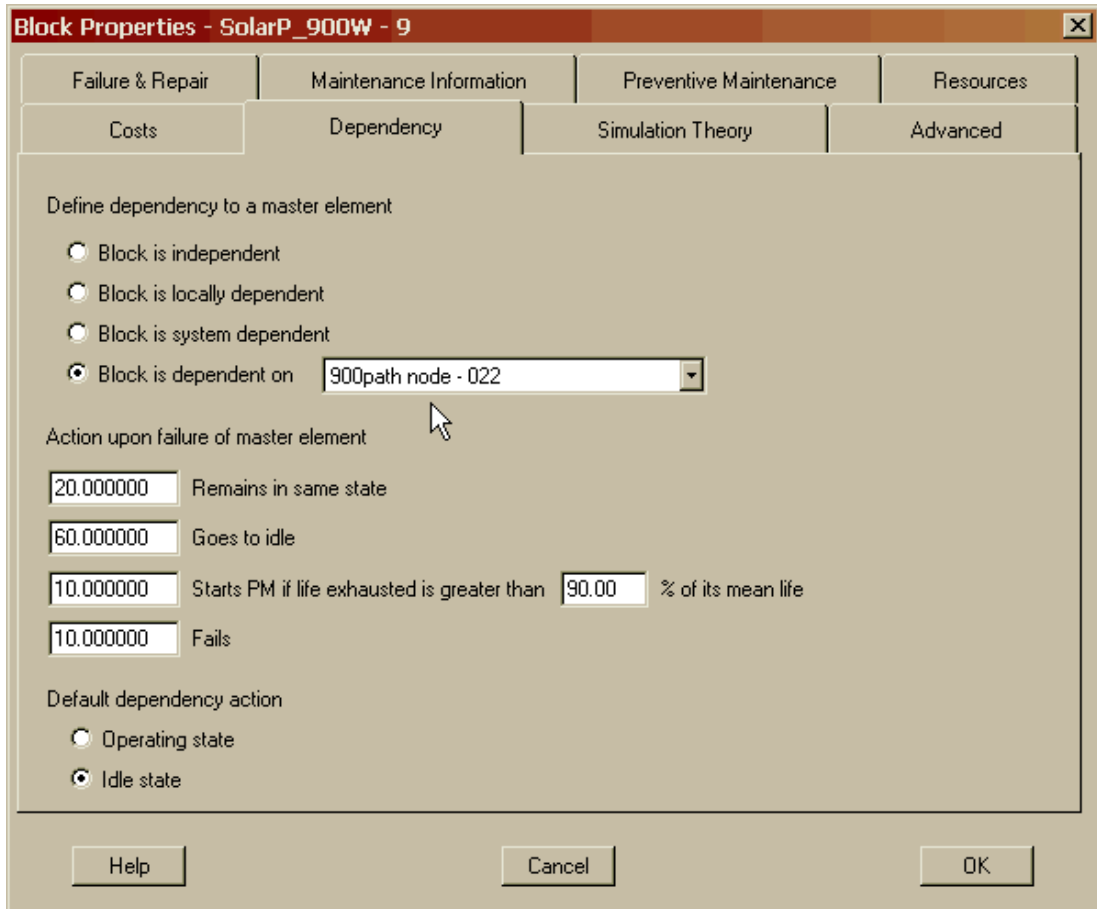


Figure 7.8 Elementally Dependent 900 Watt Component

When instructed by the 900path node, the 900 Watt component will remain operating twenty percent of the time as if there were no dependency effects at all. Sixty percent of the time the 900 Watt component will stop operating and be placed into an idle status. Ten percent of the time this component will engage its preventive maintenance activities provided that ninety percent or more of the 900 Watt component's mean expected life is currently exhausted. The remaining ten percent of the time, this component will be induced to fail itself as a result of either the sodium-sulfate batteries or the fully-regulated bus component failing.

Notice that all of the components set in this model have their default dependency actions set to the idle state (as shown in Figures 7.6, 7.7 and 7.8). This option ensures

that a component that is exiting its repair cycle and dependent upon an element that is currently failed will be sent into an idle state. Thus, the dependency percentage rules are only employed when a component is operating (i.e., not repairing) and its master element has gone down. The default dependency action could also be specified to be the operating state.

It is interesting to note that the local dependence setting for the 500 Watt solar panel has the same dependency affect as the elemental dependence of the 900 Watt solar panel. Both components are dependent on their respective downstream node and both of these configurations are often called string dependence, since if any member of a string fails all other good members engage their dependency rules. If all dependencies have been set correctly, the Dependency tab of the Block Input Tables dialog box should be identical to Figure 7.9. We have thus augmented our model from the dependencies shown in Figure 7.4 to those in Figure 7.5.

Block Name	Type	% Same	% Idle	% PM	If Exhaust	% Fail	Default
Attitude	System Dependent	70.000000	15.000000	5.000000	75.000000	10.000000	Idle
BatteriesNaS	900path Node	20.000000	60.000000	10.000000	75.000000	10.000000	Idle
BatteriesNiCd	Locally Dependent	0.000000	90.000000	0.000000	75.000000	10.000000	Idle
BatteriesNiH	Locally Dependent	10.000000	80.000000	0.000000	75.000000	10.000000	Idle
Bus_FR	900path Node	20.000000	70.000000	0.000000	75.000000	10.000000	Idle
Bus_QR	Locally Dependent	10.000000	80.000000	0.000000	75.000000	10.000000	Idle
Bus_UR	Locally Dependent	0.000000	90.000000	0.000000	75.000000	10.000000	Idle
HighGain	System Dependent	0.000000	100.000000	0.000000	75.000000	0.000000	Idle
LowGain	System Dependent	0.000000	100.000000	0.000000	75.000000	0.000000	Idle
SolarP_500W	Locally Dependent	0.000000	90.000000	0.000000	75.000000	10.000000	Idle
SolarP_650W	Locally Dependent	10.000000	80.000000	0.000000	75.000000	10.000000	Idle
SolarP_900W	900path Node	20.000000	60.000000	10.000000	90.000000	10.000000	Idle
Thermal	Independent	N/A	N/A	N/A	N/A	N/A	N/A

Figure 7.9 Dependency Tab of the Block Input Tables

Simulate this model in step mode and take a single step such that the 500 Watt solar panel and the nickel-cadmium components have engaged their preventive maintenance activities as shown in Figure 7.10. Notice that the 500path node has gone into a preventive maintenance status due to the 500 Watt solar panel and the nickel-cadmium components being triggered to a preventive maintenance status. Further notice that the unregulated bus has changed its color to sienna, which is the status color for elements

that are idle as a result of their dependency rules. Recall from Figure 7.9 that the unregulated bus goes into an idle state ninety percent of the time. Figure 7.10 is an example of a locally dependent component.

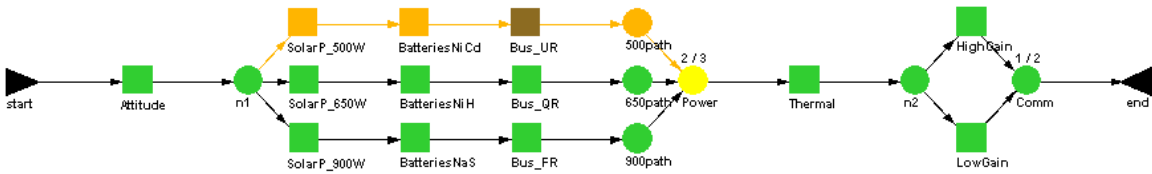


Figure 7.10 Local Dependence in Action

Take another step and notice in Figure 7.11 that the 500 Watt solar panel has finished its preventive maintenance action and is then sent to the idle state due to its default dependency criteria.

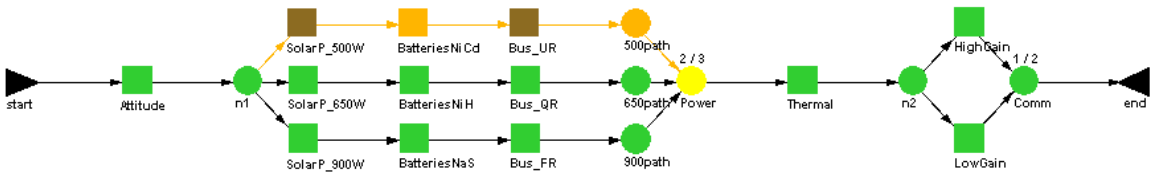


Figure 7.11 Default Dependency in Action

Click five more times on the Step button and note in Figure 7.12 that the 900 Watt solar panel and the sodium-sulfate components have been triggered to engage their preventive maintenance. Furthermore, the status of these components has caused the fully-regulated bus to fail by induction as a result of the elemental dependency relationships within the 900path power string. Recall that the fully-regulated bus will be induced to fail ten percent of the time due to its elemental dependency relationship with the 900path node.

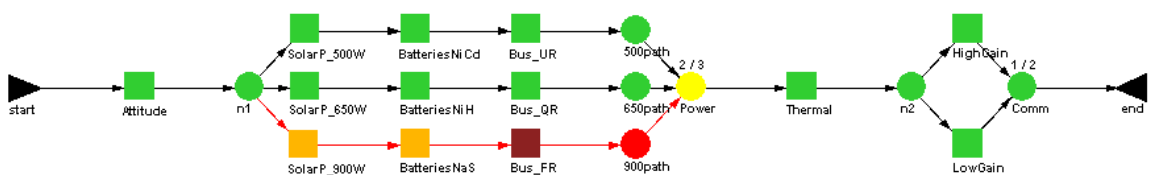


Figure 7.12 Elemental Dependence in Action

Skip to the 7th trial by clicking the Jump button six times and then take four steps (i.e., simulation time 76.556965) such that your RBD appears as the one shown in Figure 7.13. Be sure to note the status of all components during each step.

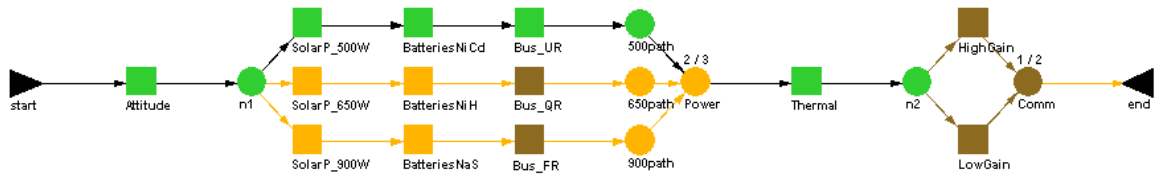


Figure 7.13 System Dependence in Action

To understand the situation in Figure 7.13 you would have had to notice that the unregulated bus failed at simulation time 24.480330. The unregulated bus failed as a result of the 500W-PowerString preventive maintenance trigger firing and inducing the unregulated bus to fail. The unregulated bus, however, cannot begin its repair action since it requires five astronaut resources and two of them are currently occupied (one each for the 500 Watt solar panel and the nickel-cadmium battery components due to their preventive maintenance activities). Since the unregulated bus cannot start its repair cycle the preventive maintenance actions for the 650 Watt and 900 Watt power strings are deferred when triggered at simulation time 48.808960. Recall from Chapter 6 that the components that comprise the power string triggers will defer their preventive maintenance actions until enough spares and resources are available.

Rather quickly, the 500 Watt solar panel and the nickel-cadmium battery components complete their preventive maintenance activities and at simulation time 31.731446 the unregulated bus begins its hands-on repair yet the 650 Watt and 900 Watt power strings are still deferred since the unregulated bus requires the entire astronaut repair crew to conduct its repair actions. Once the unregulated bus releases its resources at simulation time 76.556965, the 650 Watt and 900 Watt power string preventive maintenance triggers are implemented. This causes the 650 Watt solar panel, the nickel-hydrogen battery, the 900 Watt solar panel and the sodium-sulfate battery components to each obtain one resource and begin their preventive maintenance activities. The quasi-regulated and fully-regulated buses are sent into an idle status as a result of the 650 Watt and 900 Watt triggers. Furthermore, the entire system has gone down since the Power node has been rendered inactive due to preventive maintenance activities. This causes all system dependent components (the Attitude and the high and low gain antennas) to determine their dependency status. Recall from Figure 7.9 that the high and low gain antennas go to an idle state 100% of the time while the likelihood of the Attitude component going to an idle status is only 15%.

Figure 7.13 is a complicated example of system dependency issues coupled with preventive maintenance activities and logistical constraints. The local, elemental and system dependency types have now all been demonstrated in Figures 7.10, 7.12 and 7.13, respectively. Click on the depressed Paused button and allow the simulation to run unabated. View the results as shown in Figure 7.14.

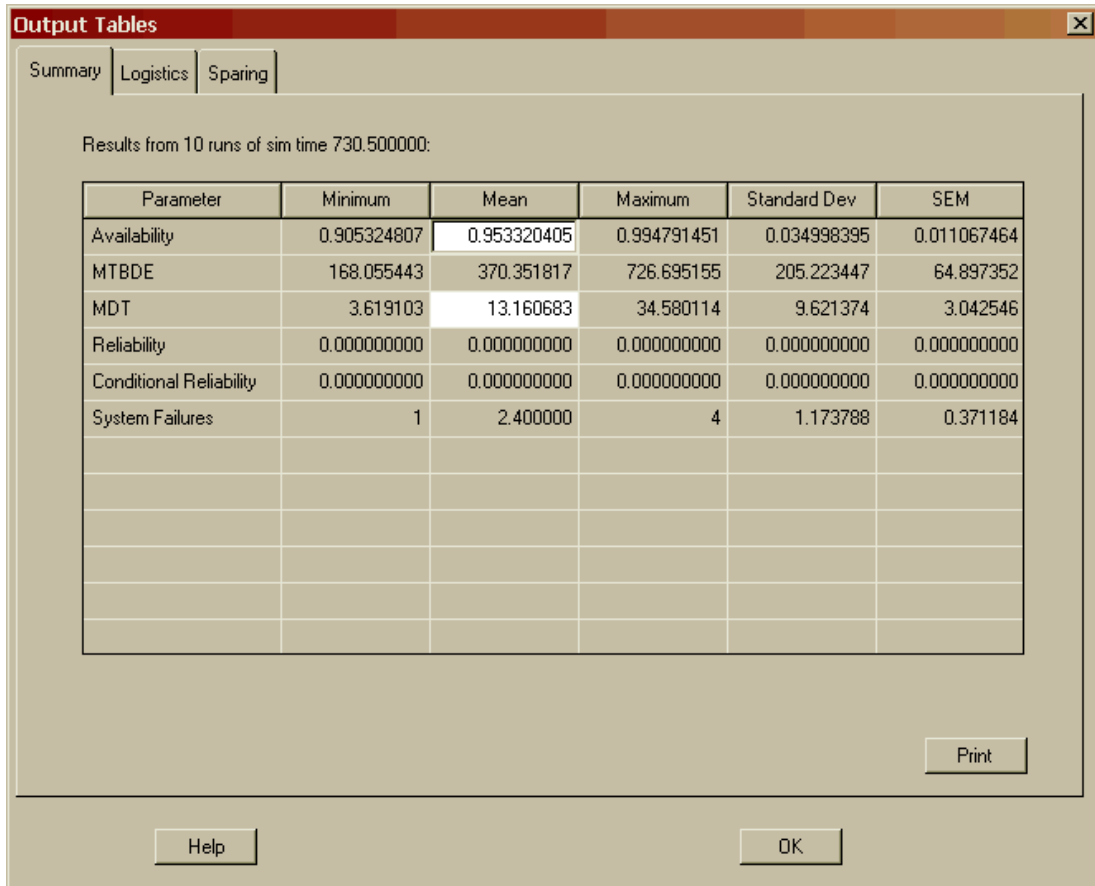


Figure 7.14 Summary Results from Chapter 7

You cannot directly compare these results to those in Chapter 6 since the addition of dependent relationships dramatically alters this model. We can say, however, that the results shown in Figure 7.14 (i.e., an availability of 0.953320405 and a mean down time of 13.160683 hours) are more accurate since this model contains more fidelity with respect to operational realism.

Some may argue that deferring the power string triggers until resources are available is unwise since it caused a large drain on the maintenance personnel when they all came due at the same time. Yet, one must realize that this system remained operating for more than 27 hours that it otherwise would have been down if the preventive maintenance actions were not deferred. That is nearly 4% of the total operating time of this system and thus the deferring of the preventive maintenance seems advantageous.

Determining the sequence of events for Figure 7.13 can often be quite difficult and in most complex cases can only be determined explicitly by viewing the detailed event log. This can be generated by selecting the Detailed event log checkbox of the Files & Reports tab of the Simulation Options dialog box as shown in Figure 7.15. A portion of this log that is relevant to Figure 7.13 is shown in Figure 7.16.

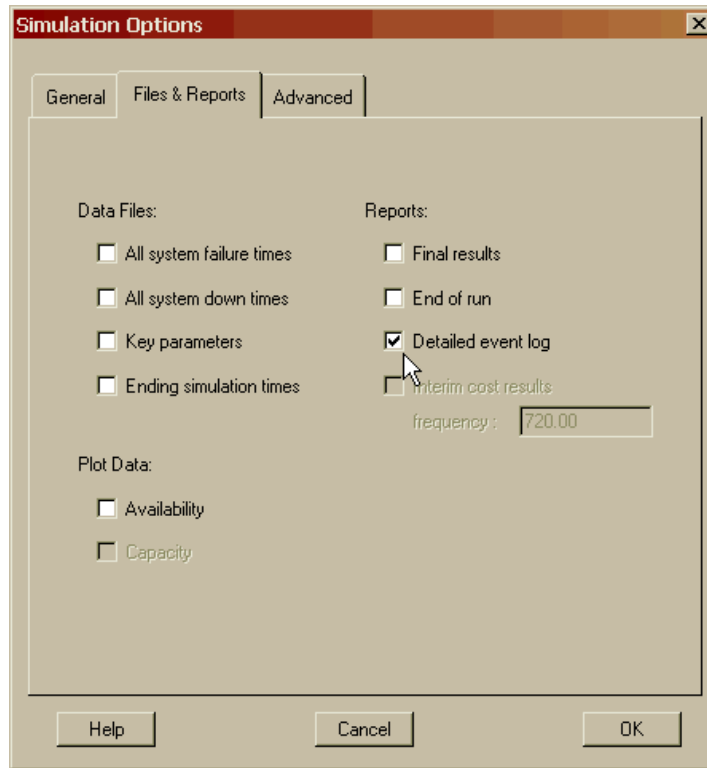


Figure 7.15 Generating the Detailed Event Log

26.140972	Block	SolarP_500W	PM_complete	MaintTime=1.660642
26.140972	Block	SolarP_500W	Idle	change_block_state
27.480330	Block	Bus_UR	End_Pre-LDT	Time=3.000000
31.731446	Block	BatteriesNiCd	PM_complete	MaintTime=7.251117
31.731446	Block	BatteriesNiCd	Idle	change_block_state
31.731446	Block	Bus_UR	Sparing	Obtained_5_Resource(s)
31.731446	Block	Bus_UR	Repairing	5Astronauts
48.808960	Trigger	650W-PowerString	Fired	System_event
48.808960	Block	SolarP_650W	PM_is_due	Triggered_PM_Due
48.808960	Block	BatteriesNiH	PM_is_due	Triggered_PM_Due
48.808960	Block	SolarP_650W	pmReqDefer	No_resource_in_stock-Astronauts
48.808960	Block	BatteriesNiH	pmReqDefer	No_resource_in_stock-Astronauts
73.509577	Trigger	900W-PowerString	Fired	System_event
73.509577	Block	SolarP_900W	PM_is_due	Triggered_PM_Due
73.509577	Block	BatteriesNaS	PM_is_due	Triggered_PM_Due
73.509577	Block	SolarP_900W	pmReqDefer	No_resource_in_stock-Astronauts
73.509577	Block	BatteriesNaS	pmReqDefer	No_resource_in_stock-Astronauts
76.556965	Block	Bus_UR	Repaired	MaintTime=44.825518
76.556965	Block	Bus_UR	Good	Stress=1.000000
76.556965	Node	500path	Idle	0-1/1_di19
76.556965	Node	Power	Degraded	2-2/3_ur22_ur23_di24
76.556965	Block	SolarP_500W	Good	DependenceStress=1.000000
76.556965	Block	BatteriesNiCd	Good	DependenceStress=1.000000
76.556965	Node	500path	Good	1-1/1_ur19
76.556965	Node	Power	Good	3-2/3_ur22_ur23_ur24
76.556965	Block	SolarP_650W	Resources	Obtained_1_Resource(s)
76.556965	Block	BatteriesNiH	Resources	Obtained_1_Resource(s)
76.556965	Block	SolarP_900W	Resources	Obtained_1_Resource(s)
76.556965	Block	BatteriesNaS	Resources	Obtained_1_Resource(s)

Figure 7.16 Portion of a Detailed Event Log

Figure 7.16 clearly shows that at simulation time 48.808960 the 650W-PowerString trigger fired but the preventive maintenance on the 650 Watt and nickel-hydrogen battery components was deferred. Likewise at simulation time 73.509577, the 900W-PowerString trigger fired but the preventive maintenance on the 900 Watt and sodium-sulfate battery components was deferred. At simulation time 76.556965 the unregulated bus is repaired freeing up the astronaut resources so that all of the deferred preventive maintenance activities can be accomplished.

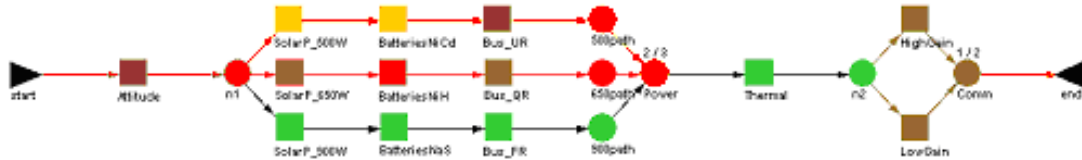
• • •

It should be noted that most reliability textbooks containing theoretical equations that determine the reliability and availability of particular configurations more often than not assume that all components operate independently. Improper use or appreciation of dependency concepts is a common error of novice reliability engineers, especially for those who attempt to compare dependent systems' results to independent systems' results.

Problems

1. Provide an example of a system that contains system dependent components.
2. Provide an example of a system that contains independent components.
3. Can several components within a string be “daisy chained” with elemental dependence (i.e., each component dependent on the component following it) as another method of implementing string dependence?
4. Can a component be dependent on an event?
5. Can an event be dependent on a component, node or another event?
6. What are some advantages of using local dependence instead of elemental dependence within a string if they both perform the same dependency function?
7. With the current configuration of **Dependence2.rbd**, if two strings were to fail the blocks in the remaining string will continue to operate. Change this RBD such that the components of a third functioning string will go into an idle status whenever two strings have failed.
8. Starting with the solution to Problem 7, further modify the RBD so that the power strings go idle if the system fails but they still retain their string dependence properties.

9. Explain how and why each non-green colored component is the color that it is as shown in the Figure below. Generate the detailed event log for **Dependence2.rbd** and review the steps of the ninth trial.



10. Consider a system with inherent dependencies. Suppose one has modeled this system without considering the effects of those dependencies. Do you suspect the true availability and reliability to be better or worse than the results that were obtained?

11. An electric motor consists of five components in series. The components, along with their failure and repair distributions, are given below.

Block Distribution Input Table

Block Name	Fail Distro	Param1	Param2	Param3	Mean
Armature	Weibull	1.75	174	0	155
Brushes	Weibull	2.3	112	0	99
Casing	Weibull	1.3	823	0	760
Magnets	Weibull	1.2	637	0	599
PowerSupply	Triangular	24	40	48	37

Block Name	Repair Distro	Param1	Param2	Param3	Mean
Armature	Pearson5	1.5	9	0	18
Brushes	Lognormal	23	2		23
Casing	Lognormal	13	1.75		13
Magnets	Lognormal	10	2		10
PowerSupply	Lognormal	8	5.5		8

What is the steady-state availability of the system if all components are independent?
 What is the steady-state availability of the system if the armature and brushes go idle upon a failure to the power supply?

Chapter 8



Objectives

1. Understand the concept of standby.
2. Be able to build or modify an RBD to establish standby relationships.
3. Understand the standby features of priority return and switching.
4. Understand how dependence options can be used with standby to model groups of standby components.

Recommended Problems

1, 2, 3, 5 and 6

Standby

Up to this point, we have seen how redundant components can be modeled using nodes with their *k-out-of-n* logic. When these components were connected in parallel (e.g., the high and low gain antennas in the previous RBDs) both components operated continuously. Occasionally, components are configured in a similar parallel relationship, but one in which both components do not operate simultaneously. One of the components, called the primary, is operating and the other, denoted the alternate, is in a standby state waiting to be called upon if needed. Raptor allows for the modeling of alternates that may accumulate life while being in a standby state.

When a component is in a standby state it is capable of operating but is not, as a result of a user's elective. A standby component is in a redundant relationship and will be called upon to operate when its associated alternates fail. A standby component is known as "cold standby" if no life is accumulated while acting as an alternate. A standby component is known as "warm standby" if life is accumulated at a lesser than normal operating rate while acting as an alternate. A standby component is known as "hot standby" if life is accumulated at a greater than normal operating rate while acting as an alternate. Figure 8.1 emphasizes the aforementioned information by expressing the standby states as stresses and the associated colors that will appear when simulating alternate components.




Standby Type	Stress	Color
Cold	$\pi_r = 0$	
Warm	$0 < \pi_r < 1$	
Hot	$\pi_r \geq 1$	

Figure 8.1 Standby Types, Stresses and Colors

Nodes are the controlling element for any standby structure. The status of a node is determined by the status of its inbound links, and therefore, a node has insight into how many of the immediately upstream elements are functioning at any given time. Raptor's standby logic takes advantage of this fact by allowing a node to determine when a sufficient number of good paths are present and then instructing excess components to transform to an alternate state.

This chapter will instruct a user on how to set up and properly utilize the various standby options. Begin by opening **Standby1.rbd** and double click on the Comm node to open the Node Properties dialog box. As Figure 8.2 indicates, select the Standby tab and mark the first checkbox, which engages the remainder of the tab. Additionally, be sure to mark the second checkbox. This ensures that if a primary component fails, an alternate component will take its place only until the primary component is repaired and then again operates as a primary. If this checkbox is not checked, alternates that become primaries will remain as primaries until they themselves fail. The last checkbox ensures that an attempt will be made to turn on each alternate using automatic switchover before resorting to manual switchover.

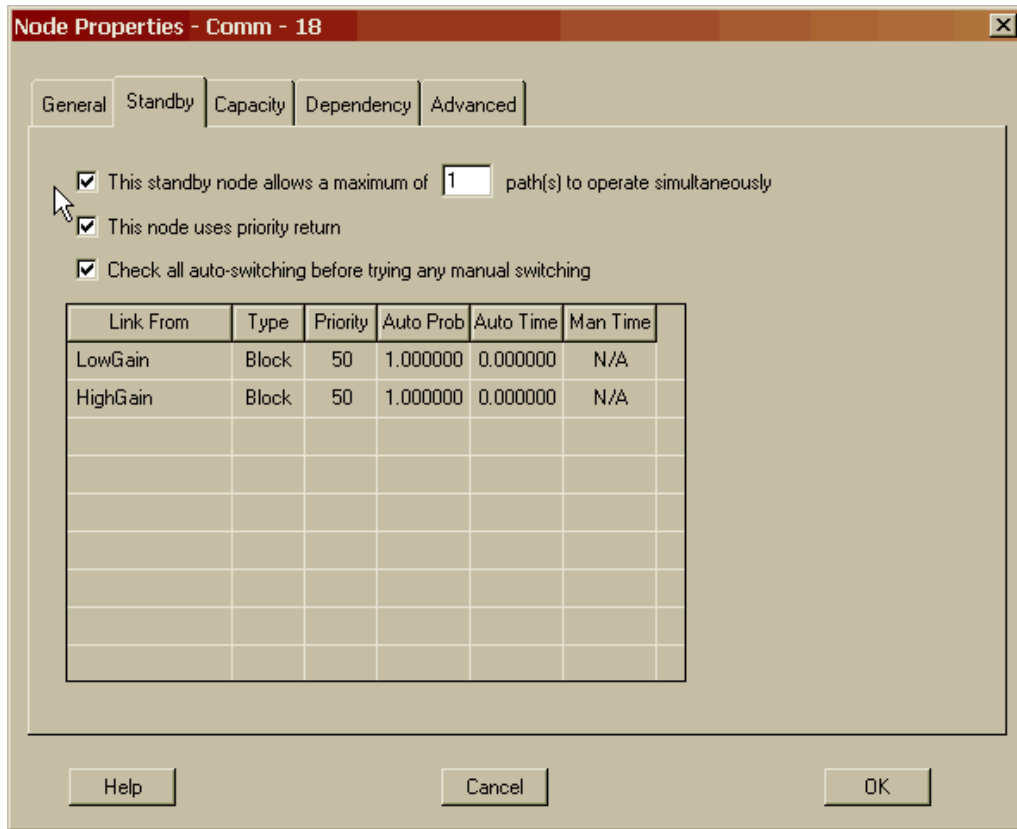


Figure 8.2 Standby Tab of the Node Properties Dialog Box

While a node's *k-value* is defined on the first tab of the Node Properties dialog box, a node's *k-star* value is set on the standby tab. The *k-star* value is the value box associated with the first checkbox. Recall that the node's *k-value* defines the minimum number of paths that must be operating for the node to be considered operating. The *k-star* value indicates the number of paths that a user would desire to have operating at anytime. For example, a user might need a minimum of three of seven paths to be operating (i.e., *k-value* equals three) but would desire six of seven paths to be operating (i.e., *k-star* value equals six). The value of *k-star* must be greater than or equal to *k*, and less than or equal to *n*. The default value for *k-star* is equal to *k*. If *k-star* equals *n*, then a standard redundancy topology exists where all components are

primaries. If k -star equals k , a standard standby redundancy topology exists. The k -star function is most often associated with capacity issues and will be discussed more in Chapters 13 and 14.

Highlight the high gain's priority cell (see Figure 8.3), which is currently displaying the value of fifty. Enter a value of twenty-five in the value box that appears and select the Accept Change button. This will make the high gain component the primary of this k -out-of- n structure and thus, the low gain component will be designated as the alternate. A lower priority number represents a higher priority. The highest priority is a value of one and the lowest is ninety-nine. The default priority value is fifty. Priorities are evaluated based on a relative ranking scheme such that a value of 10 has the same rank over a value of 50 as it does over the value of 51.

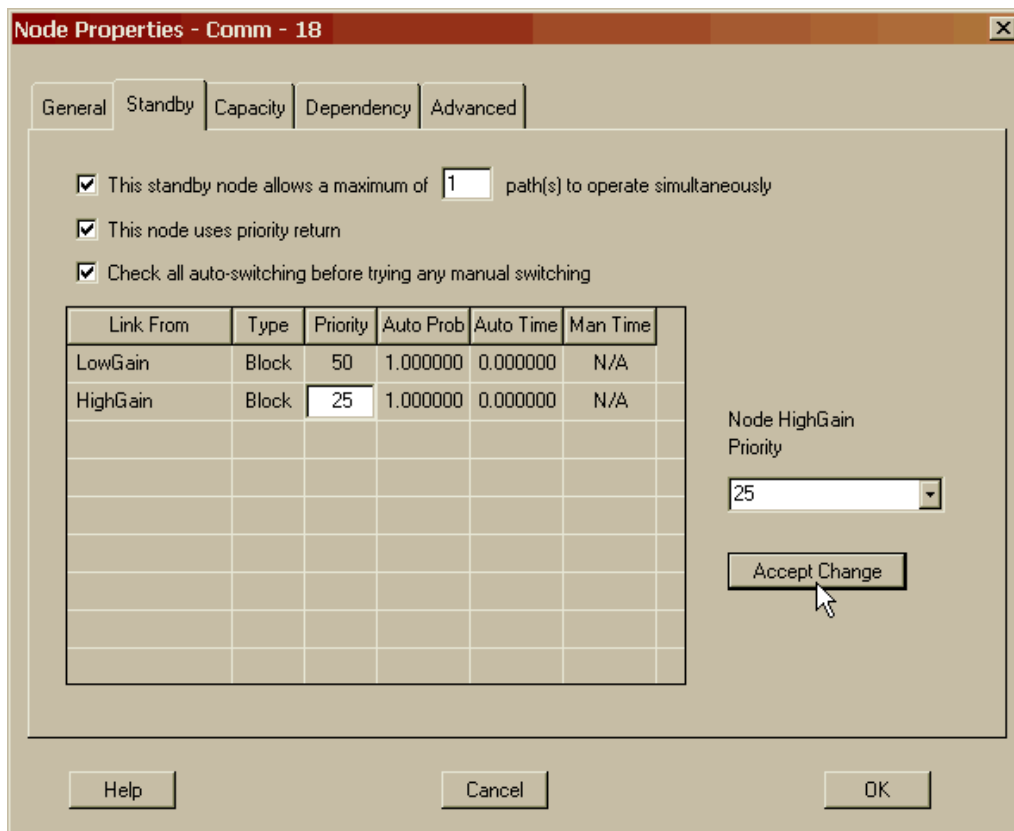


Figure 8.3 Changing a Node’s Standby Path Priority

Click in a cell under the “Auto Prob” column and note that the same value box is used to update other cells in this table. Use the value box and update the table until it appears as the one in Figure 8.4. The “Auto Prob” represents the probability that a component will automatically switchover to an alternate component in the time specified under the “Auto Time” column. Often, the auto-switch time is zero, allowing for an immediate switchover with no interruption in system performance. If the auto-switch fails, the switchover will manually switch in the time specified under the “Man Time” column. Manual switching may be set to “N/A” if manual switching is not

allowed. In this case, an alternate component would no longer be available for the remainder of a simulation trial. Continue to modify this tab such that it appears as in Figure 8.4.

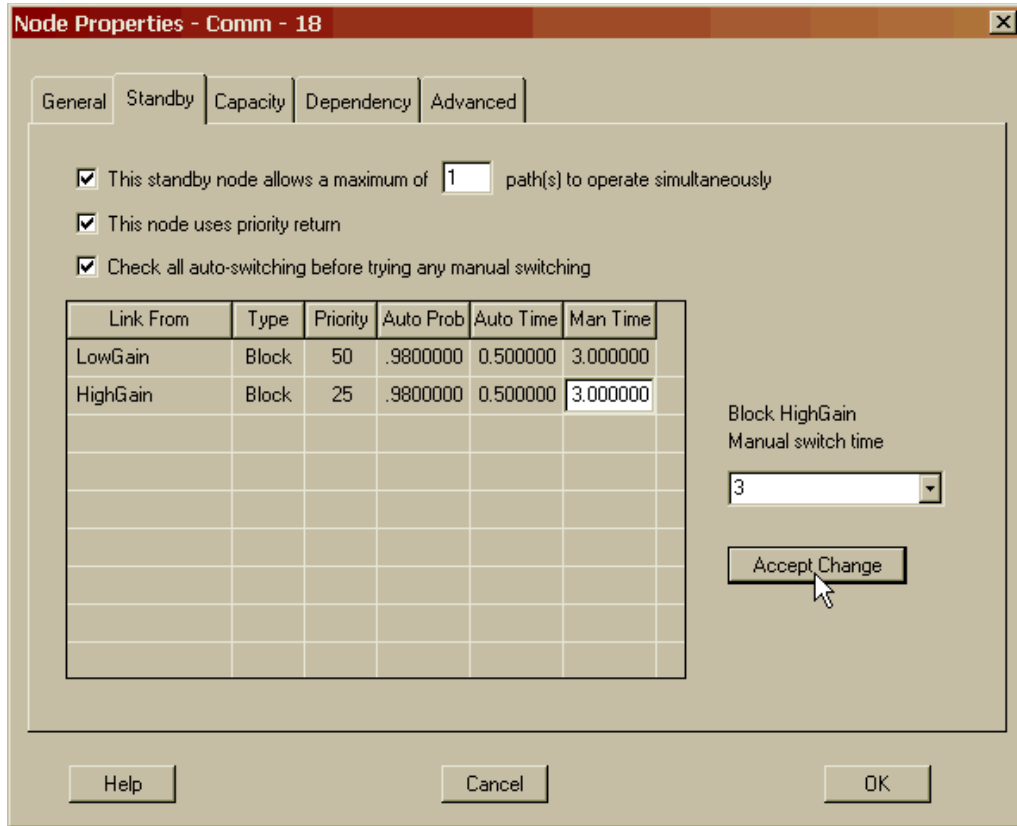


Figure 8.4 Changing a Node's Switching Information

The standby stress for a component defaults to zero and as mentioned earlier represents a cold standby condition. We will leave the high gain antenna component to be a cold standby component since it will never act as an alternate given its priority value. Let us modify the low gain antenna component's stress from zero to a value of five percent. This implies that the low gain antenna component, while acting as an alternate, will only acquire life at five percent of its normal operating rate. Thus, the low gain antenna component has been changed from a cold standby alternate to a warm standby alternate. Standby stress is unique to each component and is thus specified within a component's Block Properties dialog box. Open the low gain antenna component's Block Properties dialog box and select the Advanced tab. Modify the Standby stress value box as shown in Figure 8.5.

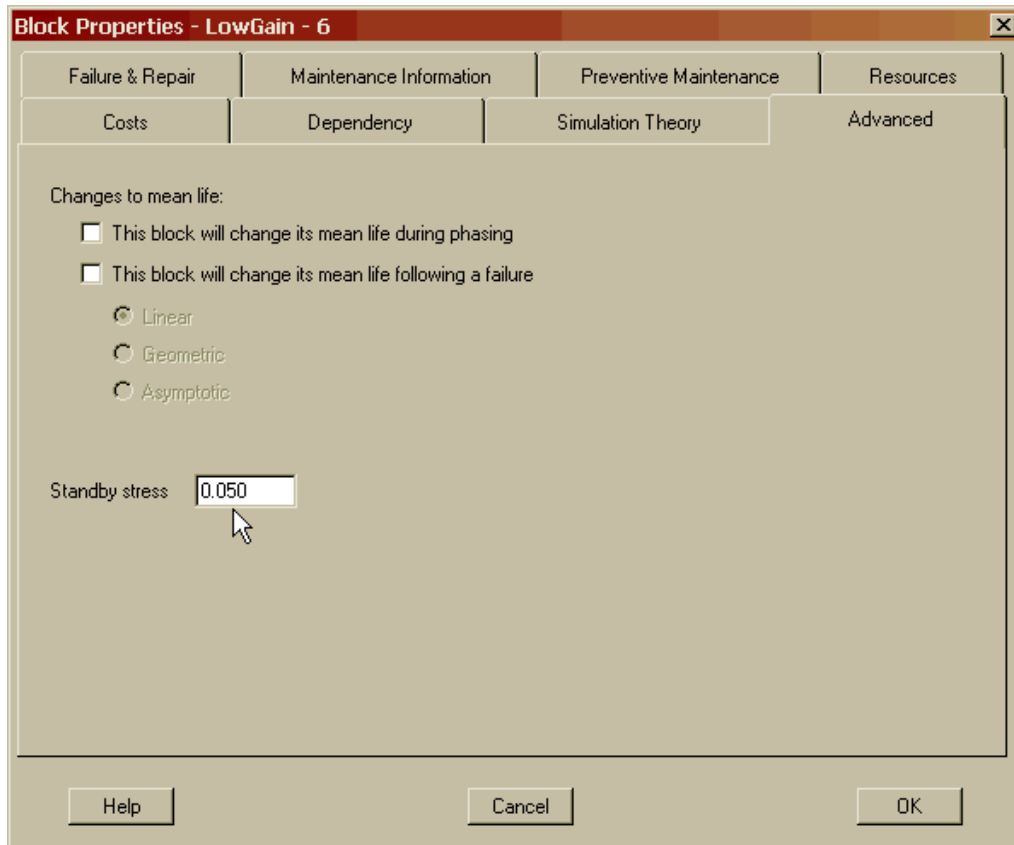


Figure 8.5 Changing a Component's Standby Stress Value

Open the Simulation Options dialog box and conduct a simulation that starts in step mode. Notice that the effect of the low gain antenna standby component is immediately apparent, as Figure 8.6 indicates.

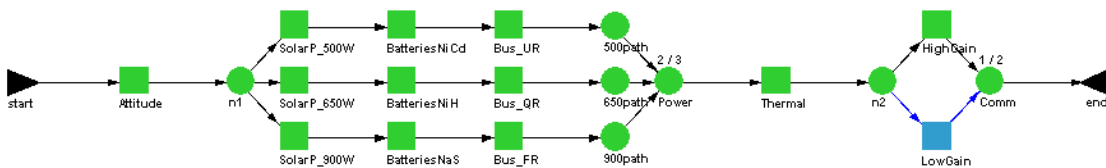


Figure 8.6 Standby RBD at Simulation Time Zero

The low gain antenna component is now a medium blue color at the start of the simulation. This indicates that it is capable of operating but is not because it is in a standby state. Recall from Figure 8.1, the medium blue color indicates that the low gain antenna component has been designated as a warm standby component. Jump to the sixth trial and then take seven steps such that the high gain antenna fails as shown in Figure 8.7.

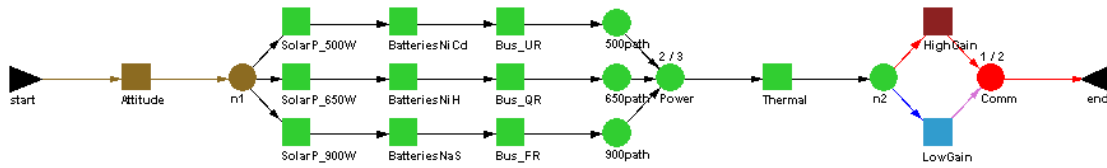


Figure 8.7 Switchover in Progress

A switchover is in progress as indicated by the light purple colored link in Figure 8.7. The system is now down, since neither the high gain nor the low gain antenna component is operating. If auto-switching with a zero switchover time was implemented, the switchover would have occurred immediately without a system downing event occurring. When the switchover is completed (another step), the low gain block becomes the primary and the system comes back up to a yellow or degraded status, which is shown in Figure 8.8.

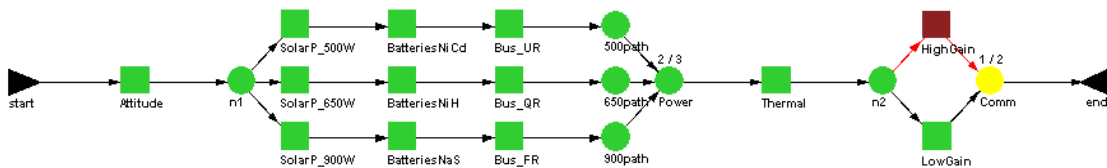


Figure 8.8 Switchover Completed

It is interesting to note that it took a half hour to switch over to the low gain antenna component. This means that the automatic switchover probability was successful since, had it not succeeded, the switchover time would have been three hours as indicated in Figure 8.4. When the failed high gain component is repaired (i.e., six additional steps), it returns to its role as the primary communications antenna and the low gain component returns to its role as an alternate in a warm standby status (Figure 8.9). Click on the depressed Pause button and allow the simulation to run until completion and note the results in Figure 8.10.

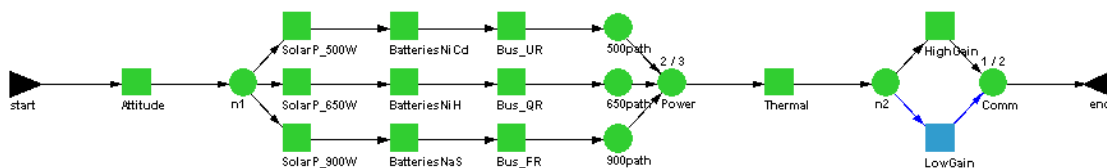


Figure 8.9 HighGain Component Regains its Role as Primary

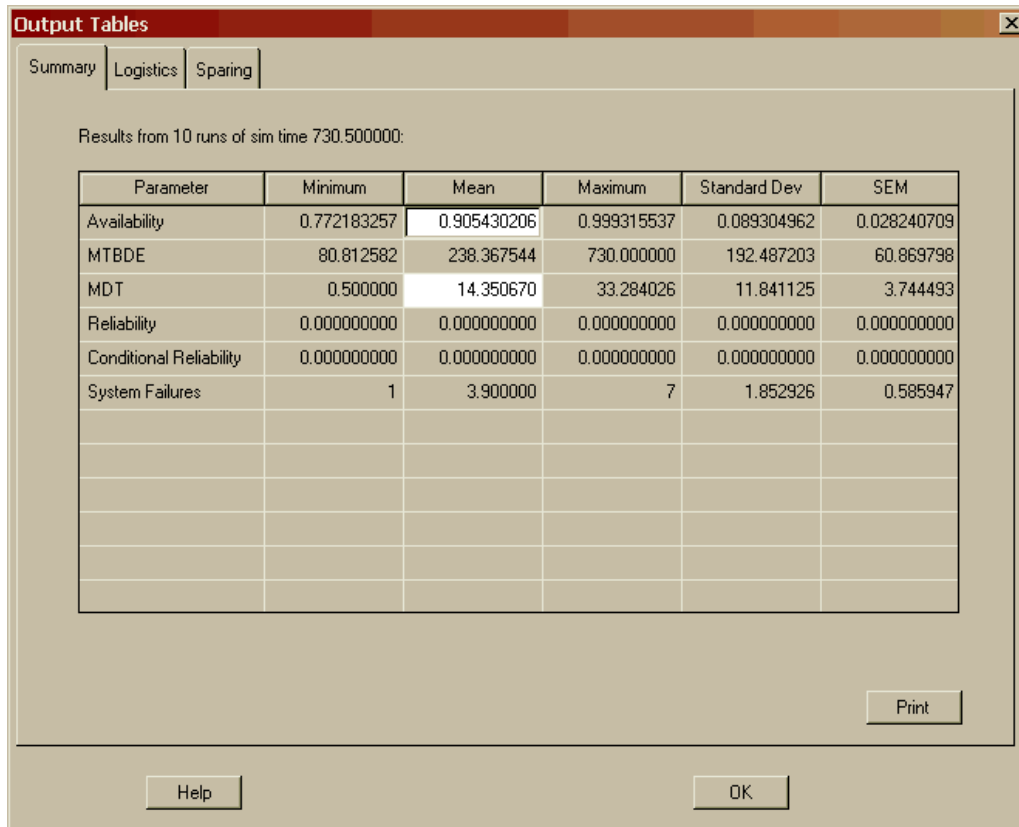


Figure 8.10 Summary Results for Chapter 8

You cannot directly compare these results to those in Chapter 7 since the addition of standby relationships dramatically alters this model. We can say, however, that the results shown in Figure 8.10 (i.e., an availability of 0.905430206 and a mean down time of 14.350670 hours) are more accurate since this model contains more fidelity with respect to operational redundant realism.

The standby feature can be used along with the dependence features presented in the last chapter to create strings or subsets of components that collectively act as a standby structure. Open **StandbyDep1.rbd** and notice that this RBD appears to be identical to the preceding model. The only difference is that the Power node was modified to be a standby node. Recall that internal dependencies within the power strings were defined in the previous chapter (i.e., elements within the strings are tied to the node at the end of a string). Open the Power node's Node Properties dialog box and note how the standby attributes were applied (see Figure 8.11). Since there is no priority distinction among paths into the Power standby node, Raptor defaults to the order in which the components were linked to the standby node as they were placed within the Workspace View.

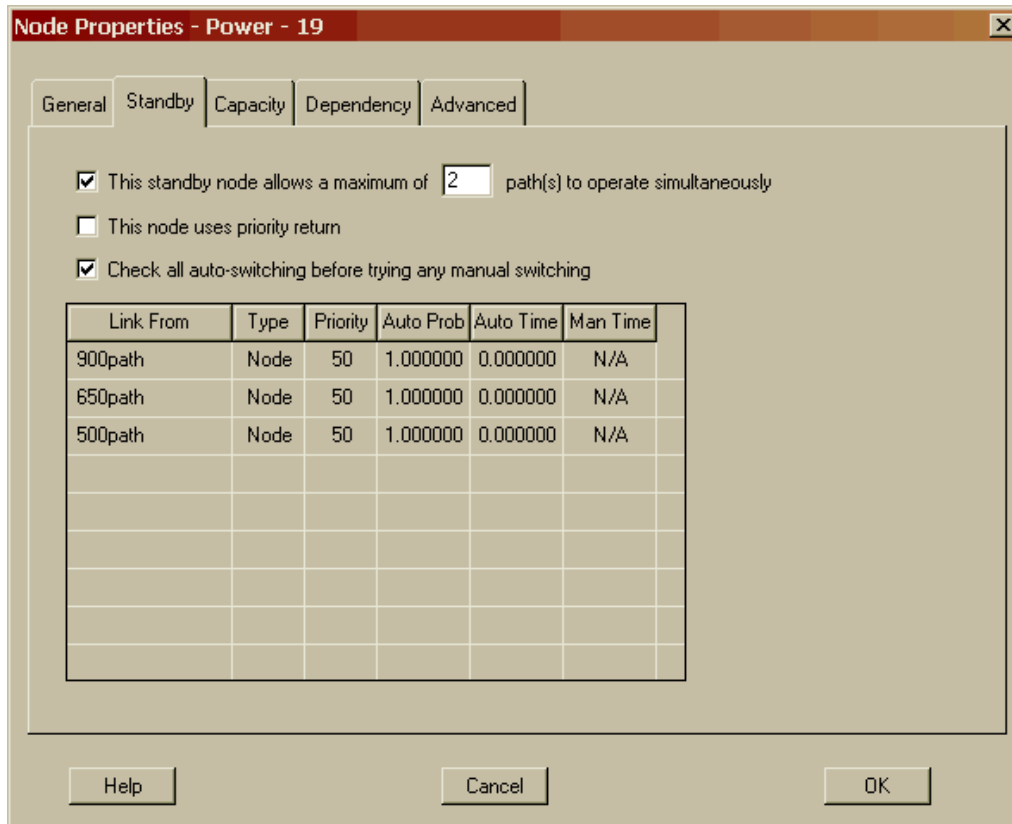


Figure 8.11 Power Node's Standby Attributes

Open the Simulation Options dialog box and conduct a simulation in step mode. For this model, of the three paths linking into the Power node, the link from the 500path node was the last link placed within the Workspace View. Thus, it is treated as the lowest priority as indicated in Figure 8.12. Notice that the 500path power string is entirely in a cold standby status while the low gain antenna component is in a warm standby status.

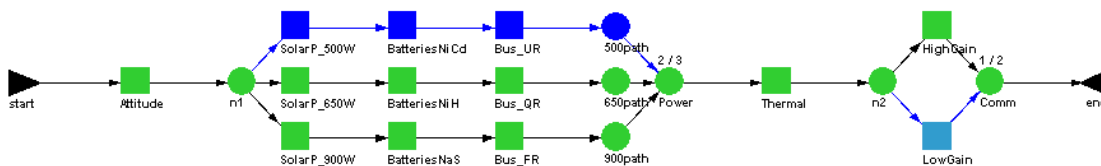


Figure 8.12 Standby and Dependence Features Combined

Step four times and note that the 650path power string is incapable of functioning because of triggered preventive maintenance as shown Figure 8.13. In this case, the switchover times have been specified to be zero, so the switchover to the top power string is immediate and no system down time is incurred. Notice that the Power node changes to a degraded status, but still meets its requirement of 2-out-of-3 paths operating.

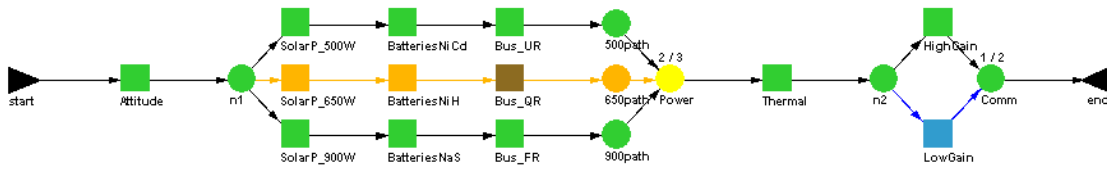


Figure 8.13 Immediate Switchover

Step twice more and notice that the standby string is now the middle string (Figure 8.14). This is because the priority return option was not checked within the Power node's standby tab (Figure 8.11). Click the depressed Pause button and allow the simulation to run and notice that the string that fails always becomes the standby string. This model construct ensures equal wear on the three power strings.

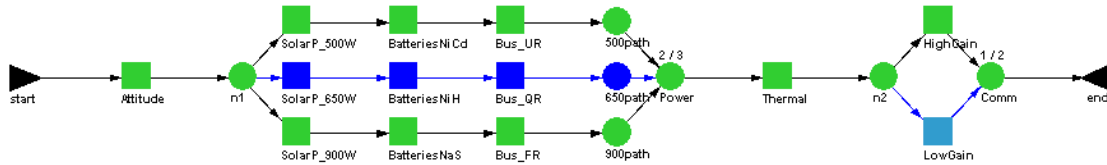


Figure 8.14 Non-Priority Return

This group standby structure is possible because of the interaction of the standby and dependence features of Raptor. Without dependence, standby will only affect a component immediately upstream (i.e., to the left as dictated by a left to right reading bias) of a standby node. Figure 8.15 is an example of how standby would appear within a string where dependence is not implemented. The remaining components in the string are unaffected by the Power node's standby directions.

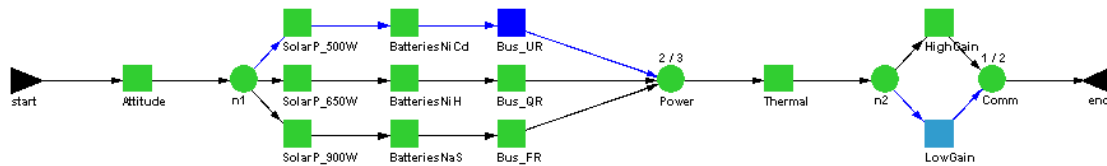


Figure 8.15 Cold Standby Without Dependence

• • •

One feature that was not used in the examples of this chapter is the option to check all auto-switching before trying any manual switching. When several components are on standby and the auto-switch fails for the highest priority standby, that standby will begin manual switching. If this option is selected, the standby node will check other standby components to see if an auto-switch can be conducted with one of the other standby components.

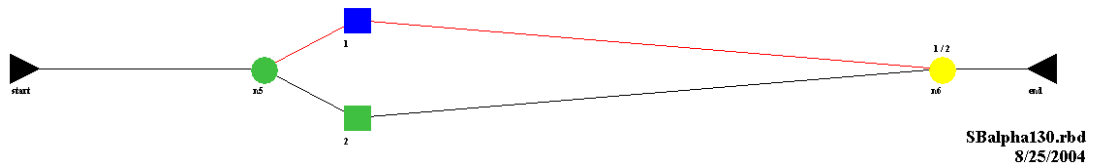
• • •

Readers are cautioned that it is possible to construct RBDs that have several standby nodes that together provide conflicting control over an element. If this occurs, an element will flash between two states in an infinite loop. This scenario can only occur when an element is given more than one standby master. While no man can serve two masters well, neither can a block or node. Hence, we strongly encourage users not to model components with two or more standby masters.

Problems

1. Give an example of a system that has standby components.
2. What types of redundant components typically use priority return?
3. What is the effect of designating a node to be a standby node if all inbound paths to that node are from nodes, and those upstream nodes have no dependent elements?
4. System A and system B are each comprised of the same type of components. The components have a failure and repair distribution that is exponentially distributed with a mean of 100. System A consists of two components in a 1-out-of-2 standby configuration. System B consists of four components in a 2-out-of-4 standby configuration. Both systems switch perfectly and instantaneously. Which system has the greater availability?
5. System C and system D each consist of two blocks in a 1-out-of-2 standby configuration. The only difference is that one system switches automatically in zero time and the other auto-switches in 0.0001 time units. How does this affect the MTBDE and availability of the two systems?

6. Can a passive node (one that has one inbound link and one outbound link) be designated as a standby node?
7. Name a system that would likely possess a hot standby status.
8. What is the effect of placing events within a standby structure?
9. Why is the node in the following diagram yellow?



Chapter 9



Objectives

1. Understand how to generate availability plots.
2. Understand the different types of availability plots.
3. Understand the causes of startup transients.
4. Implement the delayed statistics gathering method of controlling transients.
5. Implement the random start-up method of controlling transients.

Recommended Problems

2, 5, 6, 7, 8, 9, 10, 11, 13, 14, 15, 16 and 17

Transients

The simulations conducted thus far started with every element in an up or what is considered a good state. If a model contains logistics, then at simulation time zero all stocks are full, all queues are empty, and there are no components in their repair cycles. Stated another way, a Raptor simulation will by default start with brand new components, fully stocked spares and complete availability of all resources. Starting a simulation with new components causes the initial portion of a simulation to be over represented, particularly for short-length trials. With respect to simulation theory, this initial over representation is known as “startup transients”. This chapter provides the reader with an understanding of Raptor's embedded startup transients, and when it is advantageous to do so, how to manage them. Startup transients should not be viewed as either good or bad but a consequence of simulation theory that must be consciously addressed and then managed.

Raptor allows transients to be controlled by implementing one of two methods, both of which will be discussed in this chapter. The first method, “delayed statistics gathering”, is an effective technique to managing startup transients associated with RBDs that contain logistical concerns. The second technique, known as a “random startup” simulation is an efficient method of managing startup transients for RBDs that are not dominated by logistical constraints. Additionally, the latter technique allows users to start a model in an unusually weakened condition and then be able to determine the mean time for the system to recover to a viable state.

Delayed Statistics Gathering Method

Open **Transients1.rbd** and note the similarity of this RBD to those used in the previous chapters. Open the Simulation Options dialog box and set the simulation parameters as shown in Figure 9.1. Be sure to set the time-truncated field to a value of 8,000 hours. The choice of 8,000 hours was merely a guess that we know is several times greater than the simulation length of interest (i.e., a length of 730.5 hours as defined in the earlier chapters). The 8,000 hour parameter is believed to be a point in time where we hope that the effects of the startup transients will no longer be dominating and hence overly biasing the results of this model.

The 8,000 hour parameter is often called the “steady-state point” for a particular system and its associated simulation criteria. While the Raptor tool cannot explicitly define the steady-state point for a given model, Raptor does provide a tool that can determine this point by analysis and iteration. Click on the Advanced tab of the Simulation Options dialog box and check “Display A_0 plot during simulation”, which is shown in Figure 9.2. Ensure that you set the A_0 plot rate to fifty from its default value of ten.

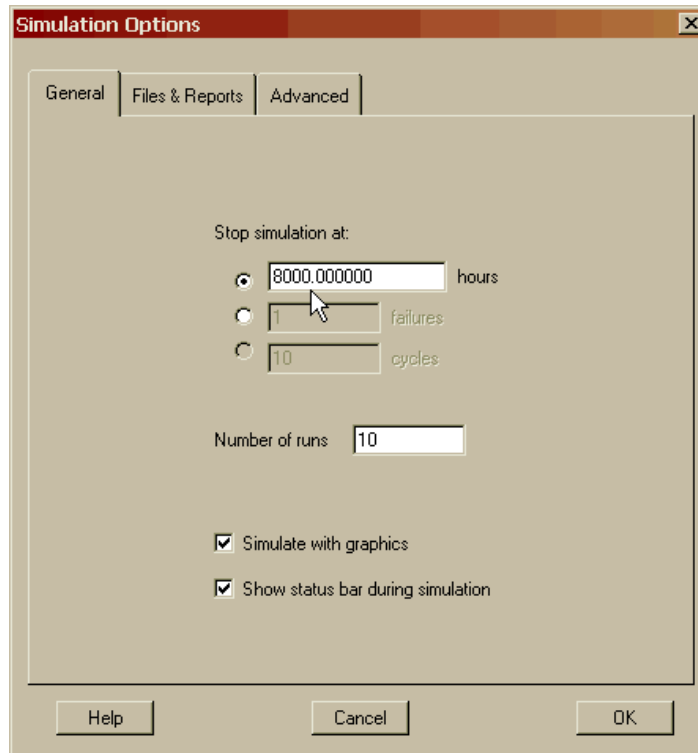


Figure 9.1 Simulation Options for Chapter 9

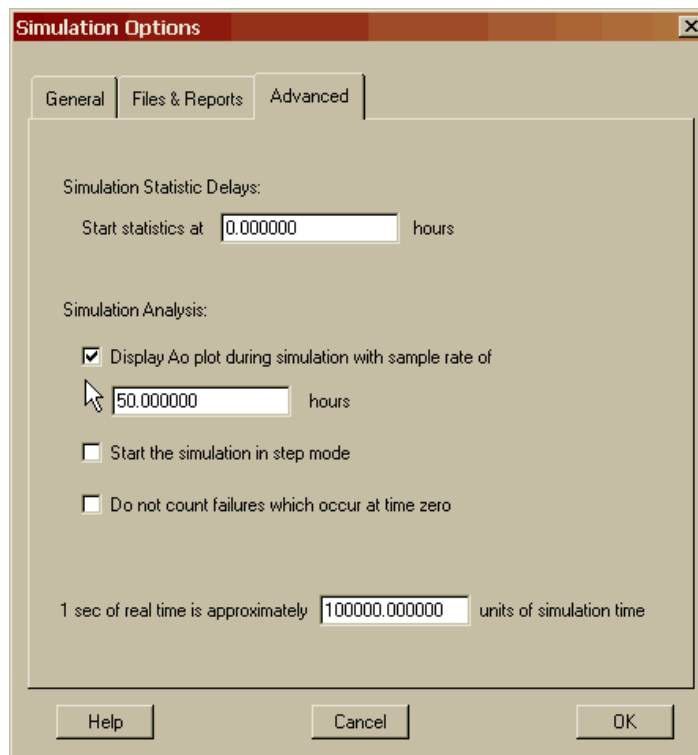


Figure 9.2 Advanced Tab of Simulation Options Dialog Box

The A_o plot sample rate value box specifies how often a data point will be placed on a graph displayed within the Simulation View. This value only modifies the density of plotted points and not the overall results of the simulation or its graphs. Since we are conducting an 8,000 hour simulation and we have modified the sample rate to a value of fifty, 160 data points (i.e., 8000/50) will be plotted for each availability graph displayed in the Simulation View. Raptor limits the number of plotted points to 1,000 for each displayed graph. The reader is cautioned to only plot enough points to acquire an appreciation of a system’s steady-state availability behavior. Plotting multiple 1,000-point availability graphs and simulating a large RBD can consume an enormous portion of a computer’s memory and thus unnecessarily slow the process of acquiring meaningful results.

Begin the simulation by clicking the OK button and notice how the Simulation View has been augmented with a graph window as shown in Figure 9.3.

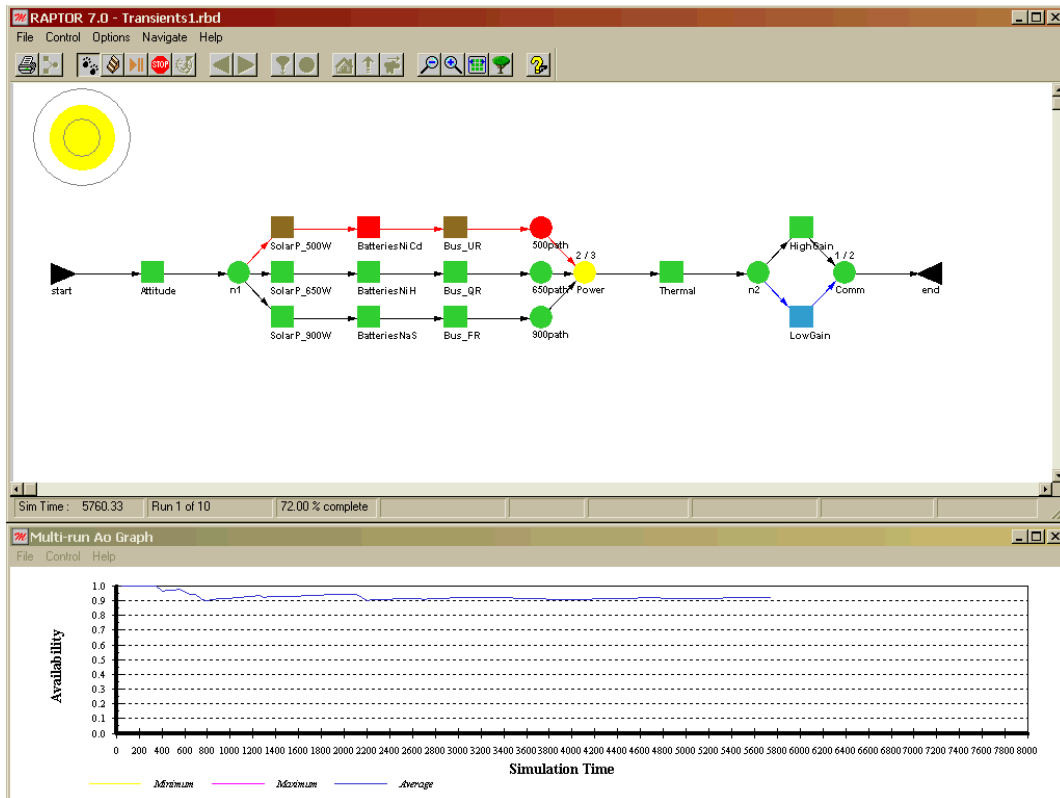


Figure 9.3 Simulation View with Availability Plots

The graph that first appears in the plot window is the “Multi-run A_o Graph”, which displays the availability of a system at any given time. For this model, all availability graphs are plotted every fifty hours since that is the value set in the A_o plot sample rate field. From the *Control* menu of this graph window change the A_o plot from the multi-run to the per trial graph by selecting the View Per Trial A_o menu item, as shown in Figure 9.4.

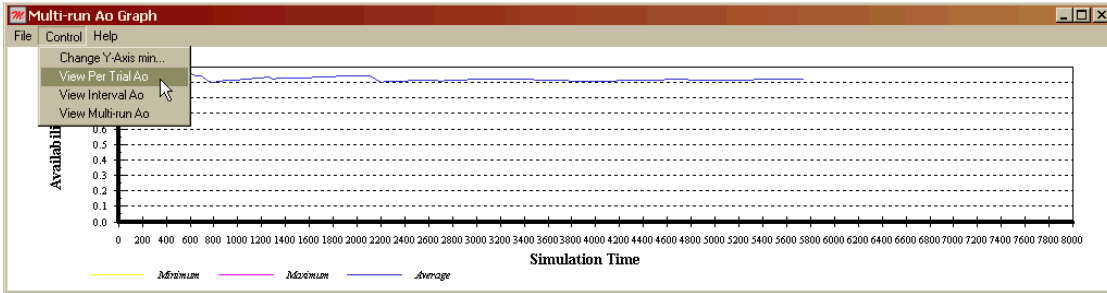


Figure 9.4 Changing the A_o plot to Per Trial

As Figure 9.5 indicates, the Per Trial A_o plot displays the percentage of time a system is in its green, yellow and red statuses. The top of the yellow region is by definition the availability of the system at any given time since a system's availability is the sum of the times it is in the green and yellow states (see Appendix F for the definition of availability as well as all other parameters used in Raptor).

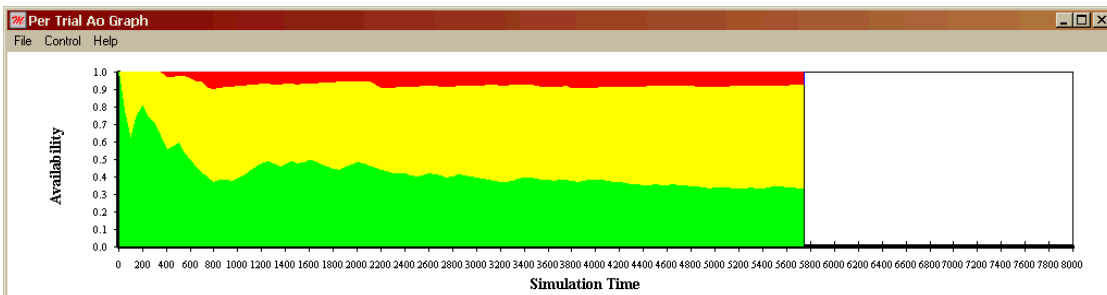


Figure 9.5 Per Trial A_o Plot

From the *Control* menu item of this graph window change the A_o plot from the per trial to the interval graph by selecting the View Interval Trial A_o menu item. This availability graph is shown in Figure 9.6.

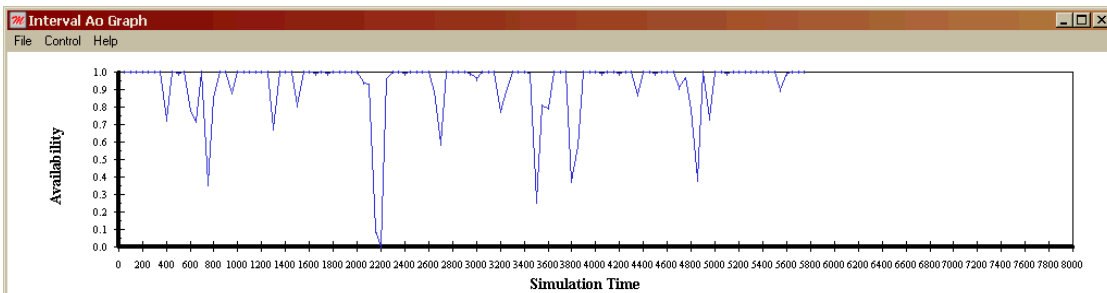


Figure 9.6 Interval A_o Plot

The interval A_o graph window plots the availability of a system based on the sample rate set within the Advanced tab of the Simulation Options dialog box. For this simulation, the sample rate was fifty; thus, the availability for every successive fifty time units is plotted. This plot is pertinent when a system is suspected of possessing constraints that cause its availability to be cyclic in nature. This type of plot is often used in conjunction with phased RBDs that will be discussed in Chapter 17.

Return to the multi-run A_o plot and allow the simulation to continue as shown in Figure 9.7.

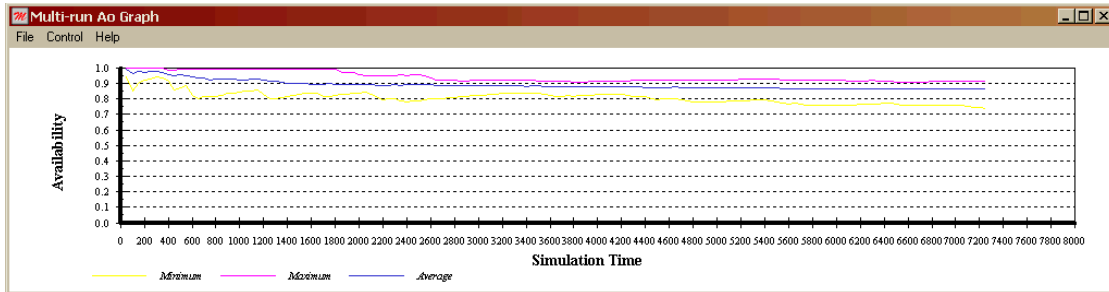


Figure 9.7 Multi-run Ao Plot

The multi-run A_o graph displays the maximum, average and minimum A_o values observed across all trials. It is important to understand that the lines for the maximum and minimum are not running averages, but absolutes. They represent bounds for which the availability of a system over time never exceeds nor is ever less than. A user should view the maximum and minimum lines as system availability extremes plotted against simulation time. The line representing the average availability (i.e., the blue line) is a running average of the availability of the system at any given point in time up to the termination point of 8,000 hours.

You can zoom into any graph via the graph window's *Control – Change Y-Axis Min* menu item. This menu option brings up the Set Y-Axis Min dialog box that allows modifications to the minimum y-axis value. Within the Set Y-Axis Min dialog box, change the minimum value from 0.0 to 0.6 as shown in Figure 9.8.

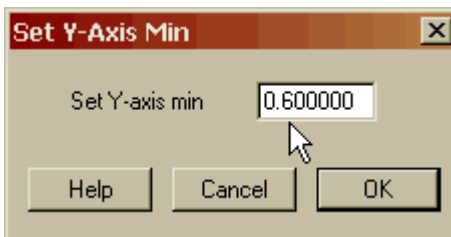


Figure 9.8 Set Y-Axis Minimum Dialog Box

Figure 9.9 displays the effects of changing the minimum y-axis value on the multi-run A_o graph.

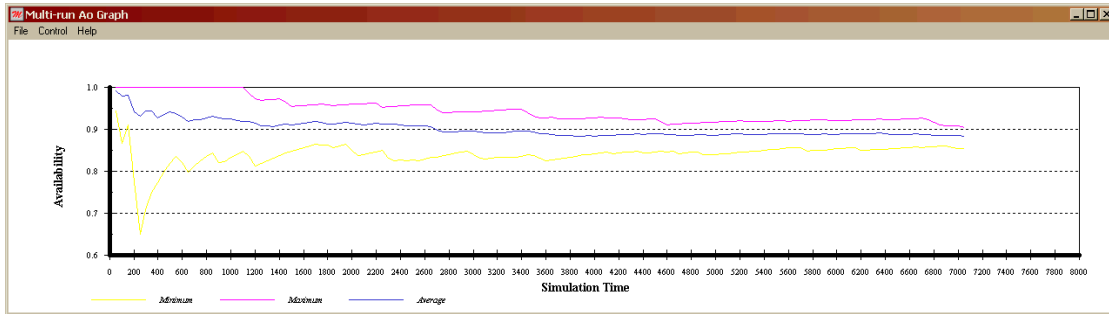


Figure 9.9 Effects of Changing Y-Axis Minimum on the Multi-run A_o Graph

An examination of the multi-run A_o graph reveals an interesting aspect concerning the startup transients of this system. As shown in Figure 9.9, the availability of the system is relatively unchanged after 6,000 hours. When the average availability line of the multi-run A_o graph begins to stabilize or level out, a steady-state condition has been reached and thus the effects of the startup transients have been overcome. Let us resolve that the average availability line is level enough such that we can conclude that the effects of the system starting in an initially good state have been overcome after 6,000 hours of simulation time. The choice of the steady-state point is a judgment call that will likely vary among analysts. When determining the steady-state point, you should always err on the conservative side. If a debate ensues that a greater time is needed to achieve steady-state, then simulate the longer time to ensure that steady-state is achieved. The cost for choosing a longer steady-state point is that more time will be needed to obtain meaningful results; the reward is that meaningful results will be obtained.

Raptor allows for the discarding of initial, and often biased, results by specifying a delayed statistics gathering parameter. This parameter instructs Raptor to discard all results up to that point in the simulation for each trial. Thus, only the last portion of a trial's results is incorporated into the calculation of the final results across all trials.

For the purpose of this chapter, we have quantified the effects of the startup transients, and we will now attempt to manage these effects so that we can obtain results that are more representative of the steady-state operations of this system. Open the Simulation Options dialog box and change the simulation termination time to 6,730.5 hours on the General tab. Select the Advanced tab and set the "Start statistics" value box to 6,000 hours as shown in Figure 9.10. Simulate this RBD by selecting the OK button. The reason for using what seems to be an unusual simulation length of 6,730.5 hours is that we wish to compare the results of this simulation with those of Chapter 8. Recall that in Chapter 8 we conducted ten trials of 730.5 hours and achieved results that we called more accurate representations of the true RAM characteristics of this system. From the steady-state analysis of Figure 9.9, we determined that we needed to eliminate the first 6,000 hours from the simulation statistics to overcome the effects of

the startup transients. Therefore, by simulating for 6,730.5 hours (i.e., the sum of 6,000 and 730.5), we can compare the results of this chapter with those of Chapter 8.

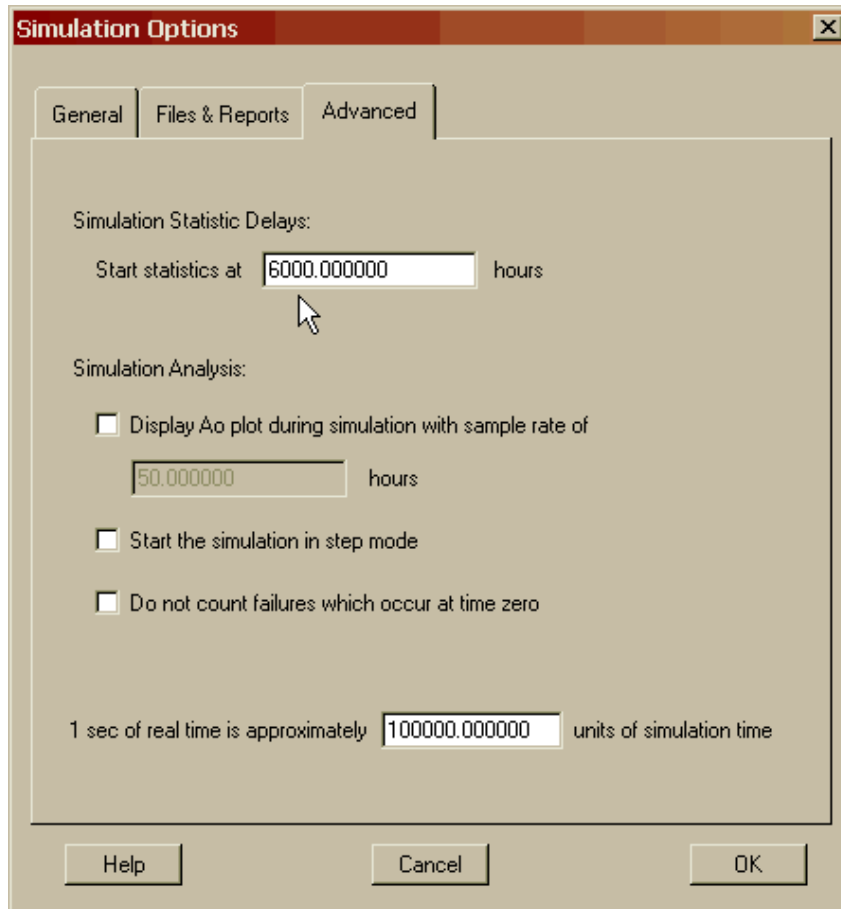


Figure 9.10 Setting the Start Statistics Value

Figures 9.11 and 9.12 display the results from this chapter without the startup transients and those from Chapter 8 (reference Figure 8.10), which contained the transients, respectively. The results that we thought were good estimators in Chapter 8 turned out to be less robust than we had anticipated. The steady-state availability of this system is much closer to 0.8596 than to 0.9054. The steady-state MDT value of about 20.41 hours is greater than twenty-nine percent more than our estimate in Chapter 8. These results should demonstrate that this system is strongly affected by its startup transients. Had you based your decisions on the results of Chapter 8, you would have been disappointed with your system's steady-state performance. If you are interested in the first month of operation, the results of Chapter 8 are more accurate than those of this chapter. Both simulations can be considered correct depending on the type of results desired (i.e., start-up or steady-state results).

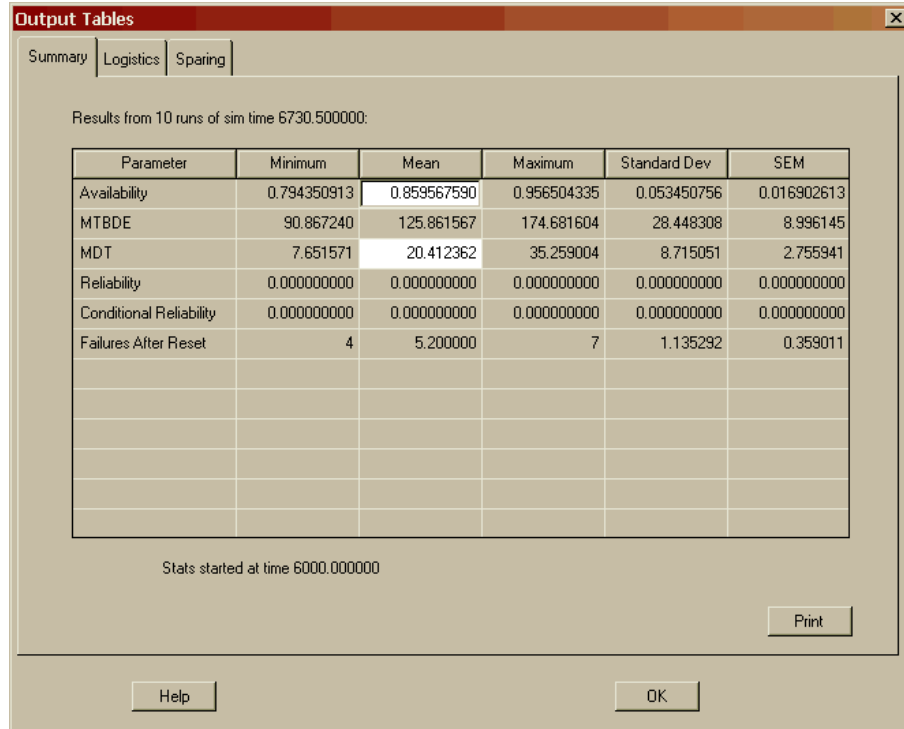


Figure 9.11 Results With Startup Transients Removed Using Delayed Statistics Gathering

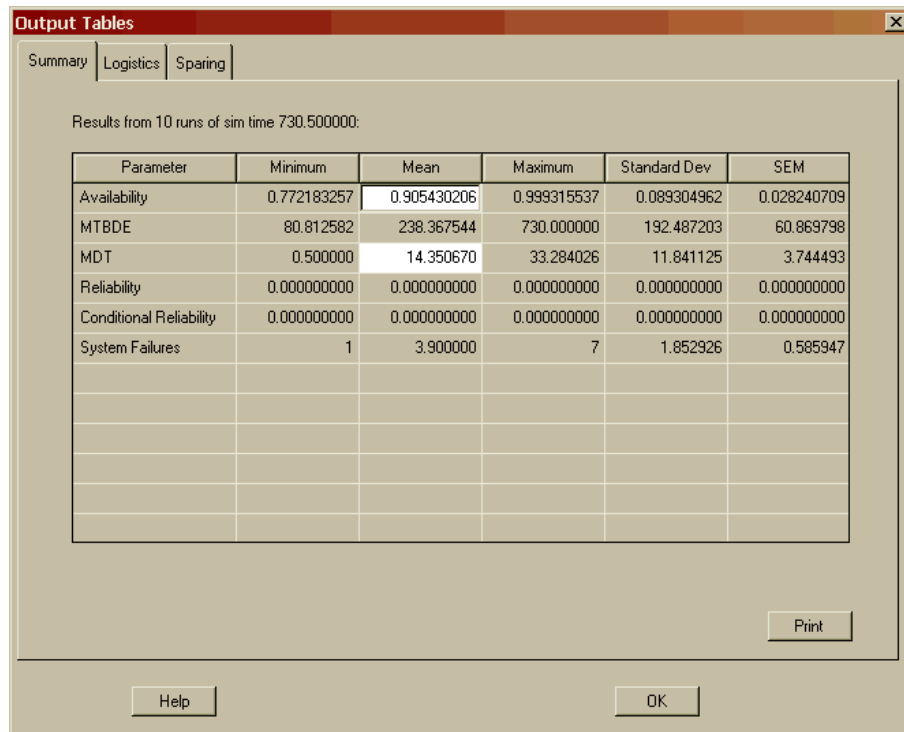


Figure 9.12 Results With Startup Transients (Chapter 8)

Random Startup Method

A second method for managing startup transients is to start each component of an RBD in a more representative or random state. Raptor allows a component to be initiated in six different manners or what are known as startup conditions. Let us review each startup condition by opening **Transients3.rbd**. Open the Block Properties dialog box for the low gain antenna and click on the Simulation Theory tab as shown in Figure 9.13. The area at the bottom of the dialog box labeled “This block begins the simulation” controls the startup conditions of a component.

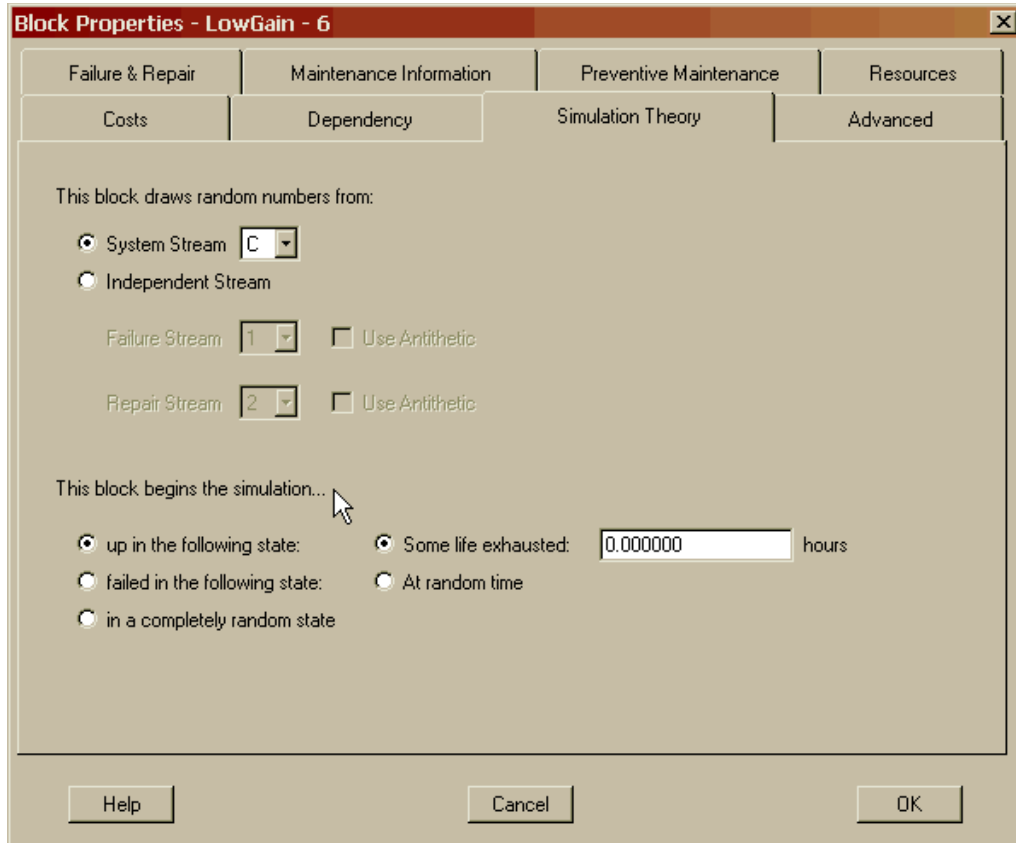


Figure 9.13 Default Startup Simulation Settings

Figure 9.13 displays the default startup condition for a component (unless modified by the block default feature). The current startup condition is up with no life exhausted. This declares that the low gain antenna component under these conditions is assumed to be brand new at the start of each trial. Thus, this type of component will start each trial in a good state or be displayed as green or blue within the Simulation View.

Click the "At random time" radio button as shown in Figure 9.14. A component with this startup condition would start a simulation trial in a good state but some random amount of life will already have been exhausted. Raptor will draw a uniform real random number from zero to one at the beginning of a simulation trial for each component with this condition and exhaust that percentage of time from its first

random mean life. For example, a component with a mean life of 100 hours that is normally distributed could obtain 120 hours as its first life. If its life exhausted random number was 0.75 for the first trial, then 90 hours (i.e., 0.75 x 120 hours) would have been exhausted before the trial begins. This component will fail during the first simulation trial after 30 hours of operation.

For each trial, a different random number is drawn and thus a different amount of life is randomly exhausted. Contrast this startup condition with the previous condition that allowed the same fixed and therefore non-random amount of time to be exhausted from each trial. This feature is used when it is known that a component possesses some amount of life exhausted but this information has not been recorded, so the specific value is unknown. For example, if you purchased a set of used tires you might not know how much life is exhausted (i.e., miles) on each tire. It might be prudent to model each tire as operating with some unknown amount of life exhausted.

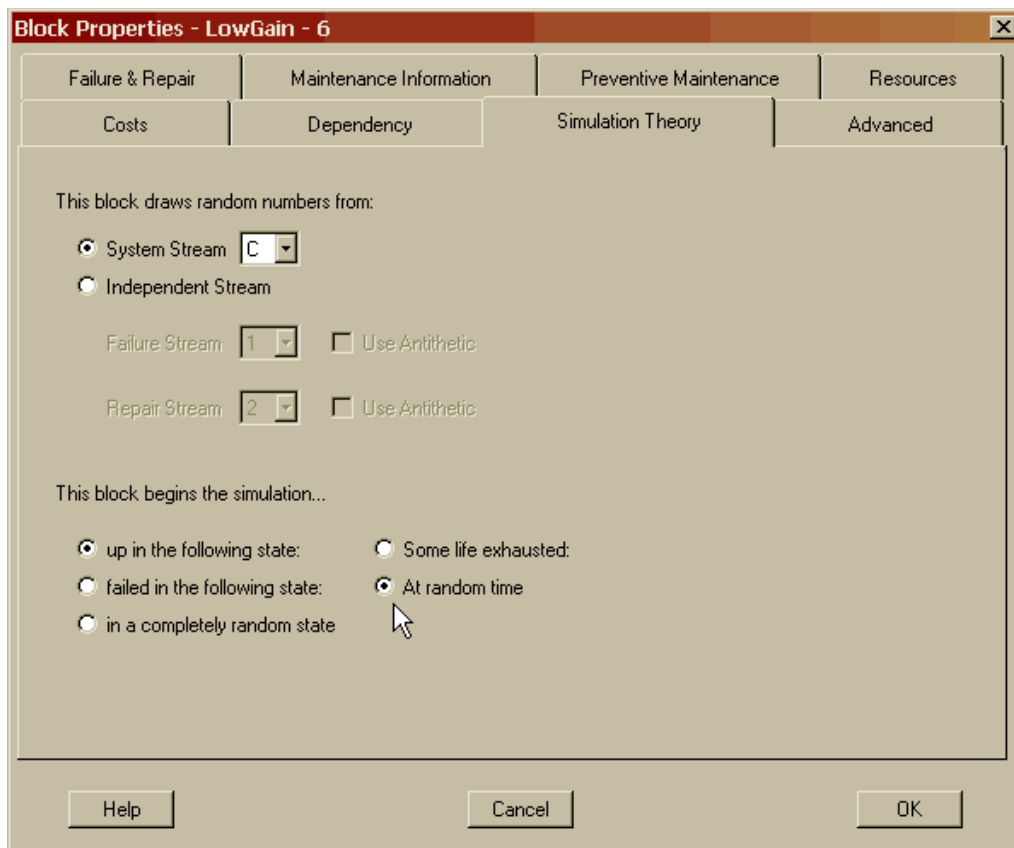


Figure 9.14 Start Up With Some Random Life Exhausted

Change the radio button to appear as shown in Figure 9.15. This startup condition instructs Raptor at the start of each simulation trial to begin this component in a failed state at the beginning of its repair cycle. Recall the repair cycle is comprised of logistics delays, waiting for spares and resources and the actual hands-on maintenance of the repair. Thus, a component with this condition essentially fails at the simulation time zero and initiates its repair cycle. If the percent complete value box is non-zero,

then that percentage of repair time would have already been completed. For example, suppose a component has a basic repair cycle length of ten hours (this includes logistical delays and hands-on maintenance but does not include delays due to spares or resources) and 40 percent was entered into the percent complete value box. Then this component would start failed with six hours (e.g., $10 - 10 \times 0.40$) remaining before it is fixed (assuming resources and spares were available).

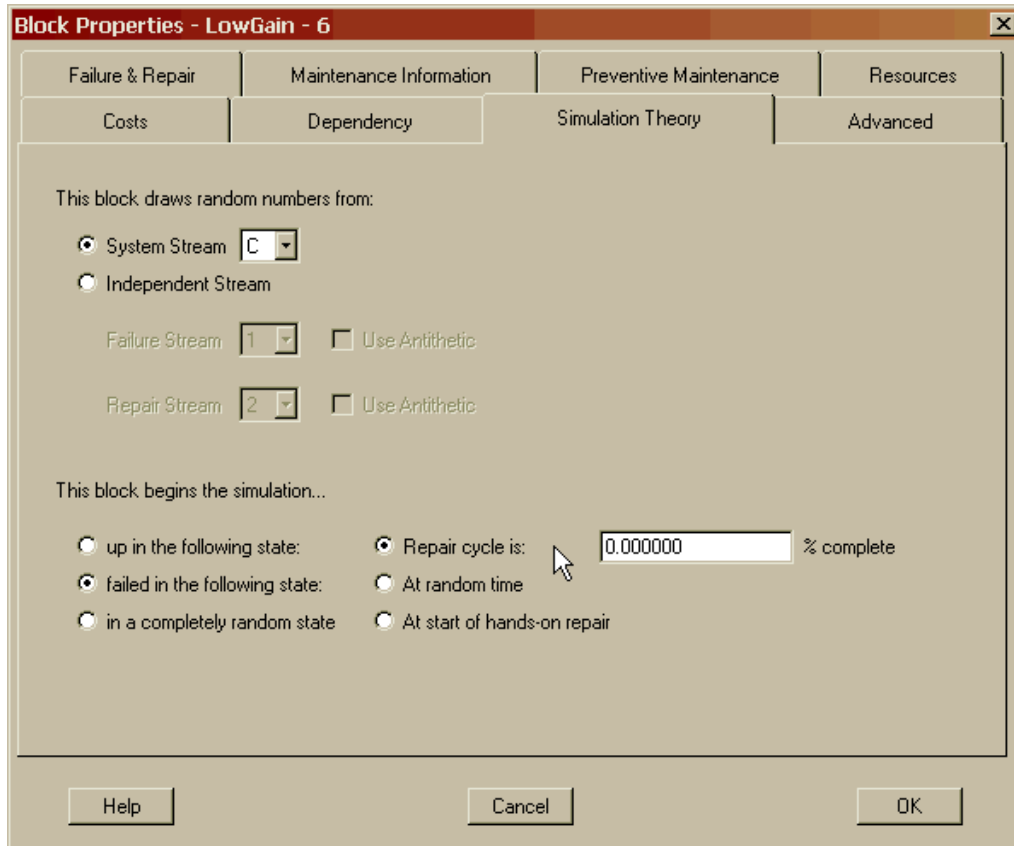


Figure 9.15 Start Failed With Repair Cycle Zero Percent Complete

Contrast this with the other two conditions for starting a component failed. First, a component can start a simulation trial failed but with some random amount of its repair cycle completed (Figure 9.16). Recall that the previous condition allows for the same fixed and therefore non-random amount of a component's repair cycle to be completed at the start of each trial. The other failed condition is shown in Figure 9.17, which indicates a component starts a simulation trial failed at the beginning of its hands-on maintenance point of its repair cycle as long as spares and resources are available.

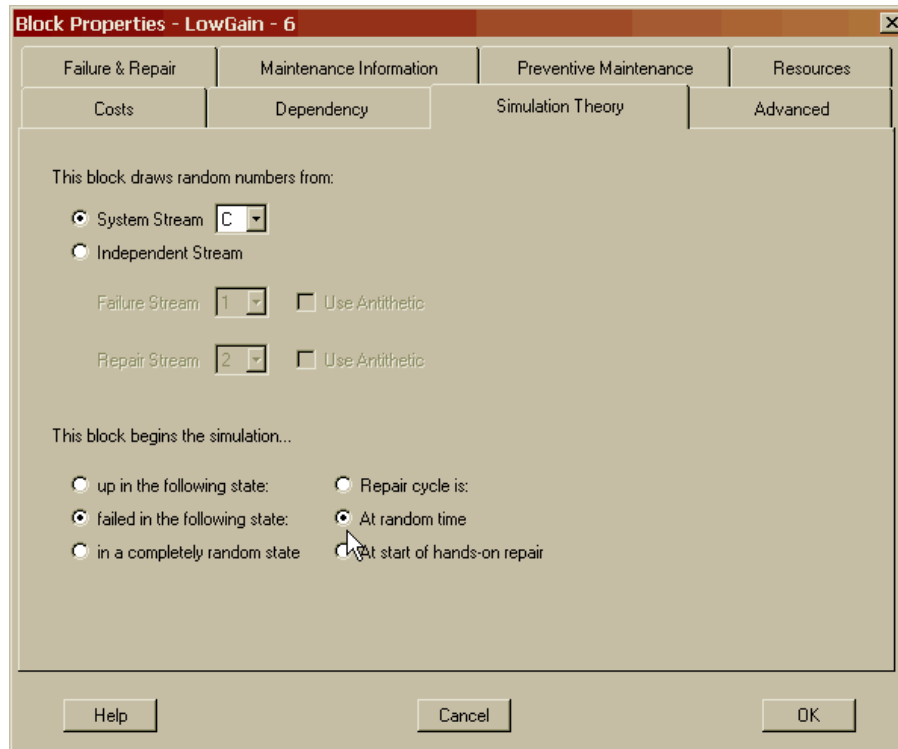


Figure 9.16 Start Failed With Random Amount of Repair Cycle Complete

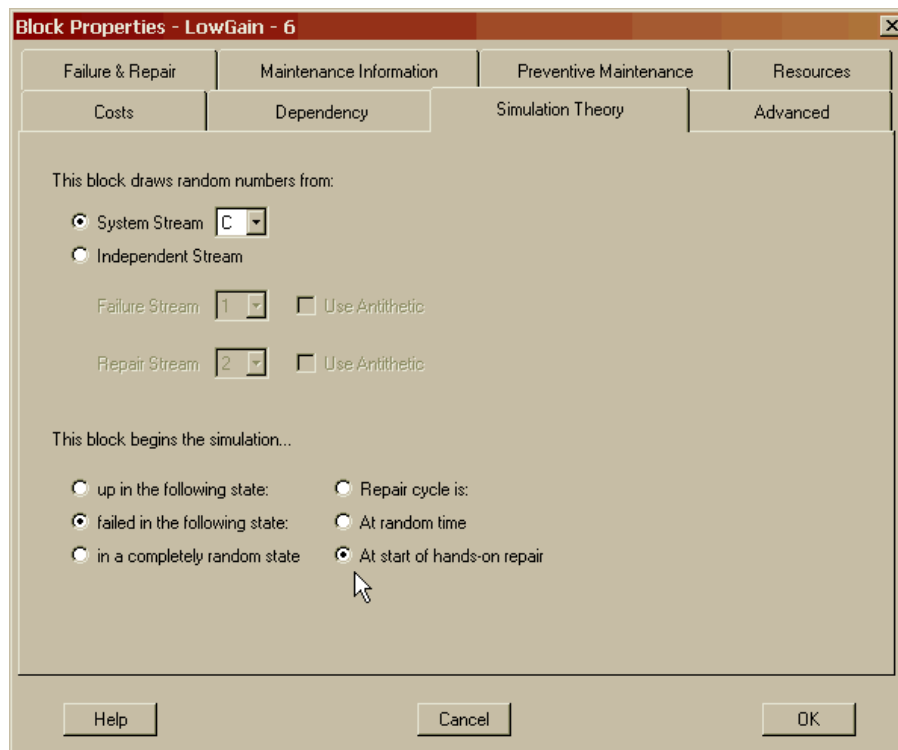


Figure 9.17 Start Failed at Hands-on Maintenance Point of Repair Cycle

Change the radio button to appear as shown in Figure 9.18 to view the last of six startup conditions for a component.

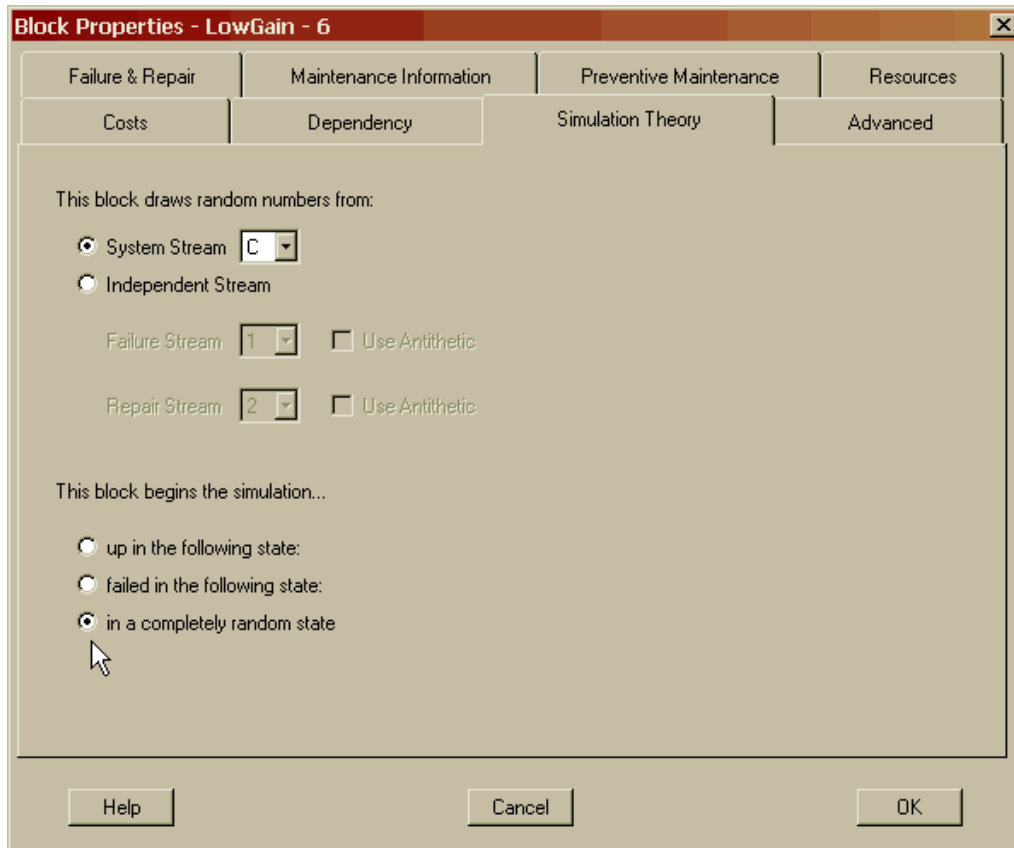


Figure 9.18 Starting in a Completely Random State

This startup condition allows a component at the start of each trial to possess a probability of starting in a failed state or an operating state with some random amount of life exhausted. The probability that a component starts a trial in an up or down status is directly related to a component's operational availability without regards to dependency, preventive maintenance, resource and sparing delay issues.

Maintain this startup condition for the low gain antenna component by clicking the OK button. Open the Block Input Tables dialog box and select the Advanced tab. As Figure 9.19 shows, all the other components have had their startup conditions modified to be completely random similar to the low gain antenna component. The impact of starting all components in a completely random state is to start each trial at some random point in time of a system's operations. Thus, the system's startup transients are managed by starting all components in a jumbled or random condition. This jumbled condition ensures that the startup transients are nonexistent. This type of simulation would not represent a new system very well but one that has already operated for some time and is likely to have reached its steady-state condition.

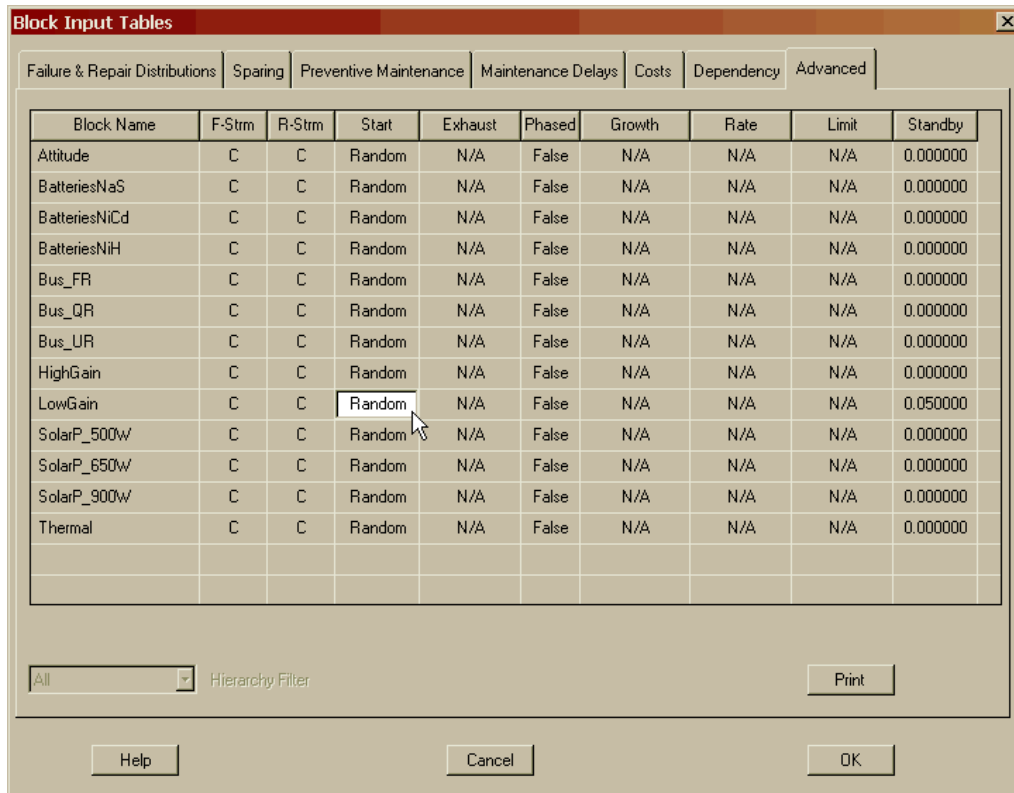


Figure 9.19 Advanced Tab of the Block Input Tables Dialog Box

Start a simulation in step mode and notice that the fully-regulated bus and sodium-sulfate battery components have failed (see Figure 9.20) at a simulation time of zero.

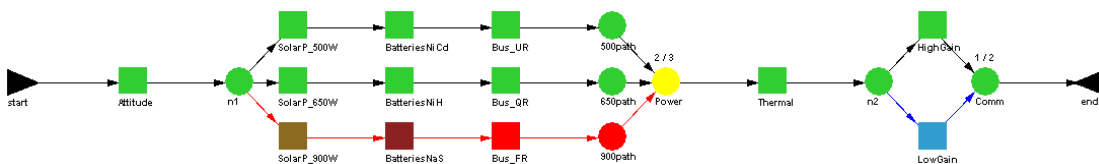


Figure 9.20 Failure at Simulation Time Zero for the First Trial

Jump to the second trial and notice that the unregulated bus component has failed at the start of this trial as shown in Figure 9.21. In fact, each trial's startup conditions will likely be different.

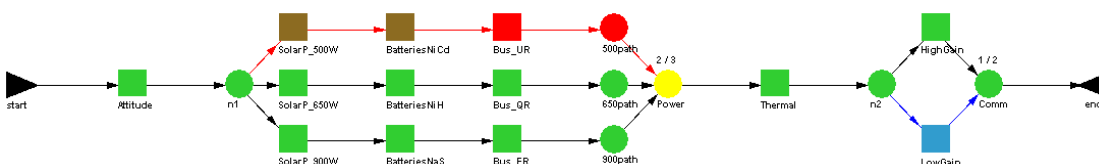


Figure 9.21 Failure at Simulation Time Zero for the Second Trial

Click on the depressed Pause button and allow the simulation to run to completion. The results of applying the random startup method to this model are shown in Figure 9.22.

Results from 10 runs of sim time 730.500000:

Parameter	Minimum	Mean	Maximum	Standard Dev	SEM
Availability	0.624480040	0.828625442	0.972746010	0.098215265	0.031058394
MTBDE	65.168953	114.015859	236.863654	62.352589	19.717620
MDT	6.636346	20.170034	39.188190	10.441145	3.301780
Reliability	0.000000000	0.000000000	0.000000000	0.000000000	0.000000000
Conditional Reliability	0.000000000	0.000000000	0.000000000	0.000000000	0.000000000
System Failures	3	6.200000	8	1.932184	0.611010

Figure 9.22 Results with Startup Transients Removed Using Random Startup Method

Notice that the availability value of this simulation is different from the availability of the previous method of transient control (i.e., 0.8286 versus 0.8596). A drawback to the random startup method is that the logistical aspects of a model have not been jumbled. All spares are fully stocked and all the repair people are available at the beginning of each trial. RBDs that contain significant logistical aspects might be better served to use the delayed statistics gathering method of managing startup transients. If an RBD does not contain significant logistical aspects, however, this method is far superior to delayed statistics gathering because it is faster since simulation time is not discarded. The difference in the availabilities of the two methods is such that it implies that the logistical characteristics of this model are causing significant system red time.

Recall that a byproduct of the random startup method of managing transients is that life can be exhausted on components before they start a simulation trial. It is common for a system to be comprised of subsystems that have varying degrees of life previously exhausted. When modifications are being contemplated for a system that has already been operating for some time, exhausting some amount of life for those components affected by aging often yields better results. Open **Transients5.rbd** and

look at the Simulation Theory tab of the Block Properties dialog box of the low gain antenna component. As shown in Figure 9.23, we have decided that the low gain antenna previously operated for 2,000 hours before being placed into the current system.

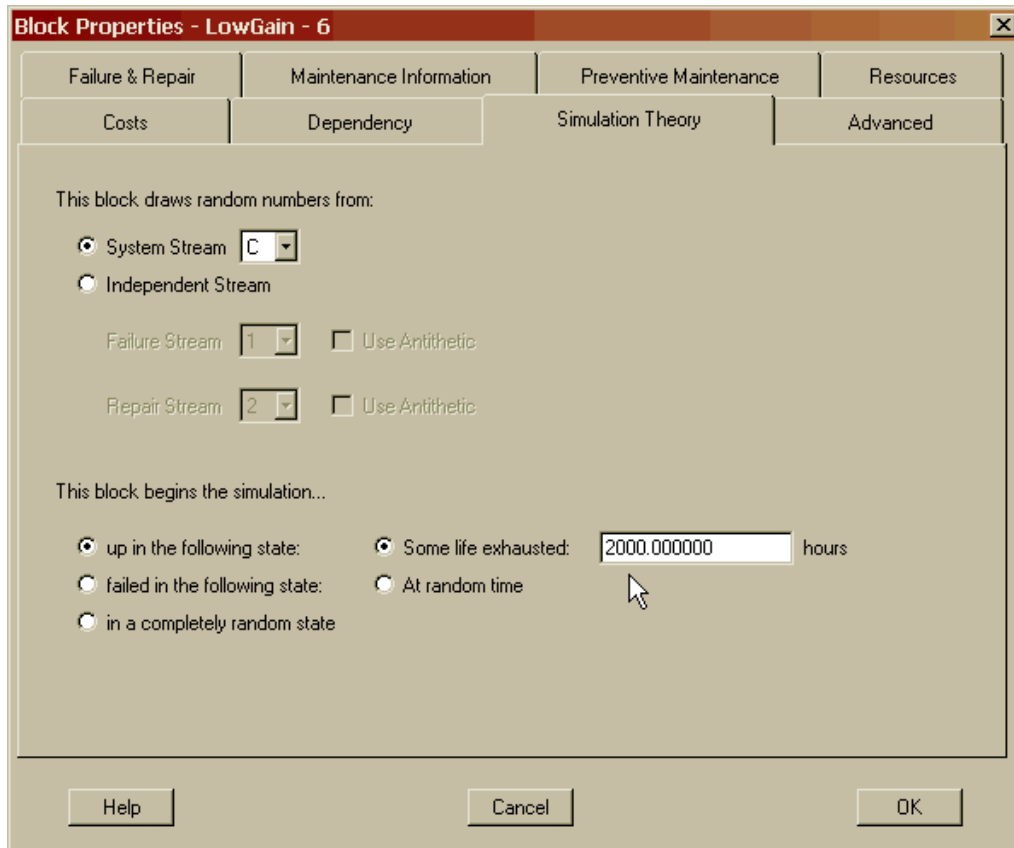


Figure 9.23 Life Exhausted for the Low Gain Antenna

We will only retain this facet of the random startup method for the low gain antenna component as we progress through the remaining chapters of this book. All other components have been set back to starting brand new with no life exhausted. Thus in essence, we have decided that the results with the startup transients are more pertinent to our system than the steady-state results. For the remaining chapters that use this model, we are interested in the availability and MDT for the first month of operation. Simulate this model and note the results that are shown in Figure 9.24. Pay particular attention to the system's availability of 0.928 since we will refer back to this value when we conduct weak link analysis in Chapter 11.

7. What happens to components that are in their repair cycles when the transition point between discarding statistics and collecting statistics occurs?
8. Describe a system in which the startup transients are desired. That is, when would the use of delayed statistics gathering not be prudent?
9. Can failure-truncated simulations with delayed statistics gathering be conducted? Is your answer the same for cycle-truncated simulations?
10. A component has a 2-hour pre logistic delay, an average hands-on repair of 5 hours and a 3-hour post logistic delay. Assuming no lack of spares, lack of resources or dependency issues, what state is the component in if it starts a simulation failed with 35 percent of its repair cycle completed?
11. A component has a 2-hour pre logistic delay, an average hands-on repair of 5 hours and a 3-hour post logistic delay. Assuming no lack of spares, lack of resources or dependency issues, what state is the component in if it starts a simulation failed with 85 percent of its repair cycle completed?
12. If multiple components start failed and insufficient spares or resources are available to initiate all the repair cycles, what priority scheme is used to assign the available spares and resources?
13. Does it make sense to exhaust life on a component that fails exponentially?
14. If two components are in series and both are system dependent and go idle 100% of the time, is it natural for both components to start a random startup simulation down?
15. What might be a reason for a system always starting a trial without any failed components even though all of the components are engaging random startups?

Chapter 10



Objectives

1. Understand the purpose and benefits of Failure Effects View (FEV).
2. Understand the rules related to FEV.

Recommended Problems

1, 2 and 3

The component itself turned red, as did the 500path node at the end of the power string. This node feeds into the Power node, which requires 2-out-of-3 inbound paths to be considered up. This caused the Power node to turn yellow. The other components in the top string are locally dependent, and set to fail 10% of the time when the local string fails. Finally, the system indicator changed to the yellow face since the system is up with some maintenance ongoing. These effects can also be viewed in text form by clicking on the Display Failure Effects button and displaying the System Status Summary dialog box as shown in Figure 10.4.

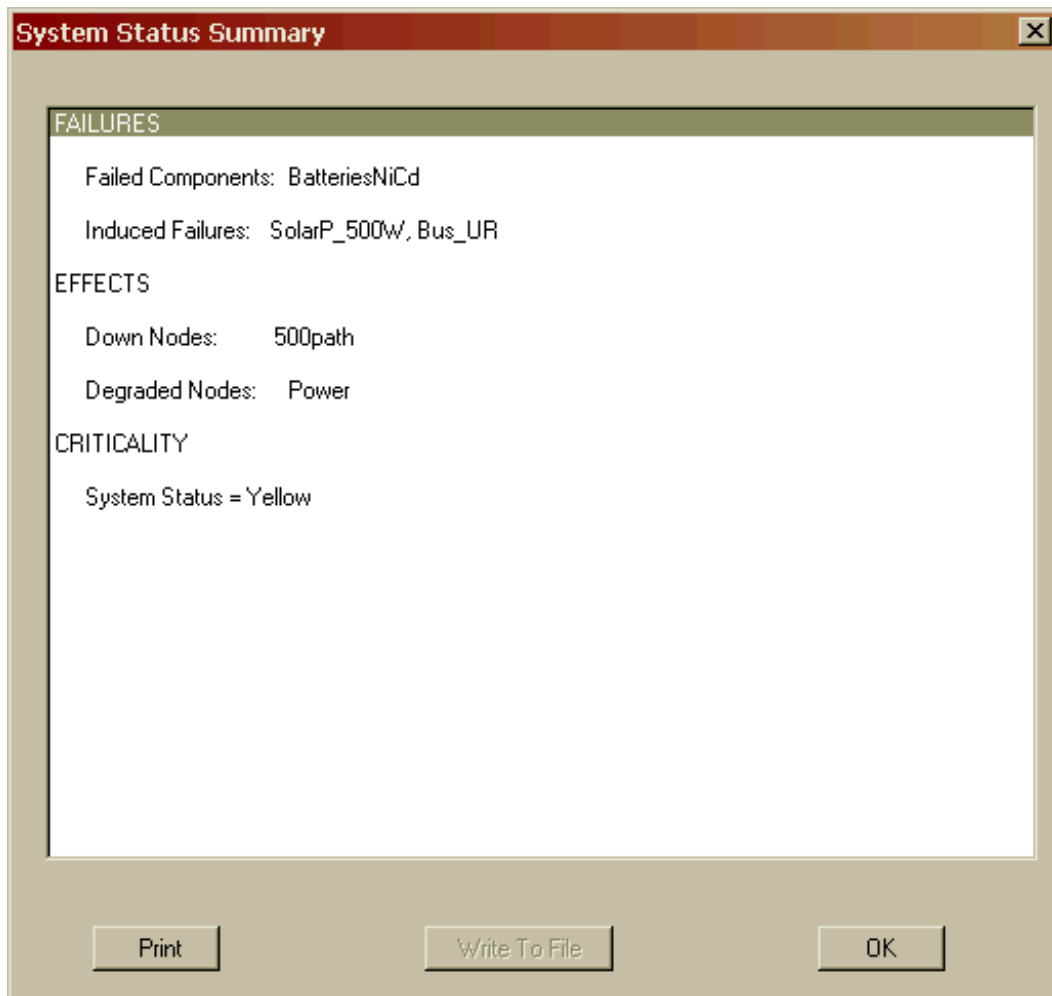


Figure 10.4 FEV System Status Summary

In this particular example, the 500 Watt solar panel and unregulated bus are displayed with red hatch lines. This indicates that these components change their status due to induced effects from the failure of the nickel cadmium battery component. Induced effects are changes to a component's status due to dependence options. Recall that these components were designated to initiate an idle status 90% of the time and to fail 10% of the time. If there are several options for dependency, the option to cause an induced failure will always be displayed if the probability to occur is greater than zero. The next highest priority state is preventive maintenance so if there is no chance of a

failure but a probability of going to the preventive maintenance state greater than zero, the dependent component would appear orange with hatch marks indicating an induced effect. The next highest priority is idle followed by the option to have no effect. If a component has several dependency options, the highest priority option is displayed first, but the user can always cycle through the remaining options by left-clicking on the hatched components. When another component is then manually failed, as in the case shown in Figure 10.5, hatched blocks from the previous step are unhatched. Induced state changes from this new failure are then hatched. In this case, the system has failed. Since the antenna group on the right is system dependent, these components were changed to the idle status. The Attitude component is also system dependent, but has a 10% probability of an induced failure upon a system failure.

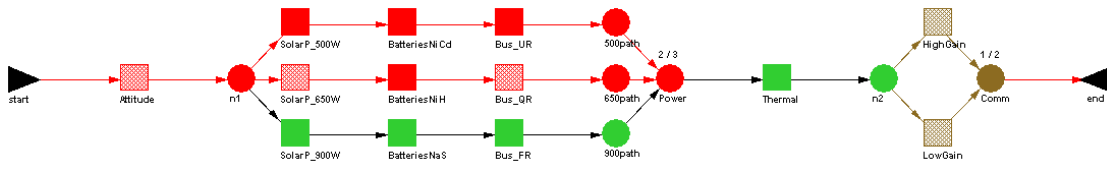


Figure 10.5 Updating the Induced Effects

At any time, a user can erase the current palette and start afresh. The small pencil eraser toolbar button will clear all components of the current failures and return them to the initial conditions, but will not reset phases if phasing is used. The large eraser button will clear all components and return the system to phase one if phasing has been implemented. Refresh the palette and left click the Bus_FR component. Again the worst case dependence option of affected dependent components is shown in Figure 10.6 (i.e., the SolarP_900 and BatteriesNaS components have failed).

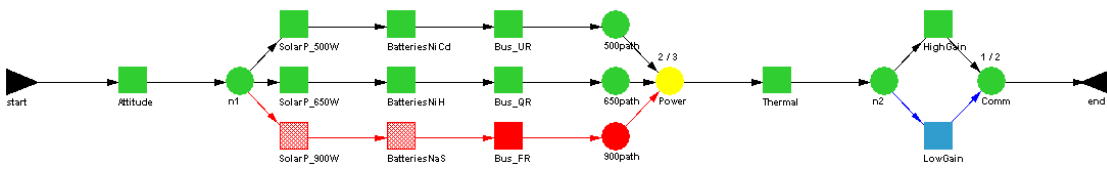


Figure 10.6 Refreshing and Failing Another Component

Failed components can also be returned to their running status by left-clicking the component again. What would happen if this situation occurred during a simulation and Bus_FR repaired? Left click Bus_FR and you see that the failed components are still failed. Refresh the palette and fail the Bus_FR component again. This time click on BatteriesNaS component next. BatteriesNaS has several dependence options and clicking the component while it is hatched will cycle through them. In this case, the component has turned orange indicating opportunistic preventive maintenance. Click on the SolarP_900 component twice and cycle it to the idle state. Your RBD should like the Figure 10.7.

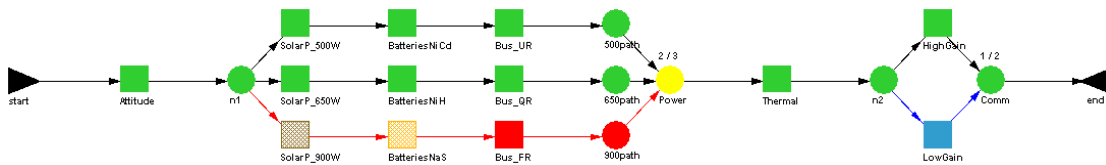


Figure 10.7 Failure Effects of the Fully-Regulated Bus

What do you expect to happen this time if you left click on Bus_FR? In the actual simulation if Bus_FR repaired it would then go to an idle status. Left clicking an unhatched red component in FEV mode always changes that component to green, so it is not exactly the same scenario as repairing the item.

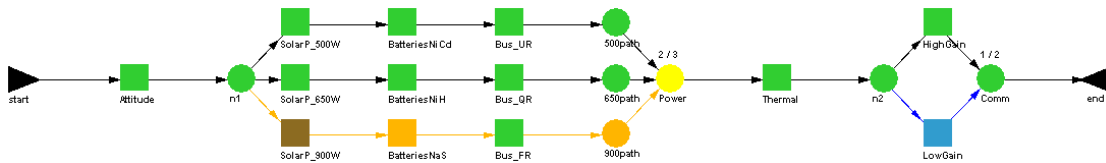


Figure 10.8 Repairing the Fully-Regulated Bus

The condition shown in Figure 10.8 can occur in this simulation since Bus_FR has its dependency set so that 20% of the time it remains in the same state when node 900path goes down. It is possible though to force the simulation into a condition using FEV mode that cannot occur in the actual simulation.

There are two useful files available to write from FEV mode. Selecting *Create Single Point Failure File* from the *File* menu will generate a file listing all components that can cause a system level failure by failing a particular component. This file includes the effects of induced failures as well. For the topology of this exercise, the Attitude and Thermal components will be in the single point failure file since they are series components. Depending on how dependence is designated, parallel components may also be single point failures.

The other file that can be created is the interactive log. This log can be opened by selecting *Open Interactive Log* from the *File* menu or by clicking the Open Interactive Log toolbar button. Once the file is named, the *Write to Log* menu item and toolbar are activated. Selecting the *Write to Log* menu item will write the text based system status summary dialog box information to a text file. During a session using FEV mode, users can write to the interactive log as often as they want. The file can be closed by selecting *Close Interactive Log* from the menu bar or toolbar. The file will automatically close if one exits FEV mode.

Problems

1. What does hatching on a component mean while in FEV view?
2. Can you always cycle through multiple states by clicking a hatched component?
3. If ten components in a 1-out-of-10 parallel system are each dependent on the one above them with a 1% chance of an induced failure, what happens when the top component is failed in FEV mode? Is the top component a single point failure? Is the fifth component a single point failure?

Chapter 11



Objectives

1. Understand how to use the weak link analysis function.
2. Understand what localized availability, dependability and reliability represents.
3. Determine which elements are the weak links of an RBD.
4. Understand the concept of a node's domain.

Recommended Problems

1, 2, 3, 4 and 5

Recommended Readings

Who's Eating Your Lunch? A Practical Guide to Determining the Weak Points of Any System (Murphy, Carter & Gass)

Weak Link Analysis

Significant improvements have been made in the realism of the models between the first one presented in Chapter 2 and the model we concluded with in Chapter 9. Implementing various distributions and considering the affects of logistics, dependence, standby and transients improved the fidelity of those models. Recall from Chapter 9 that the availability for the final model of the commercial space station was 0.928.

What if an availability of 0.928 is not good enough? Imagine that you are tasked with improving the system's availability such that you obtain a value of 0.948 or better (i.e., a two point increase in the system-level availability). Where might you begin? Given that the MTBDE and A_0 are positively correlated, you might try improving the MTBDE of the weakest element first and then determine how much improvement occurred in the system's availability. How would you determine, however, which element in particular is contributing the most to system down time? This chapter provides a methodology that facilitates the process of determining which elements of a system are the weakest links.

Open **WeakLinkAo.rbd** and notice that the system shown in Figure 11.1 is the same model we simulated in Chapter 9. Upon which elements might you focus your investigation in order to improve the system's availability?

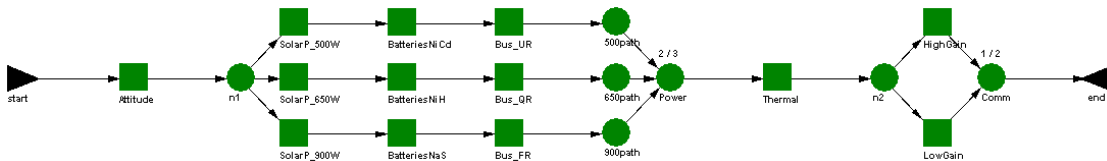


Figure 11.1 WeakLinkAo.rbd

This is not an easy question to answer. One might jump to the conclusion that the components in series (i.e., Attitude and Thermal components) are the main causes of the system's down time. Yet, series elements are often designed to have much higher reliabilities than parallel components since they have to operate without backup capabilities. Furthermore, if a series element was to fail often but is repaired quickly, it would not account for much system downtime. This leads us to an important rule of reliability optimization. One must investigate a system's logistical limitations and the failure distributions of its components as well as its series and parallel inadequacies to determine and implement improvements in a system's RAM characteristics.

Raptor provides an analysis technique that helps to focus investigations into a system's weaknesses. This technique is called weak link analysis and this chapter will provide the reader with an understanding of how this type of analysis is implemented within Raptor, and how to interpret the results of such an analysis.

Notice that the weak link analysis toolbar button for this model is depressed (Figure 11.2), which indicates that weak link analysis will be performed whenever a simulation of this system is initiated. The three analysis buttons (i.e., weak link, capacity and cost) all instruct Raptor to engage code that is not normally exercised thus improving the speed of a simulation when these capabilities are not desired.

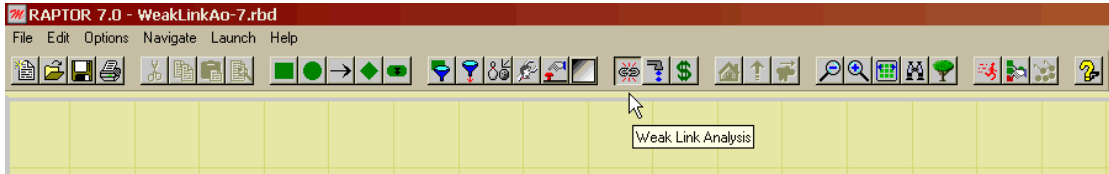


Figure 11.2 Workspace View Weak Link Analysis Toolbar Button

Simulate this system using the existing simulation conditions (i.e., 730.5 hours, 10 trials, with graphics) and notice that upon completion of the simulation, the availability of each component and node is displayed to three-digits of accuracy in the center of each element as demonstrated in Figure 11.3.

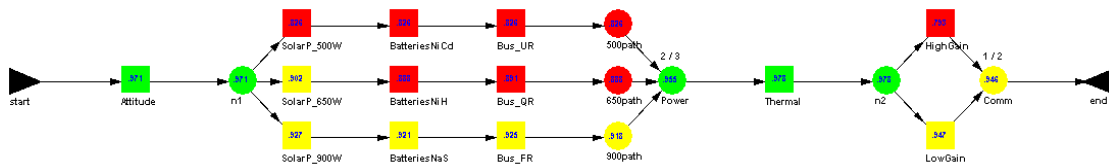


Figure 11.3 Component Availability

At the end of the simulation, the elements are all red, yellow or green in color. The red serial elements should be the first to be considered for improvement, followed by the yellow and then the green elements. A user can define the red, yellow and green thresholds used for weak link analysis by selecting the Color Thresholds button on the Simulation View toolbar. This button is available at the end of a simulation and when clicked, the Weak Link Analysis tab of the System Settings dialog box appears as shown in Figure 11.4.

This tab allows users to define a range for the red, yellow and green colors. In Figure 11.4 the red color threshold is currently defined to be any availability value from zero to less than 0.90. The yellow threshold ranges from values greater than or equal to 0.90 but less than 0.95, while the green threshold ranges from 0.95 to one. Notice that only the green and yellow thresholds are set. The red threshold is by definition the remainder since availability values only range from zero to one. This tab also allows the user to specify two other analysis types, dependability and reliability, which both will be discussed later in this chapter. Color thresholds can also be set before the start of a simulation by modifying the System Settings dialog box via the *Options – System Settings* menu item.

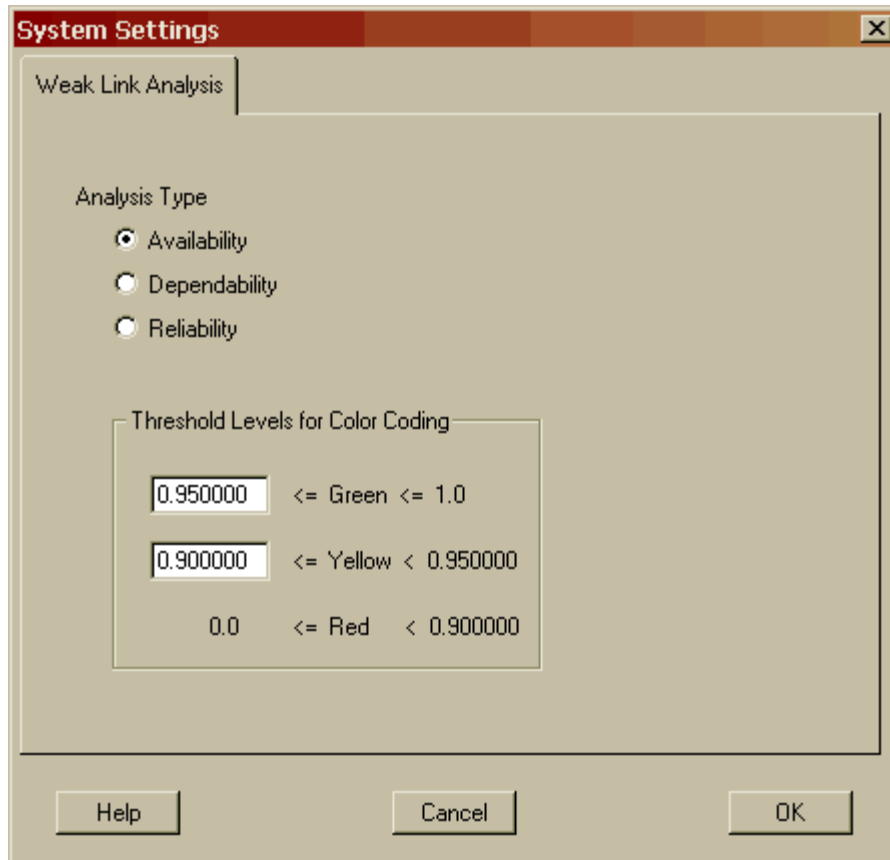


Figure 11.4 Weak Link Analysis Tab of the System Settings Dialog Box

The availability numbers on the block elements are relatively straightforward and easy to understand. For example, the Thermal component displays a value of 0.978 and thus we know that this element was in an up state on average 97.8% of the time and down only 2.2% of the simulated time. The weak link values displayed on the nodes can be a bit more difficult to comprehend. The key concept that must be understood in interpreting weak link analysis results for nodes is the notion of a node's domain. The domain of a node is the region of a system that affects the status of that node as a result of failures and repairs of that region.

For example, the status of the n2 node in Figure 11.3 is only affected by the status of the Thermal component. Thus, if the Thermal component fails, the n2 node changes its status to failed and is displayed red in color. When the n2 node turns red, it propagates the affect of the Thermal component failure to the End marker, causing the system status indicator to turn red as well. Hence, the Thermal component and the n2 node are by definition elements of the End marker's domain. Since all elements of an RBD can affect the status of its End marker, every element of an RBD is contained within an End marker's domain. The failure of the Thermal component has no affect on any other nodes in the RBD shown in Figure 11.3 and therefore is not an element of their domains. A domain diagram for this model is shown in Figure 11.5. This illustration takes the form of a hierarchical tree diagram so that every element underneath a

particular branch is within that element's domain. Creation of node diagrams is relatively simple for uncomplicated topologies and can be quite difficult for complex topologies. You can acquire some practice developing domain diagrams in the problems at the end of this chapter.

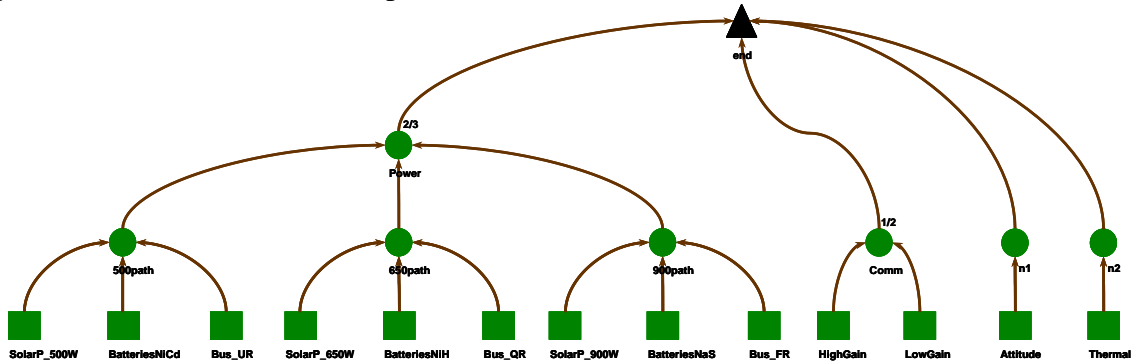


Figure 11.5 Domain Diagram of Current System

From this table, it is easier to read the availability numbers. There is an additional benefit in that nine digits of precision are displayed within this table. Select the Output Tables dialog box and then click on the Node Analysis tab as shown in Figure 11.6.

Node Name	Color	Minimum	Mean	Maximum	Standard Dev	SEM
500path	Red	0.685226696	0.826076575	0.969314281	0.103178141	0.032627793
650path	Red	0.640897084	0.888253146	0.976978674	0.107715886	0.034062754
900path	Yellow	0.804082188	0.917691793	0.983683118	0.060125337	0.019013301
Comm	Yellow	0.844048773	0.945765811	1.000000000	0.066338978	0.020978227
n1	Green	0.910718579	0.971230923	0.999315537	0.036091969	0.011413283
n2	Green	0.886235514	0.977957308	1.000000000	0.046500216	0.014704660
Power	Green	0.851353634	0.955165615	1.000000000	0.055741050	0.017626868

Figure 11.6 Node Analysis Tab of the Output Tables Dialog Box

Figure 11.6 displays the availability values for all the nodes to nine digits of accuracy and provides the minimum, maximum, standard deviation and standard error of the mean across all trials. Additionally, the weak link analysis color is provided in the second column. Based on Figure 11.6, the areas that should be considered for

improvement are the 500path and 650path power strings. That is, we should first consider improvements to the elements contained within the domains of these two nodes. Making improvements to other areas of the RBD, even serial components such as the Attitude and Thermal, would not yield as large of a return on investment with respect to system-level availability. Recall that the system-level availability is 0.928, which can be viewed on the first tab of this dialog box.

Select the Block Analysis tab of this dialog box and then click on the Mean column to sort the values from the lowest to the highest. It should be noted that all Raptor tables can be sorted both from low to high and from high to low by repeatedly clicking on a particular column heading. After sorting the table by the Mean column, it should appear as shown in Figure 11.7

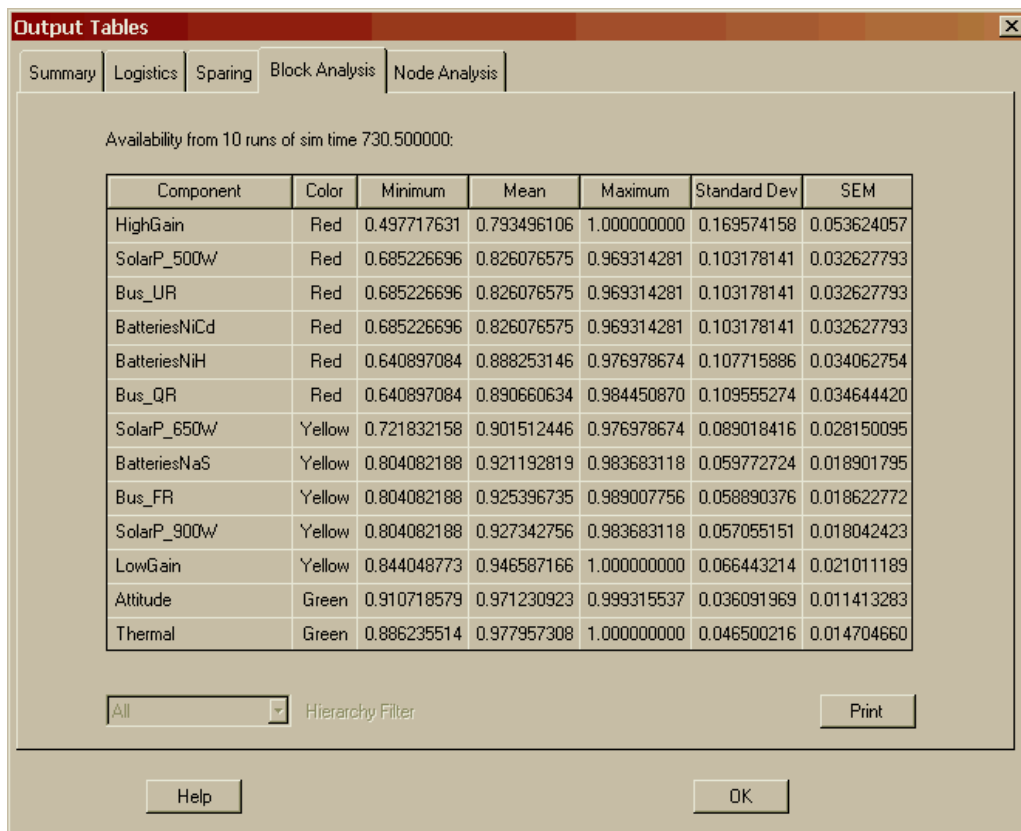


Figure 11.7 Block Analysis Tab of the Output Tables Dialog Box Sorted by Mean

Notice in Figure 11.7, that the 500 Watt solar panel, the unregulated bus and the nickel-cadmium battery components all have identical availability values. This should disturb some of the more statistically inclined readers since the odds of these three components having the same availability value to nine digits of precision is extremely rare given that they all have different failure and repair distributions. A more shrewd reader should recall from Chapter 6, however, that when the 500path node is down due to triggered preventive maintenance, the 500 Watt solar panel and the unregulated bus components are in preventive maintenance and the nickel-cadmium component

engages its dependency rules. Furthermore, with the dependence constraints set in Chapter 7, when the 500path node is down, the 500 Watt solar panel, the unregulated bus and the nickel-cadmium components all engage their respective dependency rules. These components often go down together when any one of them has gone down due to a hard failure or preventive maintenance. This explains why these three components display the same availability values.

Unfortunately, the availability parameter does not yield a clear answer as to which of these three components is the weakest. What is needed is a readiness parameter that ignores the outside influences (e.g., dependency, preventive maintenance, standby) on a component so that we can look at the inherent aspects of the components. Raptor allows for the weak link analysis calculations to be based on the *dependability* of the components, which is often known in the logistical communities as *inherent availability*. Let us review the definitions for a block's availability and dependability calculations as shown in Figures 11.8 and 11.9.

$$A_b = \frac{\textit{uptime}}{\textit{uptime} + \textit{downtime}}$$

$$A_b = \frac{\left[\begin{array}{c} \text{light green} \quad \text{dark green} \quad \text{lime green} \quad \text{blue} \quad \text{light blue} \quad \text{cyan} \end{array} \right]}{\left[\begin{array}{c} \text{light green} \quad \text{dark green} \quad \text{lime green} \quad \text{blue} \quad \text{light blue} \quad \text{cyan} \end{array} \right] + \left[\begin{array}{c} \text{brown} \quad \text{orange} \quad \text{tan} \quad \text{dark red} \quad \text{red} \quad \text{dark brown} \end{array} \right]}$$

Figure 11.8 Block Availability Definition

$$D_b = \frac{\textit{principal uptime}}{\textit{principal uptime} + \textit{principal downtime}}$$

$$D_b = \frac{\left[\begin{array}{c} \text{light green} \quad \text{dark green} \quad \text{lime green} \end{array} \right]}{\left[\begin{array}{c} \text{light green} \quad \text{dark green} \quad \text{lime green} \end{array} \right] + \left[\begin{array}{c} \text{dark red} \quad \text{red} \quad \text{dark brown} \end{array} \right]}$$

Figure 11.9 Block Dependability Definition

Notice from Figure 11.8 that a component's uptime is comprised of all up conditions of the functioning (i.e., shades of green) and reserved (i.e., shades of blue) states. A component's downtime is comprised of all up conditions of the constrained (i.e., brown color), scheduled (i.e., shades of orange) and failed (i.e., shades of red) states. Therefore, the availability of a component is its up time divided by the sum of its up and down times. This is the fundamental definition of availability and is acknowledged formally by the reliability engineering community. The dependability equation is a measure of the component's inherent reliability and maintainability

properties and excludes the outside influences of standby (blue), preventive maintenance (orange) and dependency (brown). By outside influences we are referring to those actions dictated by policies, which include the standby topology design, the preventive maintenance schedules and the dependency scenarios established for a system.

It is interesting to note that Raptor views the shades of blue as up states while the shades of orange and the brown color are considered down states. It is true that all of these states are precipitated by outside influences but in the case of the reserved states, the component could operate if it was called upon. The availability and dependability of a node are defined as shown in Figures 11.10 and 11.11.

$$A_n = \frac{\textit{uptime}}{\textit{uptime} + \textit{downtime}}$$

$$A_n = \frac{[\text{green} \quad \text{yellow} \quad \text{blue}]}{[\text{green} \quad \text{yellow} \quad \text{blue}] + [\text{brown} \quad \text{orange} \quad \text{red}]}$$

Figure 11.10 Node Availability Definition

$$D_n = \frac{\textit{principal uptime}}{\textit{principal uptime} + \textit{principal downtime}}$$

$$D_n = \frac{[\text{green} \quad \text{yellow}]}{[\text{green} \quad \text{yellow}] + [\text{red}]}$$

Figure 11.11 Node Dependability Definition

The node availability and dependability definitions follow the same format as those defined for a block. The node availability definition considers all possible states in which a node can exist while the dependability definition excludes outside influences. The node definitions are generally less complex compared to those of a block since a node cannot exist in as many distinct states. In fact, all three green states are represented by a single green state for a node. This amalgamated nature of the node's states is also true for the shades of blue, orange and red. Select the Color Thresholds button and change the analysis type to dependability as shown in Figure 11.12. Click on the OK button and notice how the values on the elements have changed. As shown in Figure 11.13, we now see a clear difference between the three elements of the 500path power string as a result of the dependability values being displayed.

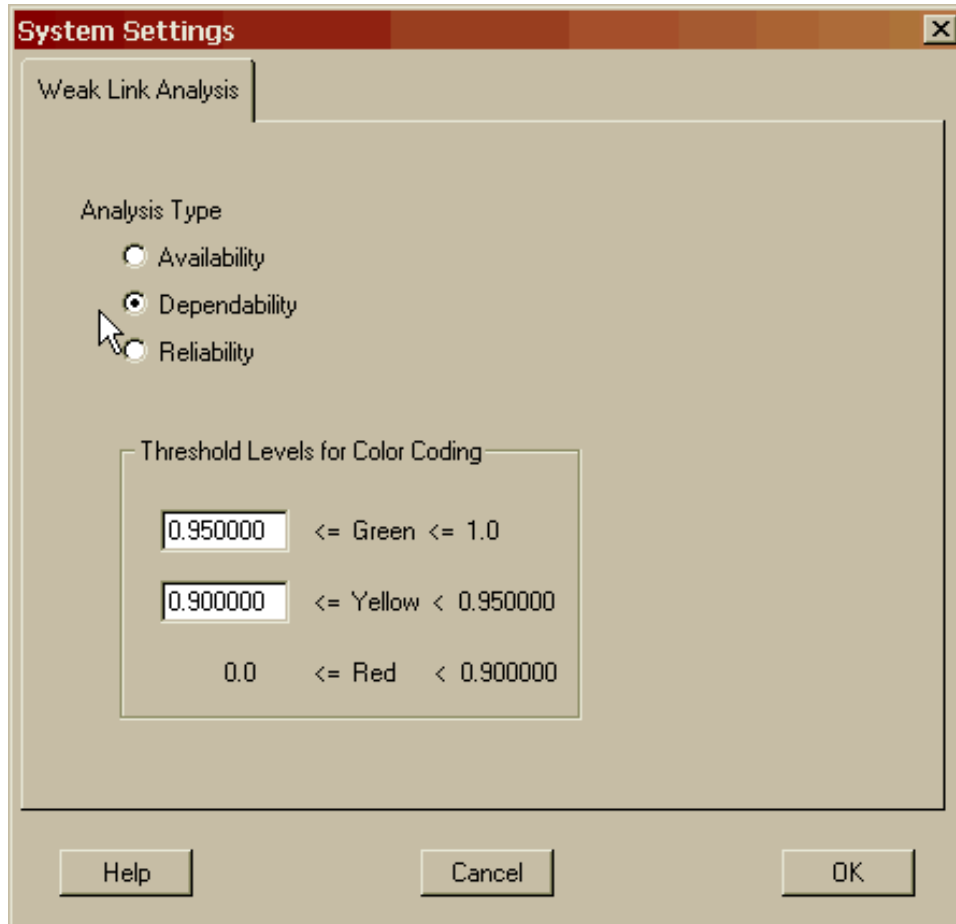


Figure 11.12 Setting the Analysis Type to Dependability

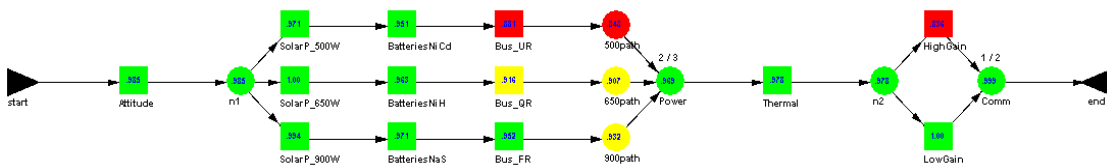


Figure 11.13 Component Dependability

Figure 11.14 zooms in on those components causing the most down time to the system. If we focus our efforts on the nickel-cadmium battery, unregulated bus and the quasi-regulated bus components, we might be able to achieve the original goal of this chapter. Recall, we are attempting to improve our current availability of 0.928 to 0.948. We now know where we will focus our improvement analysis efforts but what is not evident, is what attributes of these components should be altered. Since the unregulated bus component is the weakest of the three candidates, we will start with it and see if we can improve the system-level availability in such a manner as to achieve the required goal.

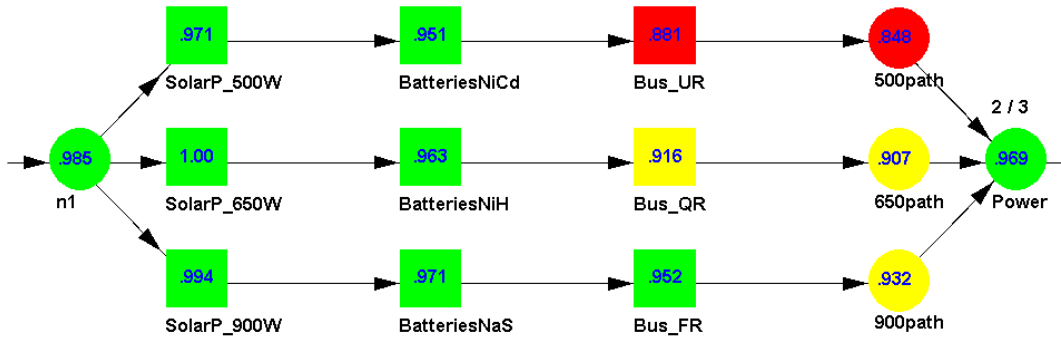


Figure 11.14 Component Dependability Close-Up

Earlier in this chapter we stated an important rule of reliability optimization is that one must investigate a system's logistics limitations, the failure distributions of its components as well as its series and parallel inadequacies to determine improvements to a system's RAM characteristics. We might attempt to add a redundant unregulated bus but given that this system is a space station, management has stated that adding mass to the system is a no-go decision. Thus, let us assume that we cannot change the topology of the system to improve the system-level availability. We might attempt to replace the unregulated bus with a component that fails less often but again management has rejected this idea since they cannot afford to replace the unregulated bus component until it has exceeded its designed life. This leaves us with one remaining avenue to improve the system's availability and that lies in the vast world of logistics.

Since we need to investigate the major logistical aspects of the unregulated bus component, we will start with its demand for resources and spares. Open the Block Input Tables dialog box and select the Maintenance Delays tab. Note the number of resources the unregulated bus needs to begin its repair cycle (see Figure 11.15). The number of resources needed seems like an opportunity for improvement since the unregulated bus component requires the entire astronaut staff to perform its repair. Maybe the maintenance procedures can be streamlined to require only four astronauts. Continuing our investigation, select the Sparing tab of this dialog box and recall that the unregulated bus draws its spares from the Electronics spare pool (see Figure 11.16).

Block Input Tables

Failure & Repair Distributions | Sparing | Preventive Maintenance | Maintenance Delays | Costs | Dependency | Advanced

Block Name	Pre-Logistics	Post-Logistics	Resource	# for Mx	# for PM
Attitude	Fixd (2.000000)	Fixd (1.000000)	Astronauts	3	2
BatteriesNaS	Fixd (0.150000)	Fixd (0.150000)	Astronauts	1	1
BatteriesNiCd	Fixd (0.250000)	Fixd (0.250000)	Astronauts	3	1
BatteriesNIH	Fixd (0.250000)	Fixd (0.250000)	Astronauts	2	1
Bus_FR	Fixd (2.500000)	Fixd (0.000000)	Astronauts	3	1
Bus_QR	Fixd (2.000000)	Fixd (0.000000)	Astronauts	4	1
Bus_UR	Fixd (3.000000)	Fixd (0.000000)	Astronauts	5	1
HighGain	Fixd (2.000000)	Fixd (1.000000)	Astronauts	1	1
LowGain	Fixd (0.500000)	Fixd (0.500000)	Astronauts	1	1
SolarP_500W	Fixd (2.000000)	Fixd (0.000000)	Astronauts	3	1
SolarP_650W	Fixd (2.000000)	Fixd (0.000000)	Astronauts	3	1
SolarP_900W	Fixd (1.000000)	Fixd (0.000000)	Astronauts	2	1
Thermal	Tmg (0.300000)	Tmg (0.300000)	Astronauts	2	1

All Hierarchy Filter Print

Help Cancel OK

Figure 11.15 Unregulated Bus Component's Resource Demand

Block Input Tables

Failure & Repair Distributions | Sparing | Preventive Maintenance | Maintenance Delays | Costs | Dependency | Advanced

Block Name	Source of Spares	Stock	New	Arrive at	Order	# of	Arrive at	Emergency
Attitude	Infinite	N/A	N/A	N/A	N/A	N/A	N/A	N/A
BatteriesNaS	Cells	3	1	730.0	0	1	360.0	168.0
BatteriesNiCd	Cells	3	1	730.0	0	1	360.0	168.0
BatteriesNIH	Cells	3	1	730.0	0	1	360.0	168.0
Bus_FR	Electronics	2	N/A	N/A	N/A	N/A	N/A	24.0
Bus_QR	Electronics	2	N/A	N/A	N/A	N/A	N/A	24.0
Bus_UR	Electronics	2	N/A	N/A	N/A	N/A	N/A	24.0
HighGain	Infinite	N/A	N/A	N/A	N/A	N/A	N/A	N/A
LowGain	Infinite	N/A	N/A	N/A	N/A	N/A	N/A	N/A
SolarP_500W	PhotoVPanels	2	1	4000.0	N/A	N/A	N/A	72.0
SolarP_650W	PhotoVPanels	2	1	4000.0	N/A	N/A	N/A	72.0
SolarP_900W	PhotoVPanels	2	1	4000.0	N/A	N/A	N/A	72.0
Thermal	Custom	1	1	1460.0	N/A	N/A	N/A	48.0

All Hierarchy Filter Print

Help Cancel OK

Figure 11.16 Unregulated Bus Component's Sparing Demand

The Electronics spare pool has an initial stockpile of two spares but can only be refurbished with an emergency launch of a Spaceship One type vehicle. After the Electronics spare pool is exhausted and once another failure of one of the buses occurs, a request for an Electronics spare is placed and then received twenty-four hours later. What if we changed the sparing strategy slightly? We could specify the initial stockpile to be larger so that the implementation of the emergency spare function is required a lot less often.

Changing the number of resources or the sparing strategy might allow us to obtain our goal of a system availability of greater than 0.948. Making both changes simultaneously, however, might not be prudent. Making a single modification to the system that achieves our goal would be a more parsimonious solution and generally a superior decision. Increasing the initial stockpile seems to be an easier solution since it merely requires adding additional funding early in the life-cycle of the system. Modifying the number of astronauts needed for a repair will require as a minimum the modifications to repair procedures, ground demonstrations of the new procedures and the modification to existing repair manuals.

Cancel from the Block Input Tables dialog box and select the *Options – Spares* menu item. From the Spare Pools dialog box modify the Electronics spare pool as shown in Figure 11.17 (i.e., increase the initial stock level from two to four). Click the OK button and then agree to update the Electronics spare pool with the new data. Simulate this model with weak link analysis engaged and note the system-level availability result shown in Figure 11.18.

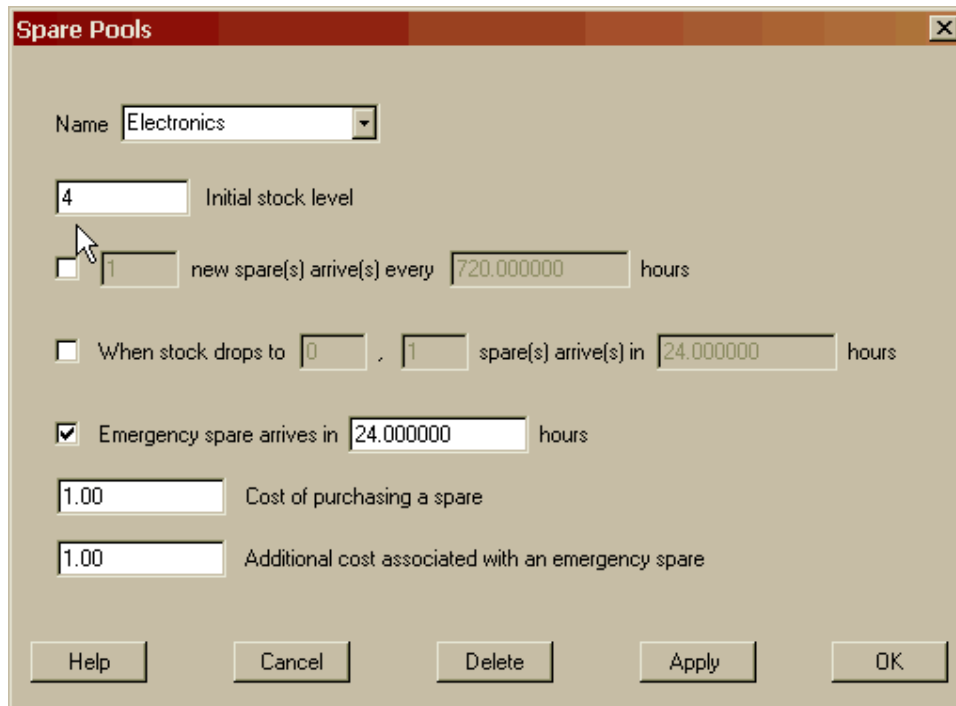


Figure 11.17 Modifying the Electronics Spare Pool

The system-level availability of 0.950 exceeds our goal of 0.948 and we have thus achieved our task of improving the system within all the constraints placed on us by management. The approach used in this chapter to make improvements in the system's availability value did not necessarily result in the optimal solution. Improvements in other elements' reliability or maintainability could produce the same or better results than those achieved in this lesson. Nevertheless, the approach used was at least logical, providing a much better chance of success than a purely trial and error approach.

When the weak link analysis preferences were discussed earlier there were three options for the type of analysis that could be performed. Thus far, we have not discussed reliability weak link analysis. Reliability is the probability that a system or subsystem exceeds a specified time without a failure. Analysis of a system which goes up and down often, usually involves the availability and dependability weak link analysis functions. Conversely, analysis of a system simulated for a relatively short duration will usually use the reliability weak link analysis function. Let us modify this model by saying that we are interested in the system remaining up for four hours during a particularly sensitive operation. Thus, we want to determine the reliability of the system for a four hour mission. Open the **WeakLinkRt.rbd** file and begin a simulation with the changes shown in Figure 11.20 implemented.

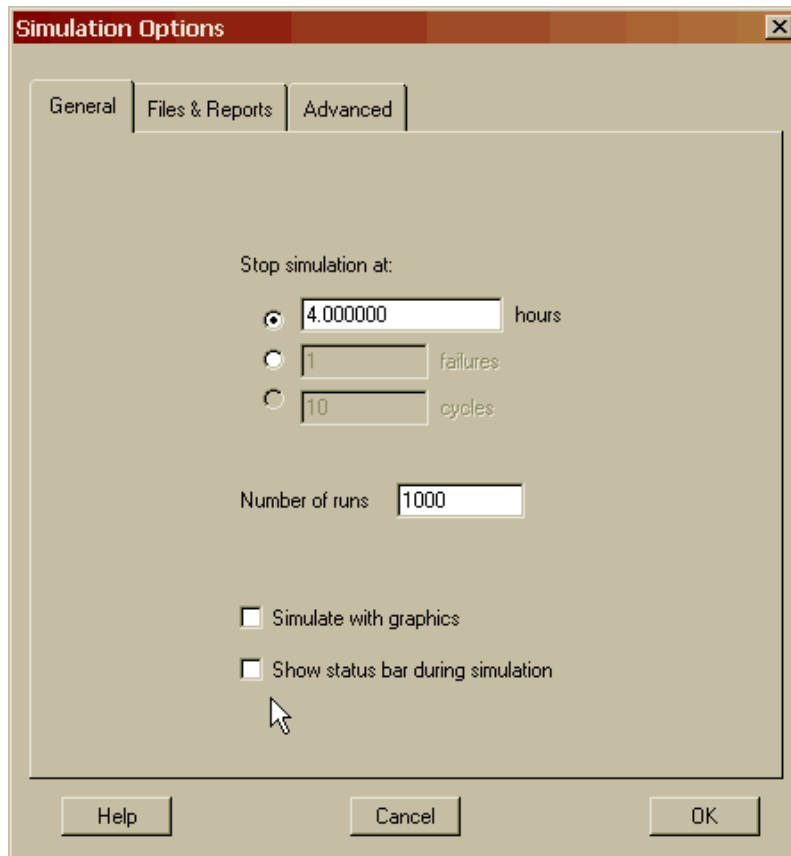


Figure 11.20 Simulating Missions without Graphics or Status Bar Updates

$$R(t)_b = \frac{\text{\# of missions completed without a block failure}}{\text{\# of missions attempted}}$$

$$R(t)_n = \frac{\text{\# of missions completed without a node failure}}{\text{\# of missions attempted}}$$

Return to the Workspace View and notice that the weak link analysis values and the color thresholds have all been removed. If you desire to view the weak link analysis values again, you can enter the Weak Link Analysis View by clicking on the Weak Link View button as shown in Figure 11.23. The Weak Link Analysis View is similar to the Simulation View except that some toolbar options are not available. This view is not available if you have not yet conducted a weak link analysis or if you have modified the RBD since the last simulation results were generated.

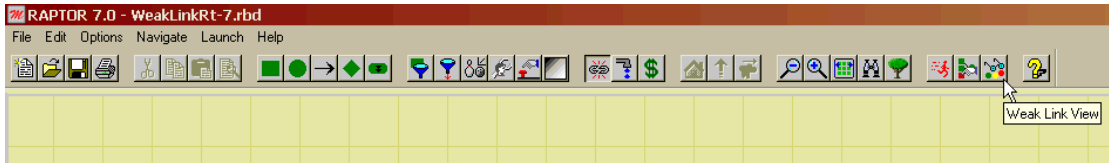


Figure 11.23 Weak Link Analysis View Button

You may have noticed, that the Node Analysis tab of the Output Tables dialog box (see Figure 11.6) is a listing containing the availability, dependability or reliability for every node contained within an RBD. Occasionally, it is desirable to display only a certain subset of crucial nodes. This can be accomplished by deselecting the checkbox on the Advanced tab of the Node Properties dialog box as shown in Figure 11.24. When this checkbox is not marked, the weak link analysis values for a node will not be displayed within the Node Analysis tab. Nodes are often used to make the appearance of a model more pleasing and thus these nodes would not be viewed as central to the understanding of the system's RAM parameters. When the Weak Link Analysis button is engaged, the analysis values are always calculated by Raptor so if you desire, you can always add a node's values back to the Node Analysis tab. This feature is not available for a block since it is viewed that every component's readiness parameter is crucial to the understanding of the system-level RAM parameters. Furthermore, Raptor calculates the availability, dependability and reliability for every element simultaneously so there is no need to re-simulate if you change your mind as to which readiness parameter you desire.

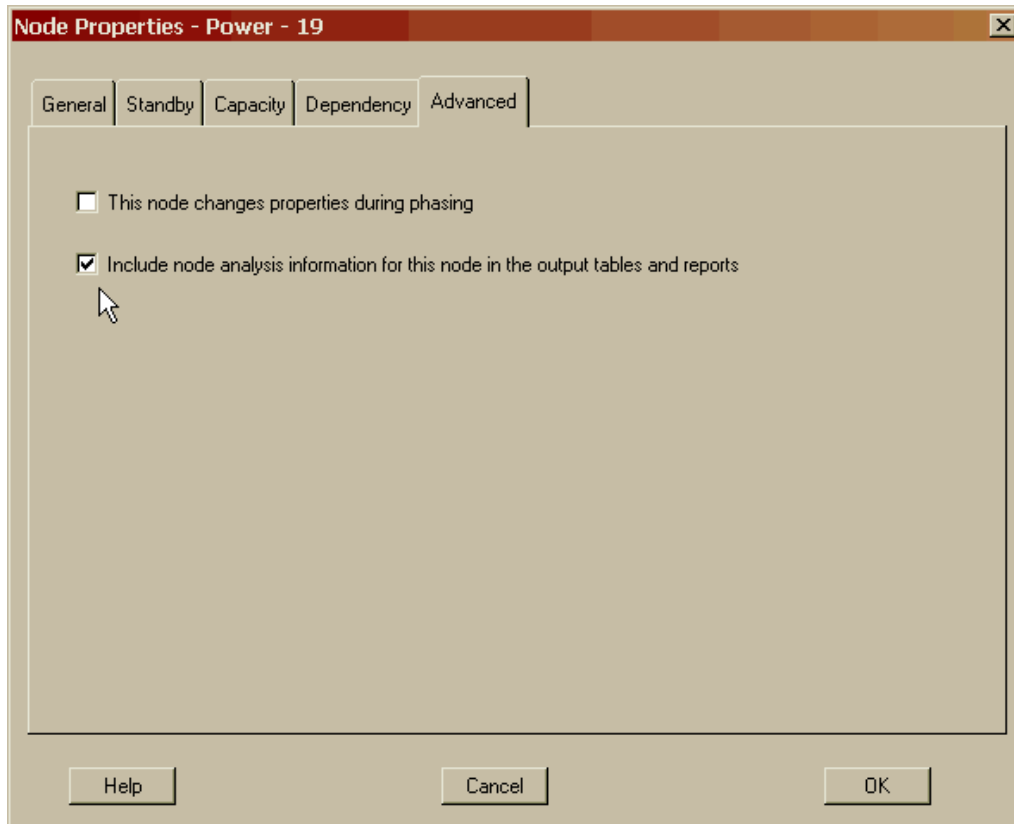


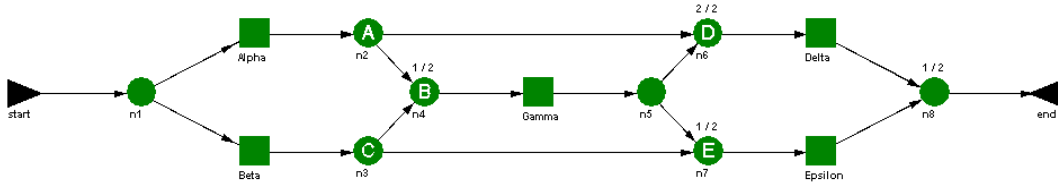
Figure 11.24 Display Node Analysis Information for a Particular Node

To summarize this chapter, weak link analysis provides localized performance values so that the weak links of a system can be determined. Weak link analysis is performed with respect to the failure and repair distribution of each component, the system's logistical constraints, and the topological structure of the system. Thus, weak link analysis yields far more powerful insights than a simple review of the component's mean life, or simply focusing on improving components that are in series.

Problems

1. Imagine an RBD that is simply a long string of several components all in series. Is the product of these components' availability weak link analysis values equal to the system's availability? Why or why not?

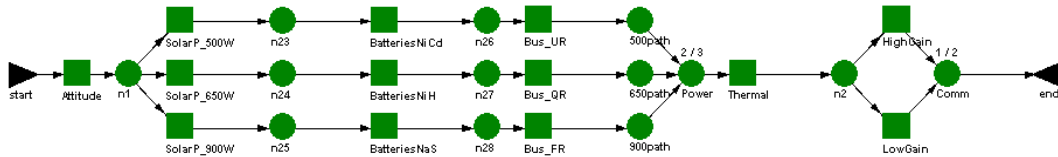
2. Explain what the node availability values represent for each of the nodes marked in the figure below (i.e., A, B, C, D, E and F). Components are independent and have no logistical delay times.



3. Name three general classes of design changes that can be made to improve the RAM characteristics of any system.

4. Explain why availability is not the best weak link analysis parameter to use to evaluate component performance and recommend improvements for a system that contains significant internal dependencies.

6. If the system shown below was created by taking the **WeakLinkDo.rbd** file and only adding nodes, would this system obtain the same weak link analysis values for the blocks? What about for the nodes?



Chapter 12



Objectives

1. Understand how to input cost information for blocks and events.
2. Understand all the input cost fields.
3. Be able to interpret the cost outputs.
4. Be able to generate a cost report.

Recommended Problems

1, 2, 3 and 4

Cost Analysis

Making changes to a system's configuration or logistics is not prudent without consideration of the financial impacts of these changes. Raptor allows a user to capture numerous cost parameters so that informed decisions can be made with respect to RAM and cost tradeoffs. Raptor's cost fields are unit-less so that analysis can be conducted in dollars, pounds, yen or any other currency the user desires. Raptor limits the accuracy of the cost fields to the nearest cent or 1/100th unit of currency.

Recall from the previous chapter concerning weak link analysis that we determined the unregulated bus component and its associated sparing pool were the likely weak links of the system. In this chapter, in addition to retaining the logistical changes to the system made in Chapter 11, we will conduct analysis to determine the cost impact of adding a redundant unregulated bus component to further improve system-level availability. Open **Cost1.rbd** and again note the similarity of this RBD in Figure 12.1 with those of previous chapters.

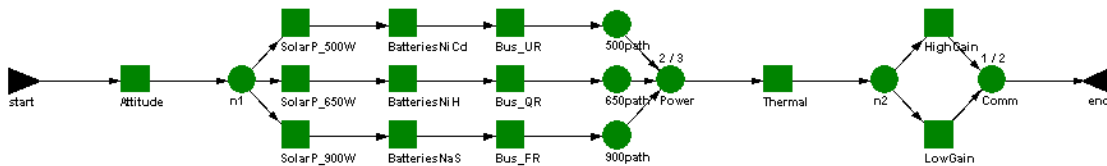


Figure 12.1 Initial RBD for Cost Analysis

This RBD is functionally identical to the final model used in Chapter 11 (after the logistical changes were implemented) except that cost fields are no longer set to their default values of 1.00. Let us explore the cost fields associated with a component by opening the Attitude component's Block Properties dialog box. Select the Costs tab and note that the cost fields for this component have already been entered as shown in Figure 12.2. Fourteen cost fields can be designated for any component.

The cost fields are grouped into three categories of up states, down states and fixed costs. There are two costs that comprise the up state cost group and include the cost to operate or run the component as well as the cost for that component to be in a standby state (recall that the standby state is considered an up state). For these two cost fields, a cost is incurred per unit time whenever a component is in a good state. Thus, a charge is assessed for each unit of time a component is or could function properly.

There are eight costs that comprise the down state cost group. The first two are associated with the cost to repair a component. These costs represent the per-unit time cost to repair a component and are only related to the actual hands-on maintenance time of a component's repair cycle. In addition to a per unit time repair cost, a fixed cost can be accrued for each repair occurrence. This field is shown to the right of the hands-on maintenance cost line. For example, some home service professionals will charge \$30 to come to your house regardless of the predicament, plus a \$75 per hour rate to accomplish the actual hands on repair.

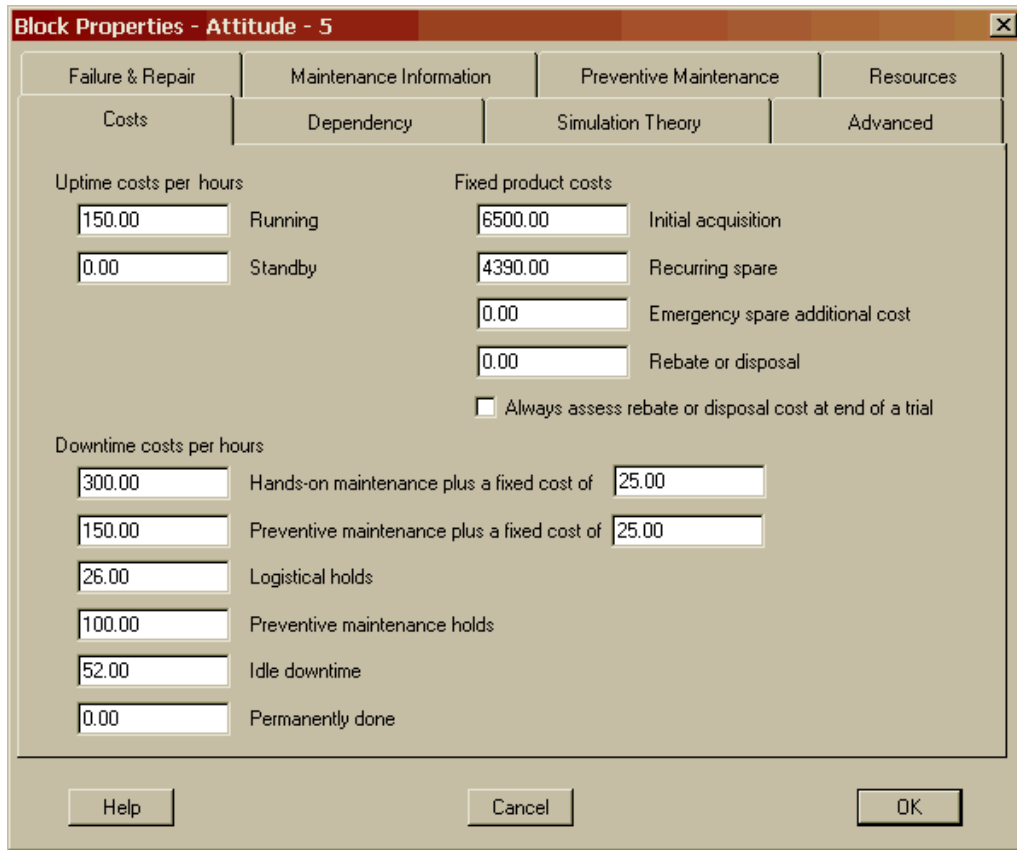


Figure 12.2 Attitude Component's Cost Fields

The next two cost fields are associated with the cost to perform preventive maintenance. These expenditures represent the per-unit time cost to conduct preventive maintenance activities as well as a fixed cost for each preventive maintenance occurrence. The next two fields deal with the time spent in a hold status due to either logistical or preventive maintenance reasons. Both of these fields accrue costs on a per unit time basis. The seventh down state cost is the per unit time cost of being in an idle state, which is caused by dependency relationships. The remaining down state expense is the per unit time cost of being in a permanently done state, which can occur when a component has failed and there is no possibility of this component operating again within a particular trial.

At this point, for each of the major colors that a component can display during a simulation, an associated cost can be defined. In two cases, the shades of the color group have been joined to form one cost category. The shades of green (running) and blue (standby) have been fused into two cost categories, respectively. Thus, a user cannot specify a cost for a component being in a cold, warm or hot standby state simultaneously. Based on Chapter 8, it should be understood that a component cannot occupy more than one of these standby sub-states within the same model. However, we will see in Chapter 17 that even though a component can exist in all three of the

green shades within one model, there is still only one cost field associated with this functioning state.

There are four costs that comprise the fixed cost group. The first field specifies an initial cost of a component, which might be higher than recurring spare costs due to setup expenses like implementing jigs, training personnel to use new equipment, the cost of the first production unit, etc. The second field is the cost of each spare in a component's stockpile and the cost for any spares ordered during a simulation. The next field is the additional cost for acquiring an emergency spare. For example, when an emergency spare is ordered, the cost of the spare (i.e., the previous field) and the cost to ship that spare quickly are incurred. Thus, this cost is always an additional fee to the previous cost associated with recurring spares. The last field of the fixed group pertains to the cost to dispose or salvage a component that reaches a done state within a trial. If a cost is incurred for the disposal of the component, then a positive value in this field should be used. If a rebate is gained by salvaging the component once it has reached its done state, then a negative value should be placed in this field. This is the only cost field that permits a negative cost value (i.e., rebate). For the Attitude component it has neither salvage nor disposal costs since the likely scenario is that this component, as well as some of the others, will eventually burn up in the Earth's atmosphere and there is generally no cost associated with this type of action.

In some cases, components will be disposed of or salvaged for a rebate at the end of a simulation trial regardless of whether the components reached their done state. To activate this feature, check the only checkbox on this tab, which is labeled "Always assess rebate or disposal cost at end of a trial". As shown in Figure 12.2, a done cost will not be assessed for the Attitude component, which correlates well to the rebate or disposal cost field being set to zero. The Attitude's standby cost is zero since this component will never be in a standby state (it has no alternates). The Attitude's emergency spare cost has been set to zero since this cost is only related to the custom and spare pool replacement strategies. Recall that the Attitude component uses the infinite spares replacement strategy.

Cancel from this dialog box and open the Block Properties dialog box for the unregulated bus component. Select the Costs tab as shown in Figure 12.3. Similar to the Attitude component, the unregulated bus has no standby or done costs. Additionally, the unregulated bus component has a zero cost for the three preventive maintenance fields since this component has no scheduled maintenance activities defined. Notice in Figure 12.3 that two of the cost fields are visible but deactivated. These fields are the component's cost for a spare and the additional cost for acquiring an emergency spare. Bear in mind that this component obtains its spares from the Electronics spare pool and fields associated with a spare pool can only be modified from the Spare Pools dialog box. Since a spare pool is a system-level entity as opposed to a component attribute, costs for such entities must be set where they are defined to ensure that they have system wide impact. Thus, the deactivated fields are a protection mechanism to ensure that all components drawing from the same spare pool have the same sparing costs.

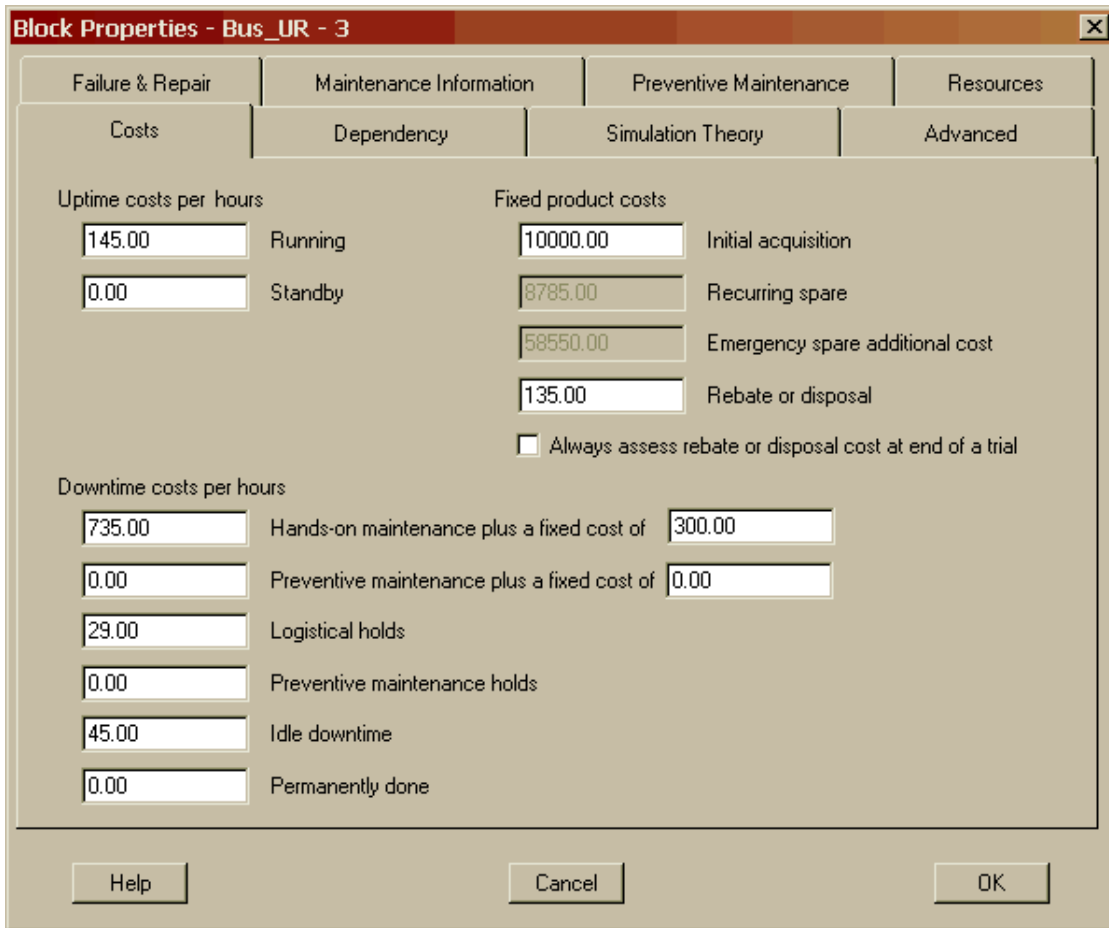


Figure 12.3 Unregulated Bus Component's Cost Fields

Cancel from this dialog box and open the Spare Pools dialog box by selecting the *Options-Spares* menu item. Select the Electronics spare pool from the pool name list box to display the properties of this pool as shown in Figure 12.4. The cost fields for this spare pool, as well as all the other pools, have already been updated to include cost information. The last two fields of the Spare Pools dialog box are used to enter the cost of obtaining a spare from this pool and the additional costs associated with obtaining an emergency spare. Notice that these two cost fields are identical to the two unregulated bus component cost fields that were shown deactivated in Figure 12.3. Any modifications to these fields in Figure 12.4 will be reflected in the corresponding component cost fields that are assigned to a particular spare pool. At the start of a simulation trial, all spare pools are charged a spare cost based on the number of spares in their initial stockpile. This applies to custom sparing as well as spare pools. Resource pools also possess cost fields and are shown in Figure 12.5.

Spare Pools [X]

Name:

Initial stock level

new spare(s) arrive(s) every hours

When stock drops to , spare(s) arrive(s) in hours

Emergency spare arrives in hours

Cost of purchasing a spare

Additional cost associated with an emergency spare

Figure 12.4 Spare Pool Cost Fields

Resource Pools [X]

Name:

Number of resources in the pool

Initial cost of each resource

Cost of using each resource per hours

Fixed cost of using each resource

Figure 12.5 Resource Pool Cost Fields

At the beginning of each trial, an initial cost is accrued for each resource. Additionally, there is a per unit time and fixed costs associated with using resources. When more than one resource is required, resource charges are multiplied by the number of resources employed for a repair. For example, if three astronauts are needed to conduct a repair, then the resource costs would be \$1,125 (i.e., 3 x \$375/hr) per hour of repair plus a fixed cost of \$261 (i.e., 3 x \$87/hr).

For some systems, there are costs associated with a system being down. This cost could represent loss of business or a penalty paid for not providing a service. The cost per unit of time for a system in a down state can be set in the System Settings dialog box via the *Options – System Settings* menu item as shown in Figure 12.6.

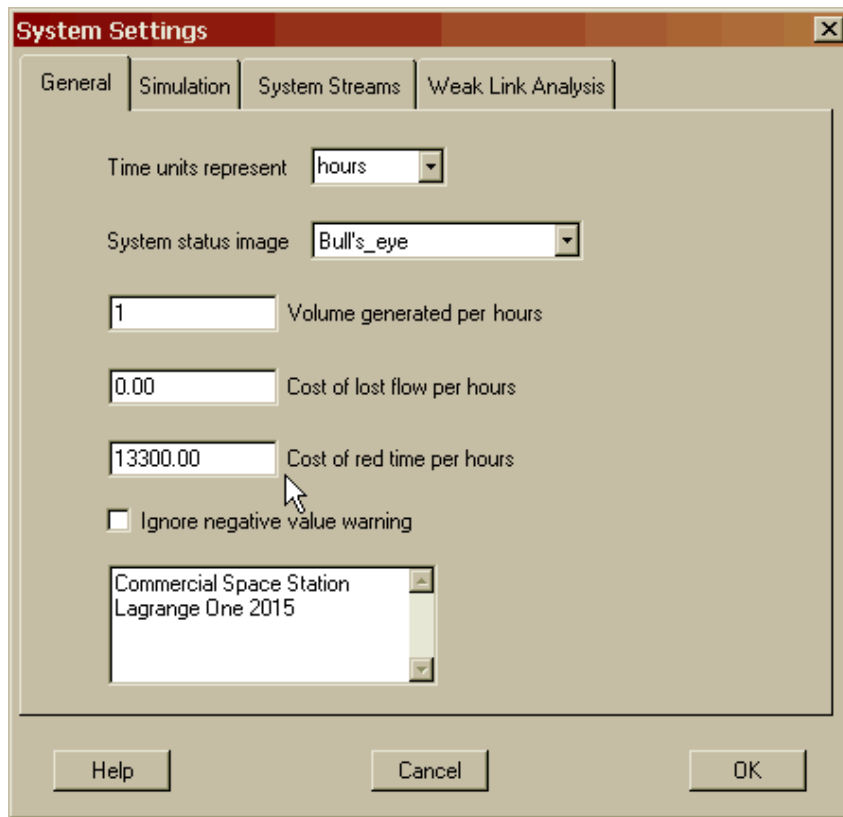


Figure 12.6 Cost of System Down Time

There is a critical cost field that has not yet been explored. This field is related to Raptor's capacity analysis feature and will be discussed in Chapter 14. This field is also entered in the System Settings dialog box as shown in Figure 12.6 and for now will be set to a cost of zero.

All cost input fields for a component can be viewed simultaneously on the Costs tab of the Block Input Tables dialog box, as shown in Figures 12.7, 12.8 and 12.9. It should be obvious that the number of columns exceeds the table's viewable area so a scroll bar is included at the bottom of the table to allow the user to view all of a component's cost fields.

Block Input Tables

Failure & Repair Distributions | Sparing | Preventive Maintenance | Maintenance Delays | **Costs** | Dependency | Advanced

Block Name	Running	Standby	Repair	Fixed Repair	PM	Fixed PM
Attitude	\$150.00	\$0.00	\$300.00	\$25.00	\$150.00	\$25.00
BatteriesNaS	\$300.00	\$0.00	\$675.00	\$675.00	\$600.00	\$125.00
BatteriesNiCd	\$375.00	\$0.00	\$1,000.00	\$375.00	\$300.00	\$125.00
BatteriesNiH	\$350.00	\$0.00	\$900.00	\$475.00	\$400.00	\$125.00
Bus_FR	\$115.00	\$0.00	\$680.00	\$300.00	\$0.00	\$0.00
Bus_QR	\$125.00	\$0.00	\$605.00	\$300.00	\$0.00	\$0.00
Bus_UR	\$145.00	\$0.00	\$735.00	\$300.00	\$0.00	\$0.00
HighGain	\$14.00	\$4.00	\$20.00	\$5,000.00	\$0.00	\$0.00
LowGain	\$15.00	\$7.50	\$30.00	\$7,500.00	\$0.00	\$0.00
SolarP_500W	\$225.00	\$0.00	\$500.00	\$9,025.00	\$100.00	\$750.00
SolarP_650W	\$225.00	\$0.00	\$500.00	\$9,025.00	\$100.00	\$1,000.00
SolarP_900W	\$200.00	\$0.00	\$350.00	\$9,025.00	\$150.00	\$1,250.00
Thermal	\$15.00	\$0.00	\$55.00	\$10,108.00	\$0.00	\$0.00

All Hierarchy Filter

Figure 12.7 First Third of the Component Cost Tab

Block Input Tables

Failure & Repair Distributions | Sparing | Preventive Maintenance | Maintenance Delays | **Costs** | Dependency | Advanced

Block Name	Logistic Hold	PM Hold	Idle	Done	Initial	Spare
Attitude	\$26.00	\$100.00	\$52.00	\$0.00	\$6,500.00	\$4,390.00
BatteriesNaS	\$55.00	\$133.00	\$100.00	\$0.00	\$9,000.00	\$3,075.00
BatteriesNiCd	\$84.00	\$233.00	\$150.00	\$0.00	\$4,000.00	\$3,075.00
BatteriesNiH	\$75.00	\$177.00	\$200.00	\$0.00	\$8,000.00	\$3,075.00
Bus_FR	\$18.00	\$0.00	\$25.00	\$0.00	\$11,000.00	\$8,785.00
Bus_QR	\$20.00	\$0.00	\$25.00	\$0.00	\$12,000.00	\$8,785.00
Bus_UR	\$29.00	\$0.00	\$45.00	\$0.00	\$10,000.00	\$8,785.00
HighGain	\$5.75	\$0.00	\$3.00	\$0.00	\$1,500.00	\$220.00
LowGain	\$5.75	\$0.00	\$3.00	\$0.00	\$900.00	\$100.00
SolarP_500W	\$170.00	\$70.00	\$33.00	\$0.00	\$3,125.00	\$3,125.00
SolarP_650W	\$150.00	\$50.00	\$33.00	\$0.00	\$3,125.00	\$3,125.00
SolarP_900W	\$150.00	\$50.00	\$33.00	\$0.00	\$3,125.00	\$3,125.00
Thermal	\$175.00	\$0.00	\$0.00	\$0.00	\$85,000.00	\$5,185.00

All Hierarchy Filter

Figure 12.8 Second Third of the Component Cost Tab

Block Name	Done	Initial	Spare	Emergency	Disposal	Assess Done
Attitude	\$0.00	\$6,500.00	\$4,390.00	\$0.00	\$0.00	False
BatteriesNaS	\$0.00	\$9,000.00	\$3,075.00	\$67,500.00	\$12,000.00	True
BatteriesNiCd	\$0.00	\$4,000.00	\$3,075.00	\$67,500.00	\$10,500.00	True
BatteriesNiH	\$0.00	\$8,000.00	\$3,075.00	\$67,500.00	\$8,000.00	False
Bus_FR	\$0.00	\$11,000.00	\$8,785.00	\$58,550.00	\$225.00	False
Bus_QR	\$0.00	\$12,000.00	\$8,785.00	\$58,550.00	\$175.00	False
Bus_UR	\$0.00	\$10,000.00	\$8,785.00	\$58,550.00	\$135.00	False
HighGain	\$0.00	\$1,500.00	\$220.00	\$0.00	\$0.00	False
LowGain	\$0.00	\$900.00	\$100.00	\$0.00	\$0.00	False
SolarP_500W	\$0.00	\$3,125.00	\$3,125.00	\$50,000.00	\$0.00	False
SolarP_650W	\$0.00	\$3,125.00	\$3,125.00	\$50,000.00	\$0.00	False
SolarP_900W	\$0.00	\$3,125.00	\$3,125.00	\$50,000.00	\$0.00	False
Thermal	\$0.00	\$85,000.00	\$5,185.00	\$12,689,644,000.00	\$0.00	False

Figure 12.9 Final Third of the Component Cost Tab

Cancel from the dialog box and notice that the cost analysis toolbar button for this model is depressed (see Figure 12.10), which indicates that cost analysis will be performed whenever a simulation of this system is initiated.

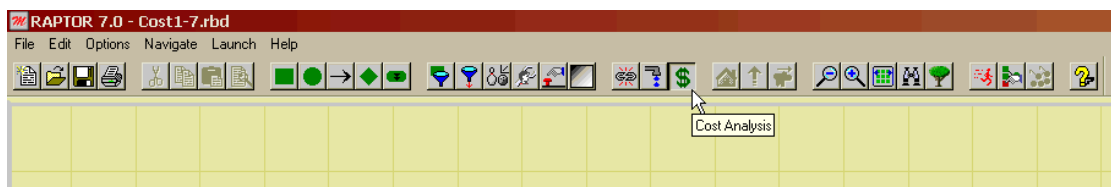


Figure 12.10 Workspace View Cost Analysis Toolbar Button

Let us conduct a simulation of the **Cost1.rbd** model and assess the impacts of populating this system with cost data. The cost results of this simulation are contained within several tabs of the Output Tables dialog box, which are shown in Figures 12.11 and 12.12. On average, this system costs approximately 21.462 million dollars to operate and maintain for one month with an availability of 0.950. A user should note that the addition of cost fields did not alter the RAM output parameters in any manner, which should be reassuring. Let us take a closer look at the cost results for each component by selecting the Block Costs tab of this dialog box, as indicated by Figure 12.12. Be sure to sort this tab from high to low based on the total costs.

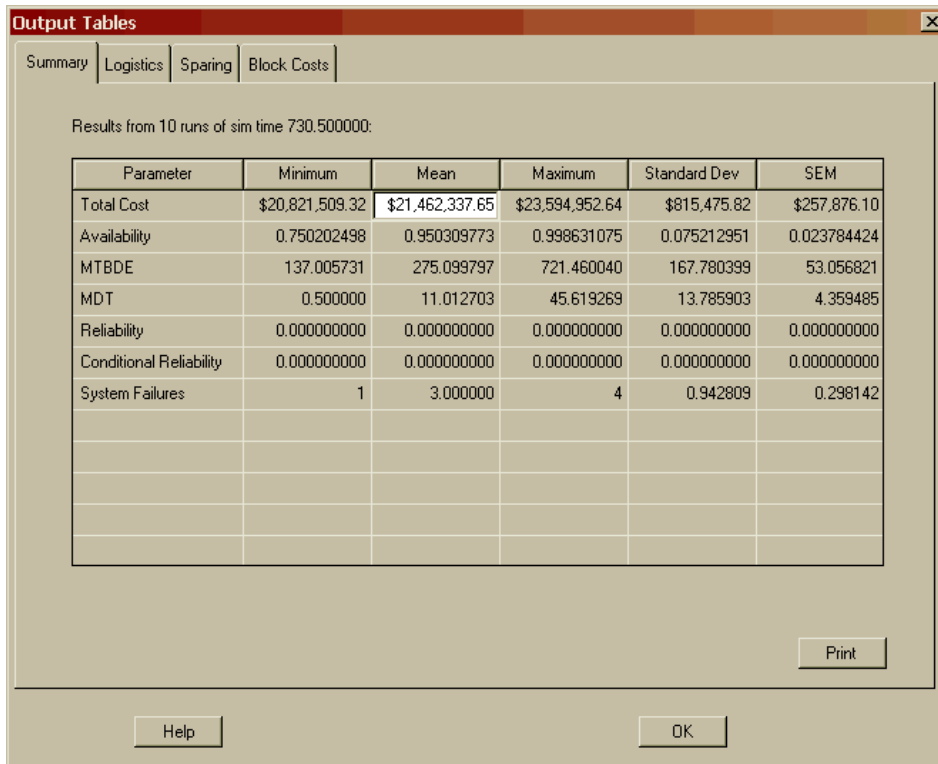


Figure 12.11 Overall Cost Results

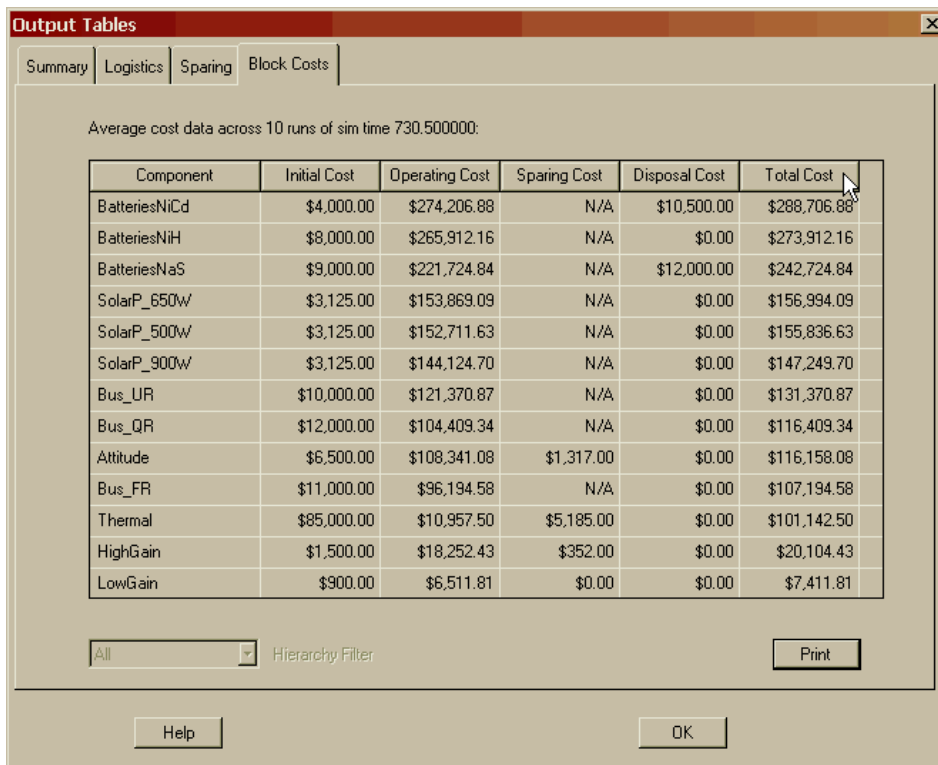


Figure 12.12 Component Cost Results Sorted by Total Cost

The Block Costs tab shows a life cycle breakdown of the cost for each component. The Initial Cost column is just the component's initial cost. The Operating Cost column is the combined cost for a component being in a variety of operating states throughout a simulation including running, repairing, hold, idle, preventive maintenance and standby states. If a component uses custom or infinite sparing, the cost of using spares is included in this table in the Sparing Cost column. If a component was permanently failed or if the option to always add done cost was selected, the disposal or salvage cost would be included in the Disposal Cost column. The total cost is displayed for each component in the last column and it is evident from Figure 12.12 that the battery components are the most expensive, followed by the solar panels.

Both the End of Run and the Final Results reports contain additional information on cost. The End of Run report contains a full cost assessment for each simulation trial, while the Final Results report has the same format but contains average values across all trials. In addition to the information in Figure 12.12, these reports contain a further breakdown of the operating costs by state for each component. Figure 12.13 is a portion of the Final Results report for this model.

Operating								
Block	Good	Repair	Idle	RepHold	PM	PMHold	Standby	Done
Attitude	106957.36	988.95	52.21	342.56	0.00	0.00	0.00	0.00
BatteriesNaS	203366.58	8612.40	2920.77	14.04	6811.04	0.00	0.00	0.00
BatteriesNiCd	237948.02	21726.75	6988.89	450.65	7092.57	0.00	0.00	0.00
BatteriesNiH	227825.70	24586.51	6932.44	209.36	6358.16	0.00	0.00	0.00
Bus_FR	78166.12	17436.20	469.86	122.40	0.00	0.00	0.00	0.00
Bus_QR	82082.47	21440.26	543.29	343.32	0.00	0.00	0.00	0.00
Bus_UR	92006.57	27013.34	1748.72	602.24	0.00	0.00	0.00	0.00
HighGain	7620.01	10508.60	89.39	34.43	0.00	0.00	0.00	0.00
LowGain	2334.27	0.00	89.39	0.00	0.00	0.00	4088.15	0.00
SolarP_500W	142768.81	4072.52	2764.67	582.19	2523.43	0.00	0.00	0.00
SolarP_650W	148336.81	0.00	2211.61	0.00	3320.67	0.00	0.00	0.00
SolarP_900W	137355.69	1223.99	1269.35	15.00	4260.68	0.00	0.00	0.00
Thermal	10957.50	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Totals	1477725.91	137609.51	26080.58	2716.19	30366.55	0.00	4088.15	0.00
Sparing								
Block	InitSpares	NewSpares	EmerShip					
Attitude	0.00	1317.00	0.00					
HighGain	0.00	352.00	0.00					
LowGain	0.00	0.00	0.00					
Thermal	5185.00	0.00	0.00					
Totals	5185.00	1669.00	0.00					
Cost_Summary								
Blocks	1865215.90							
Events	0.00							
SparePools	74505.50							
Resources	19039843.40							
SystemDownCost	482772.85							
Grand_Total	21462337.65							

Figure 12.13 Component Cost Section of the Final Results Report

Costs for spare and resource pools incurred during a simulation are also contained within these reports as shown in Figure 12.14. Spare pool costs include the initial stockpile costs, costs for ordering additional spares and the costs associated with acquiring emergency spares. Resource pool costs include the initial cost of possessing five astronauts, the fixed costs for using the astronauts and the hourly wages for the astronauts to perform repair activities. For this simulation, it should be observed that a significant contributor to the total cost of this model is the Astronaut resource pool costs (i.e., approximately 89% of the total cost).

SparePools	Initial	NewSpares	EmerShip	Total
Cells	9225.00	3690.00	0.00	12915.00
PhotoVPanels	6250.00	0.00	0.00	6250.00
Electronics	35140.00	2635.50	17565.00	55340.50
SparePool_Totals	50615.00	6325.50	17565.00	74505.50

Resources	Initial	PerUseCost	PerTimeCost	Total
Astronauts	18750000.00	3036.30	286807.10	19039843.40
Resource_Totals	18750000.00	3036.30	286807.10	19039843.40

Figure 12.14 Spare and Resource Pool Cost Section of Final Results Report

Open the **Cost2.rbd** file, which is a modified version of the previous RBD. A redundant unregulated bus was added, as shown in Figure 12.15, while retaining the correct dependencies. It is recommended that the user verify that the dependencies were maintained with this new topology using Raptor's Failure Effects View, which was discussed in Chapter 10.

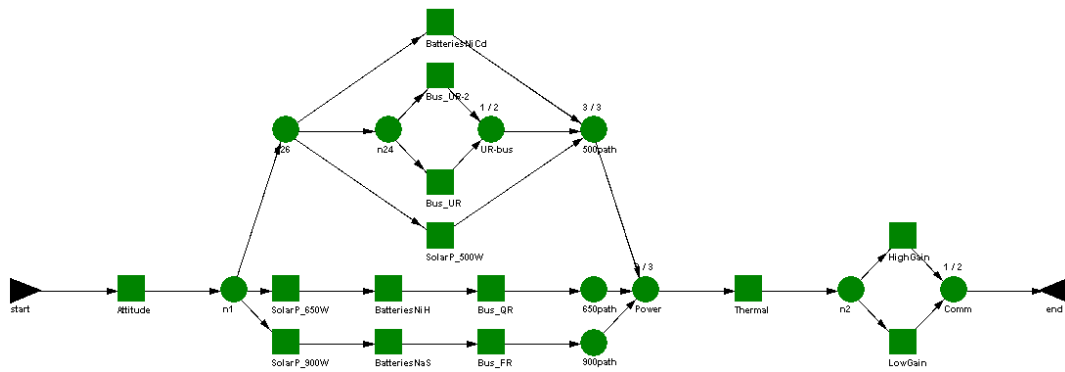


Figure 12.15 Modified Cost RBD

Simulate this RBD using the same parameters as in the previous simulation, which will allow us to determine the impact on the system's cost and RAM parameters resulting from such a modification. You should obtain identical results to those shown in Figure 12.16. The price has risen from 21.462 million to 21.624 million dollars due to adding the redundant unregulated bus component. The availability has increased

from the original value of 0.950 to 0.959. Therefore, for \$161,919 a month, this system's availability increased slightly more than 0.9 percent (i.e., 0.009). The question is whether this increase in cost is worth the increased availability. There is no clear answer to this question but for some systems, a small increase in availability is worth a lot more than \$162,000 (e.g., communication networks, missile defense shields, etc.). Since the answer to this question is highly dependent on the system being considered, Raptor cannot resolve this question directly. Nevertheless, Raptor provides the analysis methodology to conduct cost analysis versus system RAM attributes and therefore facilitate the formulation of sound design decisions.

Parameter	Minimum	Mean	Maximum	Standard Dev	SEM
Total Cost	\$20,934,952.13	\$21,624,256.62	\$22,587,298.96	\$577,015.83	\$182,468.43
Availability	0.891991115	0.959415024	0.998631075	0.043636659	0.013799123
MTBDE	84.673845	260.251192	726.564092	181.097416	57.268031
MDT	0.500000	6.990058	19.725123	6.979557	2.207130
Reliability	0.000000000	0.000000000	0.000000000	0.000000000	0.000000000
Conditional Reliability	0.000000000	0.000000000	0.000000000	0.000000000	0.000000000
System Failures	1	3.600000	8	1.897367	0.600000

Figure 12.16 Final Results with Redundant Unregulated Buses

• • •

An important consideration when conducting cost analysis for a system that will be in service for several years is the time value of money. Analysis conducted based on present year dollars may not represent a valid projection of costs ten years from now. Many spreadsheet tools exist that contain excellent time value of money capabilities. Instead of attempting to build in cost analysis algorithms and trying to mimic features that other tools do quite well, the approach Raptor takes is to provide cost data at specified time intervals so that the data can be post-processed. For example, Raptor

allows a user to simulate a system for ten years of operation and write a cumulative cost report at the end of each year. The user can now take this data to a spreadsheet post-processor and experiment with various inflation rates to project future costs. To engage this feature; select the Files & Reports tab of the Simulations Options dialog box. The checkbox labeled Interim cost results must be checked and the frequency of the report generation must be specified. For this example, Figure 12.17 indicates that Raptor will generate cost results every quarter month for a model with a simulation termination point of 730.5 hours. The format of the interim cost report is the same as the cost section of the End of Run or Final Results reports.

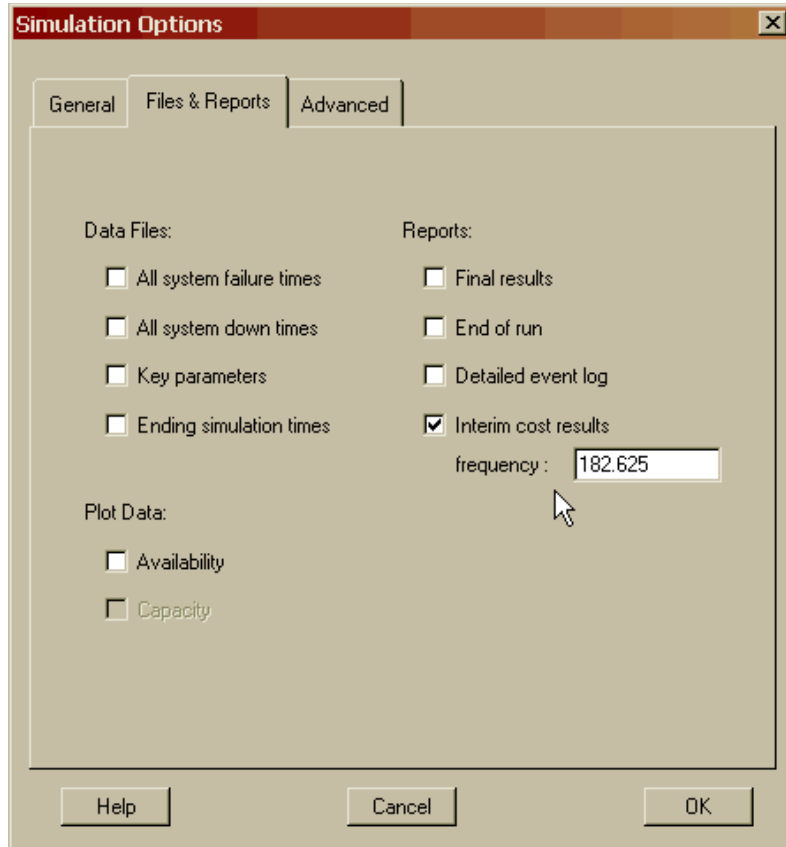


Figure 12.17 Generating Interim Cost Results

• • •

It is interesting to note that a component that is partially repaired at the end of a simulation trial will accrue a pro-rated repair cost based on the amount of repair completed. The fixed cost associated with a repair, however, is never pro-rated and is accrued at the start of the hands-on repair activities. If a component uses infinite sparing, the cost of a spare is incurred as soon as the spare is obtained and thus a full spare charge is always accrued for either a partial or clipped repair. For all other sparing strategies, costs for spares in the initial stockpile are accrued at time zero, and the costs for additional spares are accrued at the time spares are ordered.

• • •

It is also interesting to note that the new unregulated bus component had its failure and repair distribution random number streams designated as independent streams. Components that are being considered for inclusion or exclusion from a system should not draw their random numbers from system streams but from independent streams to ensure good scientific analysis when assessing the differences between two comparable models. The use of independent random number streams ensures that the differences observed are due to changes made to the models and not to the particular random numbers drawn during the simulations.

Problems

1. If a spare pool contains five spares at \$500 each, what would be the initial spare pool cost?
2. What cost field specifies the per unit holding cost for spares maintained within a stockpile?
3. What is the cost of repairing a component under the following conditions: 7 resources were required, total downtime of the component was 10 hours, logistics delay time was 2 hours, cost of repair is \$20 per hour plus fixed repair cost of \$15, cost of resources is \$15 per hour, the cost of a spare is \$1,250 and a hold cost of \$8 per hour?
4. If several components use spares from one pool, the sparing cost for these components is combined and listed in the spare pools section of the cost reports. How can you determine how much cost was due to each component?

Chapter 13



Objectives

1. Be able to convert a physical description of a system into a Raptor RBD.
2. Gain practice applying all the standby features of Raptor.
3. Be able to apply Weak Link and Cost analyses for the purpose of assessing design trade-offs.
4. Be able to use the preventive maintenance and standby features of Raptor to increase component and system level availability.

Hands On Example

The best way to understand how to apply and appreciate any software application is to actually use the software. A problem is presented below that should test and reinforce your understanding of the preceding lessons. It begins with the actual development of a system-level RBD (often the most difficult task in conducting reliability simulations), and ends with the generation of statistically meaningful results that could be used for design trade-offs. A pie-making facility (Figure 13.1) that consists of three main subsystems (power-electrical components, motors and pie-making mechanisms) will be studied.

System Description:

The power facility runs principally on AC power, but there is a battery back-up that can be turned on in the event of AC failure. The battery is turned off if the AC power is running, although it still burns some life when not active, about 10% as much as when it is running. The power system is regulated by a computer that requires a 1553 card for proper operation. There are separate motors that drive the main production line belt and the box folder. Both are necessary for the pie facility to be operational.

The pie mechanisms are the oven, the boxer, the emergency switch and the pie forming lines. The pie forming lines each have a mixer, a crust former, and a filling station. There are three pie forming lines, one each for apple pie, cherry pie, and rhubarb pie. The plant can only handle the capacity from two of three lines at any one time; the line not in use will be placed in cold standby.



Figure 13.1 Apple, Cherry and Rhubarb Pie Forming Lines

The plant runs 24 hours a day, 365 days a year. The facility produces fifteen pies per filling line per hour that it operates, and each pie generates an income of \$2.17.

A. Create an RBD of the pie-making facility in Raptor.

B. Open **PieFacility1.rbd** and note that the components have already been placed in the workspace with their relevant data, including failure and repair distributions (Figure 13.2), sparing strategy, resources and costs. They have not, however, been configured into a viable topology. Save this RBD file with a new name as to not overwrite the original and construct the RBD based on the system description. Determine a preliminary estimate of the system's availability and cost of the facility over its operating period of one year.

Block Name	Failure Distro	Param1	Param2	Param3	Repair Distro	Param1	Param2	Param3
1553card	Exponential	30.0	0.0		Fixed	0.250		
AC_power	Exponential	30.0	0.0		Lognormal	0.50	0.20	
apple_filler	Exponential	5.0	0.0		Lognormal	0.250	0.10	
batteries	Exponential	10.0	0.0		Fixed	0.50		
boxer	Extreme Value	5.0	30.0		Lognormal	0.50	0.20	
cherry_filler	Exponential	5.0	0.0		Lognormal	0.250	0.10	
computer	Exponential	100.0	0.0		Lognormal	0.50	0.20	
crust_former	Weibull	2.0	10.0	0.0	Extreme Value	0.050	0.250	
crust_former	Weibull	2.0	10.0	0.0	Extreme Value	0.050	0.250	
crust_former	Weibull	2.0	10.0	0.0	Extreme Value	0.050	0.250	
emerg_switch	Exponential	100.0	0.0		Lognormal	0.050	0.050	

Figure 13.2 Block Distribution Input Table

Using the availability obtained, make an estimate of the income generated. Is the facility profitable? Our solution is provided on the next page.

The **PieFacility2.rbd** file contains a fully assembled model of the system that follows all the rules laid out in the system description. For the purposes of completing this exercise in class, we recommend simulating for five trials. For actual analysis, the RBD would be simulated throughout many more trials. We obtained an availability of 0.7938.

Parameter	Minimum	Mean	Maximum	Standard Dev	SEM
Total Cost	\$210,231.38	\$221,028.18	\$228,694.13	\$7,276.67	\$3,254.23
Availability	0.781740615	0.793772828	0.817876150	0.014335750	0.006411142
MTBDE	3.566692	3.719604	3.917911	0.148344	0.066341
MDT	0.800906	0.968132	1.048293	0.096795	0.043288
Reliability	0.000000000	0.000000000	0.000000000	0.000000000	0.000000000
Conditional Reliability	0.000000000	0.000000000	0.000000000	0.000000000	0.000000000
System Failures	74	78.000000	83	3.674235	1.643168

Figure 13.3 Results from Simulating PieFacility2.rbd

In the period of a year, the facility produces:

$$\left(30 \frac{\text{pies}}{\text{hour}}\right) \times \left(24 \frac{\text{hours}}{\text{day}}\right) \times \left(365 \frac{\text{days}}{\text{year}}\right) \times \text{Availability}$$

$$= \left(30 \frac{\text{pies}}{\text{hour}}\right) \times \left(24 \frac{\text{hours}}{\text{day}}\right) \times \left(365 \frac{\text{days}}{\text{year}}\right) \times (0.793772828) = 208,603 \text{ pies}$$

Each pie generates \$2.17 in income, so the facility makes \$452,668.51 (i.e., 208,603 x \$2.17). Comparing this to the operating cost of \$221,028.18, we find that we have \$231,640.33 in profit.

C. For the next exercise, you will employ Raptor to perform a design trade-off analysis. Given below are three alternatives as to how to improve the pie-making facility. You will modify the **PieFacility2.rbd** three distinct times for each of the plans to see which is best.

The chief baker, Kenny, thinks that the biggest bottleneck is that the computer and 1553 card are needed to regulate the power supply. He has identified a power supply replacement, identical from a reliability metric standpoint, which is self-regulating eliminating the need for the computer and 1553 card. Create an RBD to analyze the baker's suggestion.

Plant manager Stan argues that the biggest improvement can be achieved by using redundancy, preventive maintenance and standby to improve the availability. Create a new RBD to investigate the plant manager's idea by identifying the least available component. Introduce a back-up for this component that has the same cost information and repair data and a failure mean that is half of the original unit. Place the back-up in a cold standby configuration with the primary. Next, set up a preventive maintenance schedule that optimizes the availability of the primary, with the assumption that preventive maintenance requires a fixed time of 0.5 days to complete.

Lead accountant Kyle thinks that down time can be reduced by hiring one or more additional plant mechanics to prevent hold time caused by insufficient resources. Create a new RBD to evaluate the accountant's plan.

To carry out the baker's plan, we eliminate the components representing the computer and 1553 card from the **PieFacility2.rbd** and reconnected the RBD. The results from our solution for the baker's alternative are shown in Figure 13.4. We have increased the availability and reduced the overall costs. If these components are truly superfluous, this alternative should be considered.

To implement the plant manager's plan, we first simulate the **PieFacility2.rbd** using weak link analysis. Our results are shown in Figure 13.5. The worst offender is the main motor component. We add an alternate component designated as backup motor by copying the main motor component and then pasting it near the primary. Since this backup motor is not as reliable as the prime, we change its failure distribution as shown in Figure 13.6. We then placed the backup motor in a standby configuration with the main motor, being certain to set the priority to the main motor and to specify priority return. We also set up a preventive maintenance scheme on the main motor with the maintenance activity scheduled to occur every ten days. Simulating this new configuration produces these results shown in Figure 13.7. The plant manager's alternative results in a substantial improvement in the system's availability; it also however, increases the costs.

Block Properties - backup_motor - 39

Costs | Dependency | Simulation Theory | Advanced

Failure & Repair | Maintenance Information | Preventive Maintenance | Resources

Block Name: Comment:

Block Template:

Failure Distribution: Repair Distribution:

Shape: Mean: days

Scale: days Standard Dev: days

Location: days

Mean: 4.431135 Mean: 1.000000
 Stdev: 2.316257 Stdev: 0.500000

Figure 13.6 The Backup Motor's Failure and Repair Attributes

Output Tables

Summary | Logistics | Sparing | Block Costs | Block Analysis | Node Analysis

Results from 5 runs of sim time 365.000000:

Parameter	Minimum	Mean	Maximum	Standard Dev	SEM
Total Cost	\$231,966.79	\$235,584.50	\$245,181.16	\$5,625.35	\$2,515.73
Availability	0.858714821	0.877593962	0.895069459	0.014562388	0.006512498
MTBDE	4.798770	5.378779	5.731585	0.392054	0.175332
MDT	0.671924	0.749964	0.937620	0.107045	0.047872
Reliability	0.000000000	0.000000000	0.000000000	0.000000000	0.000000000
Conditional Reliability	0.000000000	0.000000000	0.000000000	0.000000000	0.000000000
System Failures	55	59.800000	66	4.324350	1.933908

Figure 13.7 Results of the Plant Manager's Alternative

To carry out the idea advanced by accounting, we return to the baseline RBD (i.e., **PieFacility2.rbd**) and increase the number of plant mechanics from two to four. Simulating this alternative yields the results shown in Figure 13.8. This alternative did not significantly change the system's availability and resulted in an increase in the overall costs.

Parameter	Minimum	Mean	Maximum	Standard Dev	SEM
Total Cost	\$211,000.16	\$223,540.96	\$229,815.35	\$7,999.14	\$3,577.32
Availability	0.771228819	0.790220261	0.820615386	0.021857392	0.009774923
MTBDE	3.391548	3.735623	4.082775	0.312531	0.139768
MDT	0.808338	0.992119	1.187654	0.134761	0.060267
Reliability	0.000000000	0.000000000	0.000000000	0.000000000	0.000000000
Conditional Reliability	0.000000000	0.000000000	0.000000000	0.000000000	0.000000000
System Failures	70	77.600000	83	6.107373	2.731300

Figure 13.8 Results of the Accountant's Alternative

The results of the three alternatives and their respective net profits are shown in Figure 13.9.

Option	Availability	Production	Income	Mean Cost	Profit
Baseline	0.793772828	208,603	\$452,668.51	\$221,028.18	\$231,640.33
Kenny's plan	0.805396457	211,658	\$459,297.86	\$206,735.48	\$252,562.38
Stan's scheme	0.877593962	230,632	\$500,471.44	\$234,116.82	\$266,354.62
Kyle's idea	0.790220261	207,670	\$450,643.90	\$223,540.96	\$227,102.94

Figure 13.9 Summary of Results

Many readers may have been troubled by the fact that we have assumed that the plant produces thirty pies per hour when it is running, since each of the three pie-forming lines can make fifteen pies per hour, and we do not always have at least two of them up and running. The answer we calculated is an approximation and may not be very accurate. It begs the question of how can we determine how many pies are actually

being produced? It is possible using the results available in the weak link analysis section of the final results report to extract the necessary information to determine the exact number of pies produced. The final results report contains more detailed weak link analysis information, including the percent of time each line was operating. The relevant section of the **PieFacility2.rbd** final results report is shown in Figure 13.10.

```

***WEAK_LINK_ANALYSIS***
NODE_ANALYSIS
State_Data
Node      %Green  %Yellow  %Red     %Blue   %Brown  %Orange
apple_pie 67.526493 0.000000 8.863858 23.609648 0.000000 0.000000
cherry_pie 64.510826 0.000000 8.299315 27.189859 0.000000 0.000000
n26       100.000000 0.000000 0.000000 0.000000 0.000000 0.000000
n27       97.291742 2.659905 0.048353 0.000000 0.000000 0.000000
n34       82.530241 0.000000 17.469759 0.000000 0.000000 0.000000
pies      78.342244 21.193105 0.464652 0.000000 0.000000 0.000000
rhubarb_pie 63.742524 0.000000 8.714739 27.542736 0.000000 0.000000

```

Figure 13.10 Node Availability Analysis Sections of Final Results Report

Unfortunately, because of the manner in which we set up this RBD, the %Green values for the pie lines include time when the lines were up but some other component in the system was down that restricted us from making pies. Had we used dependence to shut down the pie machines when the system is down, we would have idle time showing for the nodes when they are up, but another component is down. In this case, we could multiply the %Green time for each line by 15 pies per hour to obtain an accurate result. There is a much easier way by using Raptor’s capacity analysis feature, which was designed for exactly this type of analysis. Chapter 14 discusses how the capacity features are implemented.

Problems

1. What was ambiguous in the system description of the pie-making facility?
2. If we wanted to know how many pies of each type were produced, how could we determine that?
3. What other assumptions did we make with our model? In our capacity calculation, did we really eliminate all assumptions?
4. Were you surprised at how similar the availabilities were for the apple_pie, cherry_pie, and rhubarb_pie nodes? To what do you attribute this?

Chapter 14



Objectives

1. Understand how to use the Capacity Analysis feature of Raptor.
2. Understand the difference between throughput and capacity.
3. Be able to determine if an element is underutilized.

Recommended Problems

2, 3, 4, 5, 6, 7, 8 and 9

Capacity Analysis

Some systems perform identically in a yellow state as they do in a green state, as long as a minimum set of elements is functioning. Other systems perform in a degraded manner when failures occur. One of the main reasons that the capacity analysis feature was added to Raptor was to help quantify the degree of degradation a system is experiencing as elements fail and repair. When a system performs in a degraded manner, this may indicate that it is producing fewer products, generating less power or possibly handling fewer messages. It is imperative to understand that capacity analysis is the realization that some generic substance, defined by the user, is flowing through a system and if failures occur, that flow is interrupted or diminished. The concepts of "flow" and "capacity" are two fundamental underpinnings of Raptor's capacity analysis feature. These terms are defined as follows:

Flow is the amount of substance per unit time that is being processed through a particular path or a system.

$$\text{Flow} = \frac{\text{substance}}{\text{time}} \triangleright \frac{\text{gallons}}{\text{minute}} \triangleright \frac{\text{cars}}{\text{hour}} \triangleright \frac{\text{chlorine atoms}}{\text{seconds}} \triangleright \frac{\text{messages}}{\text{nano - seconds}}$$

Capacity is the maximum amount of flow a path or system can manage per unit time.

$$\text{Capacity} = \max(\text{Flow}) \triangleright \max\left(\frac{\text{gallons}}{\text{minute}}\right) \triangleright \max\left(\frac{\text{cars}}{\text{hour}}\right) \triangleright \max\left(\frac{\text{messages}}{\text{second}}\right)$$

The reader should take note that the word "throughput" is synonymous with the word "flow" and both will be used throughout this chapter. The concept of capacity analysis is comprehended best by reviewing and understanding some examples. Open the **Capacity1.rbd** file and open the System Settings dialog box from the *Options – System Settings* menu item. Modify the two capacity fields of this dialog box as shown in Figure 14.1. For this model our capacity concerns are related to how much electrical power can be generated by this system. Electricity is needed to perform daily functions such as maintaining attitude control, providing heat and light, running the electronics of the system and providing additional power for onboard experiments. Every capacity analysis model starts with determining how much substance will be allowed to flow through a system. For this model, the electrical capacity, or maximum achievable flow, of this system is 2,050 Watts. This is merely the sum of three power strings' capacity of 500, 650 and 900 Watts of power. The cost incurred for the inability to produce electricity has been established to be \$6.48 per unit of lost flow for each unit of down time this system experiences. This cost field makes it possible to model the forfeiture of money or opportunities whenever flow is interrupted such that a system cannot process some portion of the flow generated value.

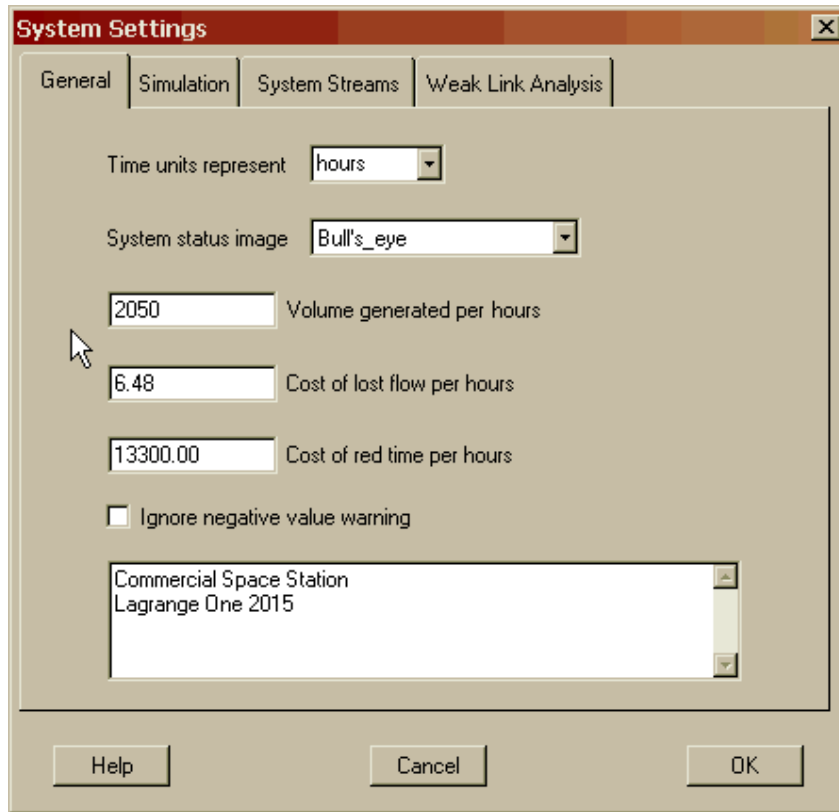


Figure 14.1 Setting the Flow Generation and Cost of Lost Flow

To facilitate the understanding of capacity analysis, imagine that this system generates 2,050 Watts of electricity at the Start marker and then flows through its different paths. This substance is generic in nature and can represent any non-varying discrete entity like messages, cars, chemicals, liquids, manufactured products or simply, the generic material widely known as widgets. In this version of the Raptor tool, only one type of substance can be generated for a given simulation.

Occasionally, all of the flow will reach the End marker. Other times, only a fraction of the flow reaches the End marker due to components being in one of their many down states. Thus, components in a down state restrict or obstruct the flow generated from the Start marker. While the flow generation can be viewed as if it starts from the Start marker, a question arises as to how Raptor determines which path the flow should take when presented with multiple choices. Is the flow sent down one preferred path or split equally (i.e., without fractionalizing any integer flow) among all possible paths? This question can be answered yes for both situations depending on how users specify the partitioning of the flow.

We have already established in Chapters 7, 8 and 11 (i.e., dependence, standby and weak link analysis, respectively) that nodes occupy a unique position in an RBD's topology. They exist at the intersection of paths and thus are a natural choice for controlling or distributing the flow between the Start and End markers. All nodes have the ability to house the flow-in as well as the flow-out capacity information. Thus,

nodes control the amount of flow that can traverse through its inbound paths and determine the direction to send its outbound flow. Nodes default in essence to being infinitely large pipes and thus can handle any amount of flow either into or out of them. Once you place any restriction on the infinite flow either in or out of a node, that node becomes a “capacity node”. Only these nodes will display capacity information within the Simulation View and have their results reported in the Output Tables dialog box. Let us create a capacity node by placing flow restrictions on the Power node. Open the Node Properties dialog box for the Power node and select the Capacity tab. Figure 14.2 displays both the “Unrestricted flow” and “Use any available path” checkboxes as being marked, which indicates that this node acts as an infinitely large pipe.

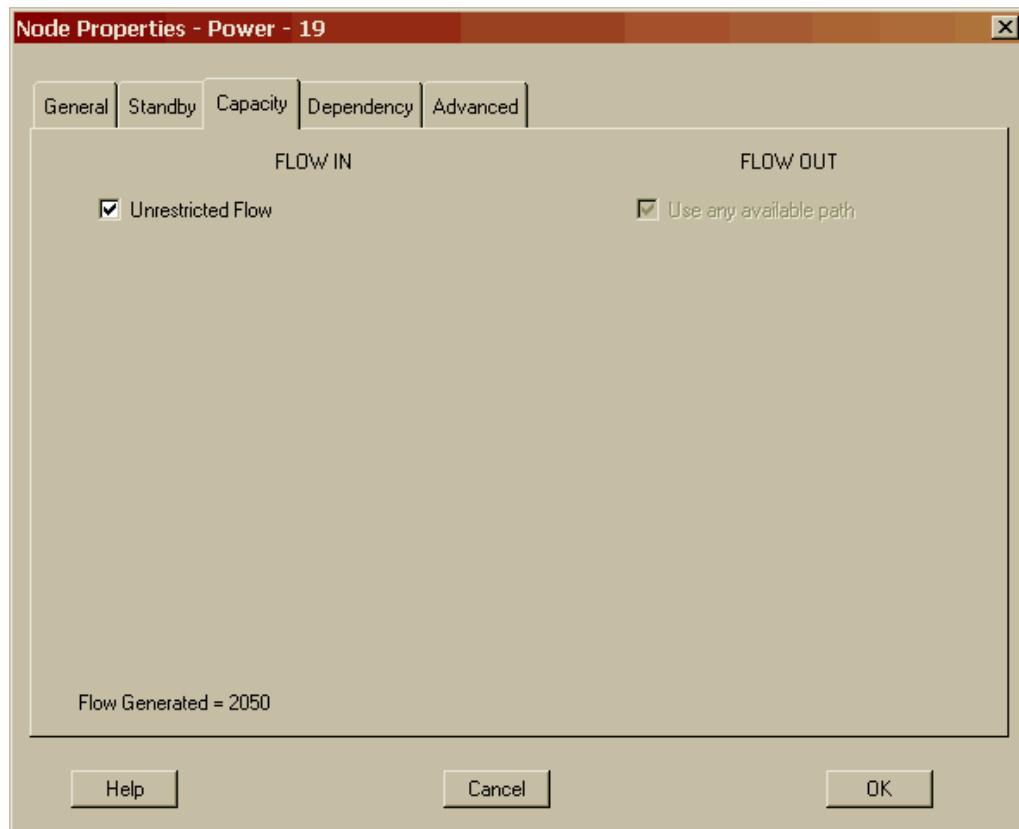


Figure 14.2 Power Node’s Capacity Tab with Unrestricted Flow

The use any path checkbox is deactivated since only one path departs from the Power node and all flow must use that single path. When the unrestricted flow checkbox is marked, all paths coming into this node have no upper limit on the amount of flow that can be channeled through them. When not marked, the capacity flow can be modified within the flow-in table as shown in Figure 14.3.

Uncheck the unrestricted flow checkbox and specify a nominal flow that the path carries under normal operating conditions and a maximum flow, or as we defined earlier, its capacity. Click on the cell in the Max Flow column for the link coming

from the 900path node to display a flow value box. Enter a quantity of 900 in the value box and click on the Accept Changes button or select the cell in which you want the value to appear. Click on the cell in the Nominal Flow column for the same path and note that a wattage value of 900 will appear in that cell as well. Continue editing the dialog box until it mimics those values shown in Figure 14.3.

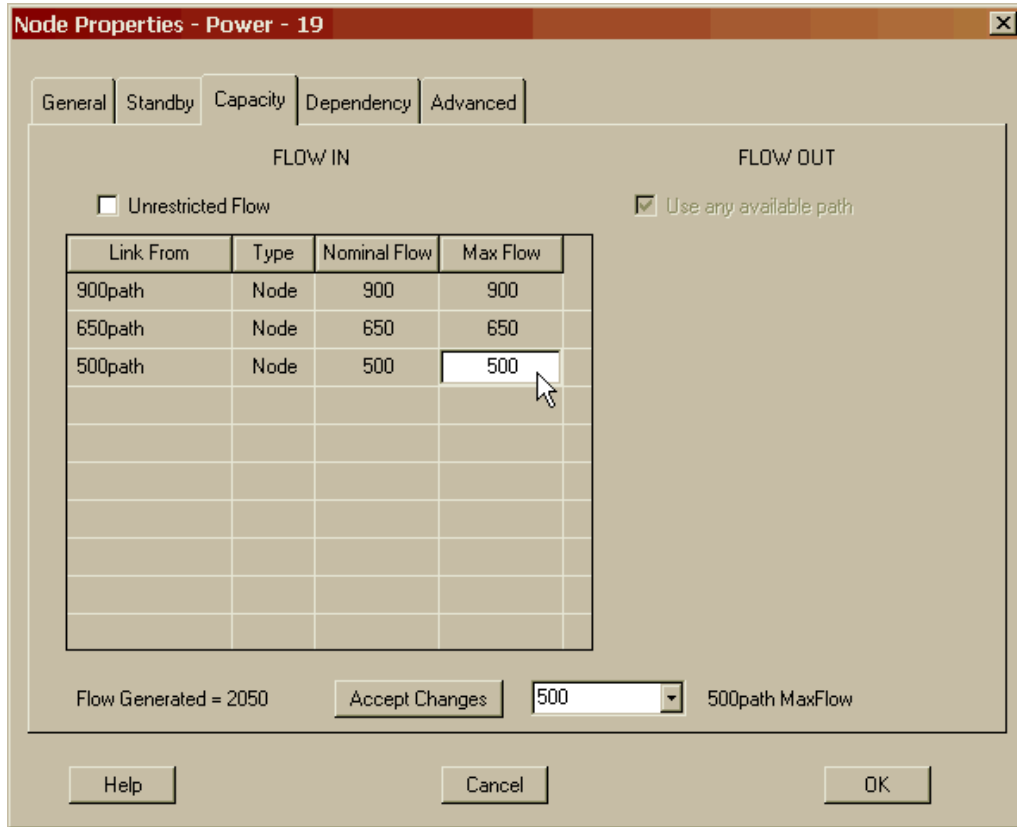


Figure 14.3 Power Node’s Capacity Tab with Restricted Flow

We have just specified that the 900path node, which controls the 900 Watt solar panels, the sodium-sulfate battery and the fully-regulated bus components, can nominally manage 900 Watts of energy and has no additional surge capacity since the maximum flow value is equivalent to the nominal value. Likewise, the 650path and 500path nodes can handle 650 and 500 Watts of energy, respectively. Recall that we are generating 2,050 Watts and we have just defined how much each power string path can handle.

Open the Node Properties dialog box for the n1 node and select the Capacity tab. Uncheck the “Use any available path” checkbox and modify the table as shown in Figure 14.4. Since this node has multiple outbound paths we can define a priority relationship amongst these paths. Outbound flow will be sent down the path with the highest priority first. A lower number represents a higher priority. The highest priority value for an outbound path is one and the lowest priority is ninety-nine. Flow will be sent to the highest priority path until that path reaches its nominal flow value.

Additional flow is then sent to the path with the next highest priority. Flow will be distributed evenly among paths with equal priority. If all paths are handling their nominal flow amounts, and excess flow needs to be distributed, flow will be added to the highest priority path until it reaches its maximum flow value. This continues until all paths are flowing at full capacity. Any additional flow is rejected but may be redirected to other paths that might be able to handle the surplus flow. Given that we are generating as much flow as this system can handle, the priorities set in this node will not affect the capacity output results. Later in the chapter we will conduct a less than maximum capacity simulation and these priorities will then become more relevant.

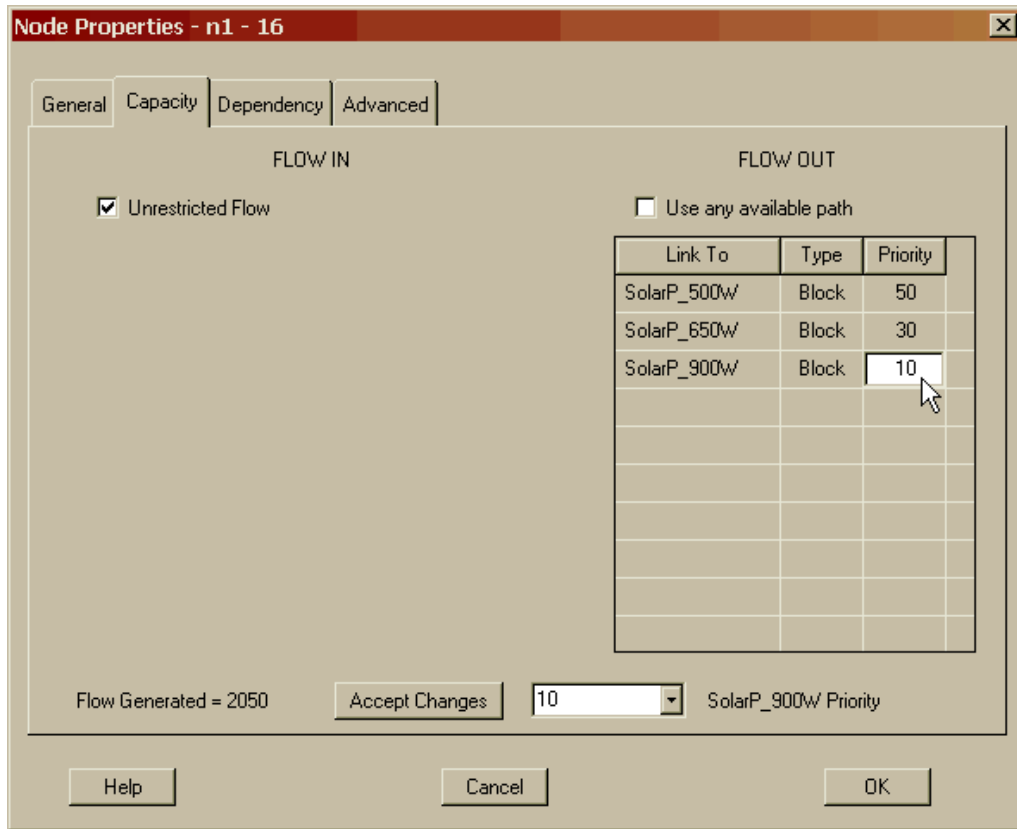


Figure 14.4 Capacity with Prioritized Paths

Return to the Workspace View by selecting the OK button of this dialog box and notice that the capacity analysis and cost analysis toolbar buttons for this model are depressed (see Figure 14.5), which indicates that both capacity and cost analysis will be performed whenever a simulation of this system is initiated.

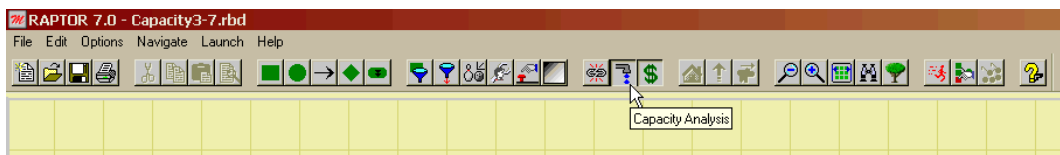


Figure 14.5 Workspace View Capacity and Cost Analysis Toolbar Buttons Engaged

Start the simulation of this model in step mode. Figure 14.6 displays the initial flow conditions, which includes only the two nodes defined to be capacity nodes and the Start marker. Displayed above the Start marker will be the system flow value, which is currently 2,050 Watts. The Power and n1 nodes display the value of 2,050 as well, which is the current amount of flow passing through them.

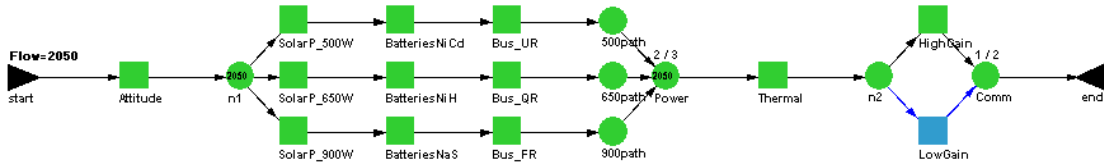


Figure 14.6 Initial Flow Conditions

Click the Step button once and the high gain antenna component will fail as shown in Figure 14.7. This same failure would have occurred even if capacity analysis was not engaged since capacity analysis is an additional layer of code that does not affect the RAM aspects of a simulation (similar to weak link and cost analysis). When this failure occurs, and because the switchover is not immediate, the system goes down causing the flow through the system to drop to zero. The current flow of zero is now displayed above the Start marker.

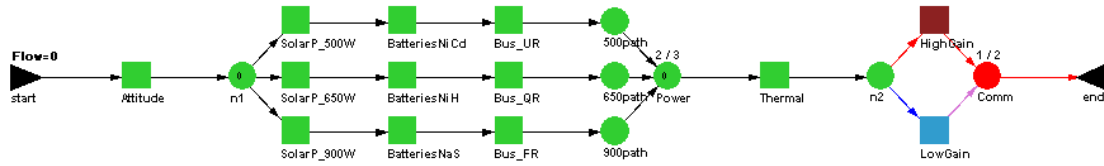


Figure 14.7 Flow Loss Due to a System Failure

Take three more steps and note in Figure 14.8 that only 1,550 Watts are flowing since preventive maintenance for the 500 Watt solar panel and nickel-cadmium batteries has been triggered. With the 500path power string down, 500 Watts of capacity has been lost and will remain lost until this power sting starts functioning again. The amount of time and the flow during a time interval is maintained by Raptor so that a time-weighted average flow can be calculated.

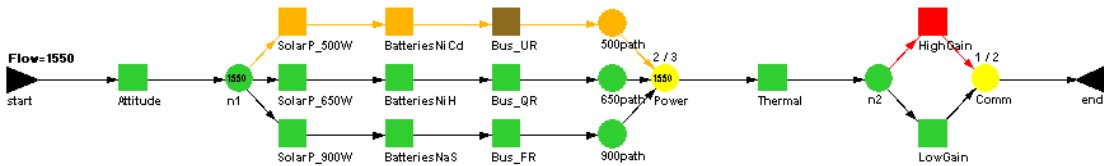


Figure 14.8 Partial Flow Loss of 500 Watts

Take three more steps and note in Figure 14.9 that only 1,400 Watts are flowing since preventive maintenance for the 650 Watt solar panel and nickel-hydrogen batteries has

been triggered. With the 650path power string down, 650 Watts of capacity has been lost and will remain lost until this power string starts functioning again.

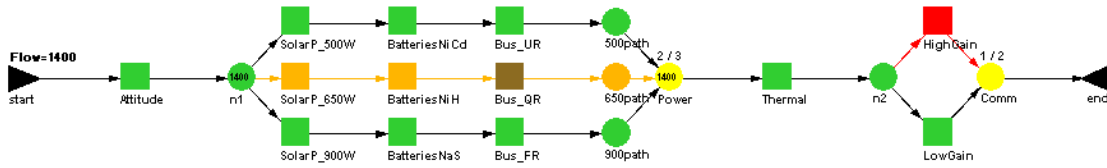


Figure 14.9 Partial Flow Loss of 650 Watts

Once again click the Step button three more times and note in Figure 14.10 that only 1,150 Watts are flowing since preventive maintenance for the 900 Watt solar panel and sodium-sulfate batteries has been triggered. With the 900path power string down, 900 Watts of capacity has been lost and will remain lost until this power string starts functioning again.

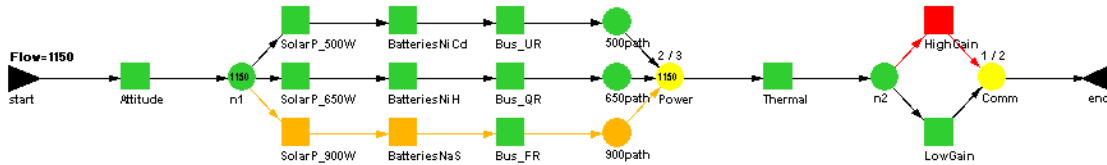


Figure 14.10 Partial Flow Loss of 900 Watts

Click on the depressed Pause button and allow the simulation to run to its completion. Open the Output Tables dialog box and note the new capacity information now available. As indicated in Figure 14.11, the system availability value is 0.950, which is unchanged from the previous chapters (i.e., Figures 11.18 and 12.11), as we should expect. In fact, all of the values in the summary table from the availability row and below are unchanged.

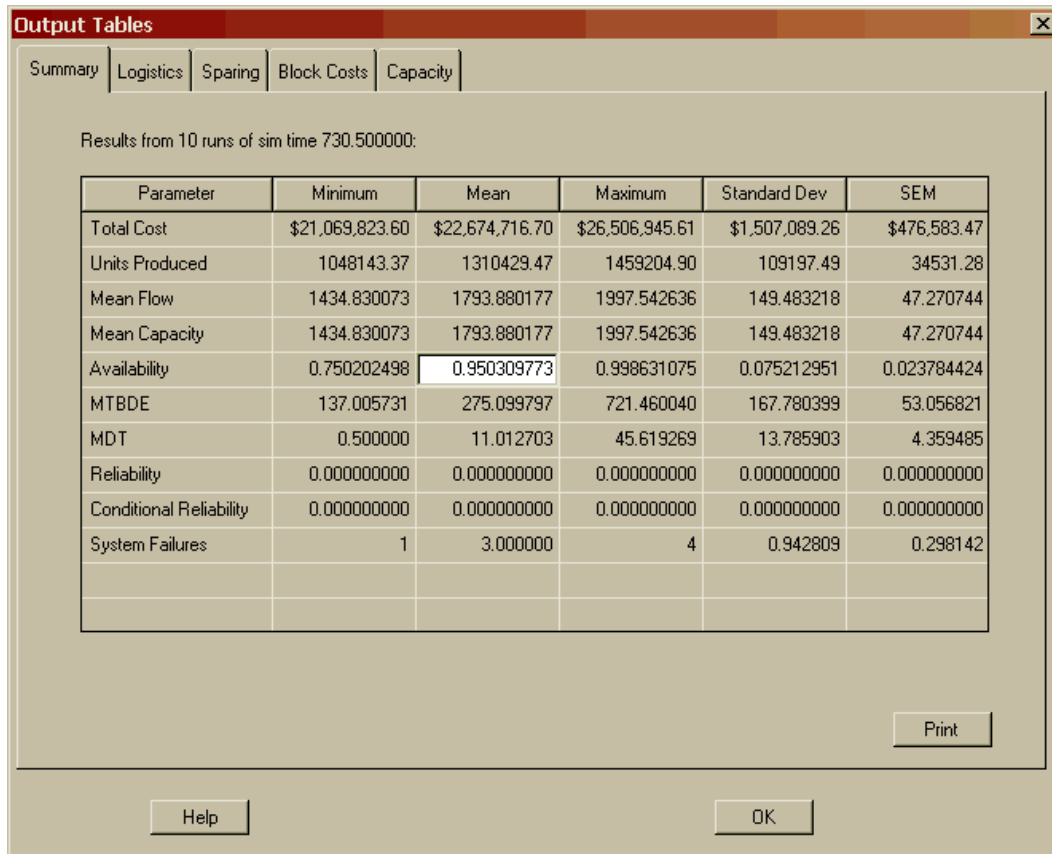


Figure 14.11 Summary Results of the first Capacity Simulation

Three additional rows have been added to the summary tab (i.e., mean capacity, mean flow, and units produced) that only appear when capacity analysis is conducted. During any individual trial, the capacity of a system is likely to fluctuate. After each trial, the mean capacity is calculated as the time weighted average of the capacity values observed. The mean capacity row displays the minimum mean capacity observed during all trials as well as the maximum. For this simulation, the flow generated equals the system's maximum capacity, so the mean flow row matches the mean capacity row. If less flow is generated than a system's capacity, these values will not match. For each run, once the mean flow rate is calculated, the total units produced can be determined as the product of the mean flow rate and the simulation time. Again, the minimum, average, maximum, standard deviation and standard error of the mean for each parameter are displayed.

In chapter twelve, the cost for operating this RBD for a month averaged \$21.462 million. Notice that the cost to operate the very same system in this chapter is \$22.675 million. The difference in cost is caused by assessing the cost of lost flow at a rate of \$6.48 per hour per unit of flow. This system produced more than 1.3 million units (Watts) and had a capacity of nearly 1,794 Watts. Hence, about 256 Watts per hour are lost due to components being in down states. Put another way, this system has the theoretical ability to produce 2,050 Watts but in reality on average it can produce only 1,794 Watts. Click on the Capacity tab as indicated in Figure 14.12.

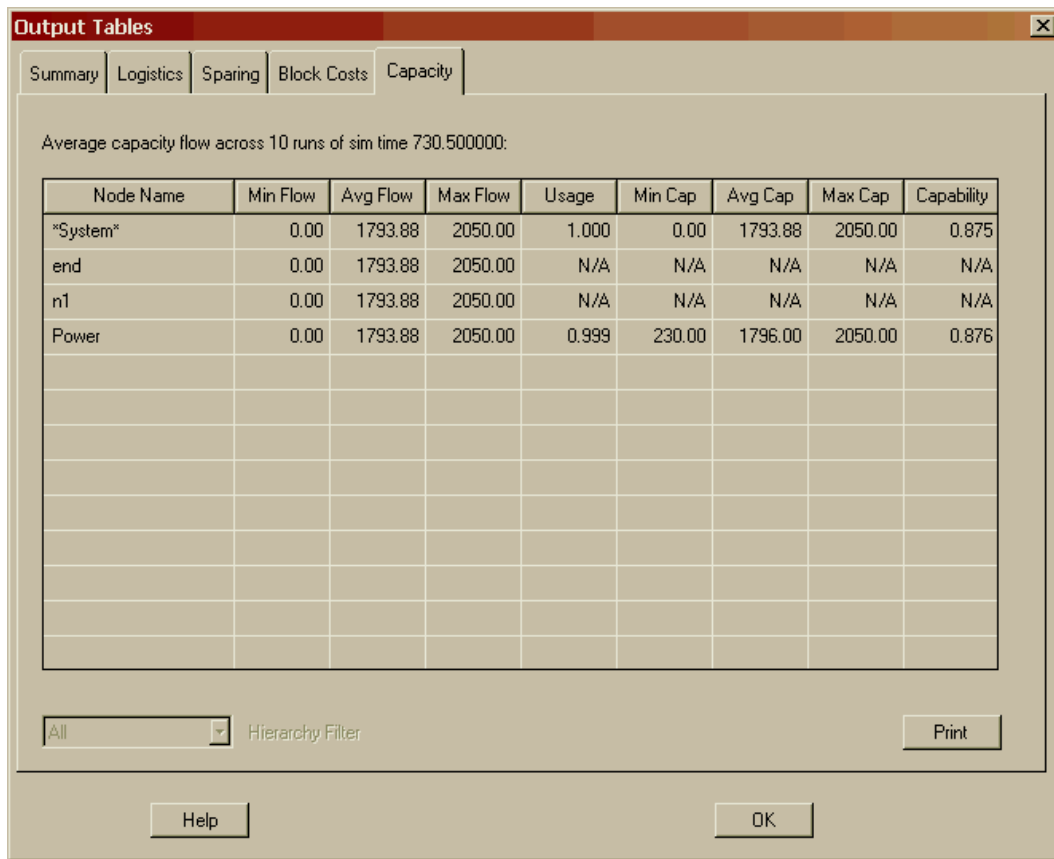


Figure 14.12 Detailed Capacity Results by Nodes

Only capacity nodes will be displayed in this table plus a row for the End marker and system will be included. The first row always displays the system-level capacity results and provides more detail than contained on the Summary tab. The first column is the minimum flow, which is the average minimum flow value observed across all trials. In the first trial, we observed a system failure that caused the flow to drop to zero. Since the average minimum flow is also zero, this must have occurred in all of the other trials as well. The second column is the average of the average flow from each of the ten trials. The third column is the average of the maximum flow observed from each of the ten trials.

The minimum capacity column is the average of the minimum capacity values observed across all trials. The average capacity and maximum capacity are the averages of the average capacity and maximum capacity from across all trials, respectively. The remaining two parameters are functions of the flow and capacity results. First, the usage rate is an indicator of how much of a system or path's capacity is being utilized. It is calculated as follows:

$$Usage\ rate = \frac{Average\ Flow}{Average\ Capacity}$$

In this simulation, the system was producing as much flow as possible, so the usage rate for the system was 1.0. The Power node was utilized 99.9% of the simulation. Small usage rates may indicate excess equipment that may not be needed. High usage rates may indicate that a path or system has no surge capacity if pressed upon to produce more product. The second parameter, known as capability, is only a function of the capacity results. The capability parameter is a measure of how well a system maintains its capacity. It is calculated using the equation below:

$$\textit{Capability} = \frac{\textit{Average Capacity}}{\textit{Maximum Capacity}}$$

On average, the system's capacity is 87.5% of what it is capable of when operating at peak capability. The flow and capacity values for the nodes are not equivalent since there are times when paths are capable of handling flow but are not due to failures elsewhere in the system that is restricting total flow. Notice that for some values of the table, the not applicable symbol is displayed. This occurs for nodes that have their unrestricted flow marked since their capacity is not defined when they can handle an infinite amount of flow.

A plot of the capacity of the system versus simulation time can be created by first requesting that the capacity graph be generated. Before a simulation begins, a user can check the Capacity checkbox on the Files & Reports tab of the Simulation Options dialog box as shown in Figure 14.13 to generate the capacity plot data. This file is relatively easy to read into a spreadsheet type tool to create a capacity plot. The capacity for the first trial is shown in Figure 14.14 and indicates that three times the system was unable to produce any flow and hence three system failures occurred during this trial. If the area under the capacity curve for each trial was calculated and divided by the simulation time (730.5 hours) then the time-weighted capacity would be 1793.88 Watts, which would match the system-level capacity reported in Figure 14.11, as it should.

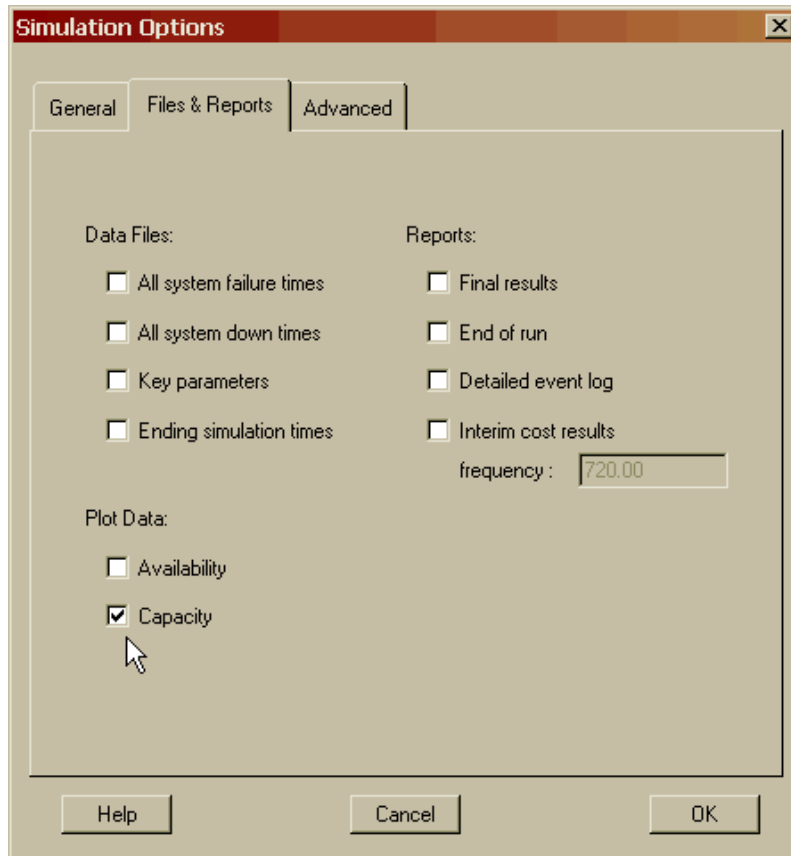


Figure 14.13 Generating the Capacity Plot Data File

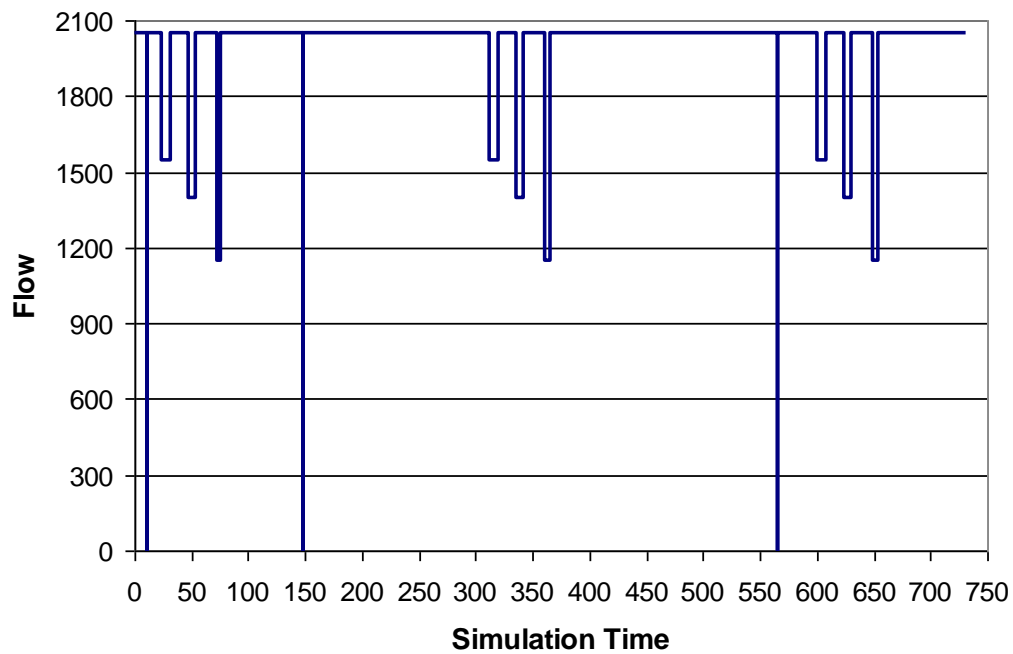


Figure 14.14 Capacity Plot of First Trial of Simulation

Let us now look at a case where the flow generated is less than the system's capacity. When this situation occurs, the system can tolerate some failures and still meet flow generation requirements. Open **Capacity2.rbd** and then modify the System Settings dialog box to generate a flow of only 1,500 Watts as indicated in Figure 14.15.

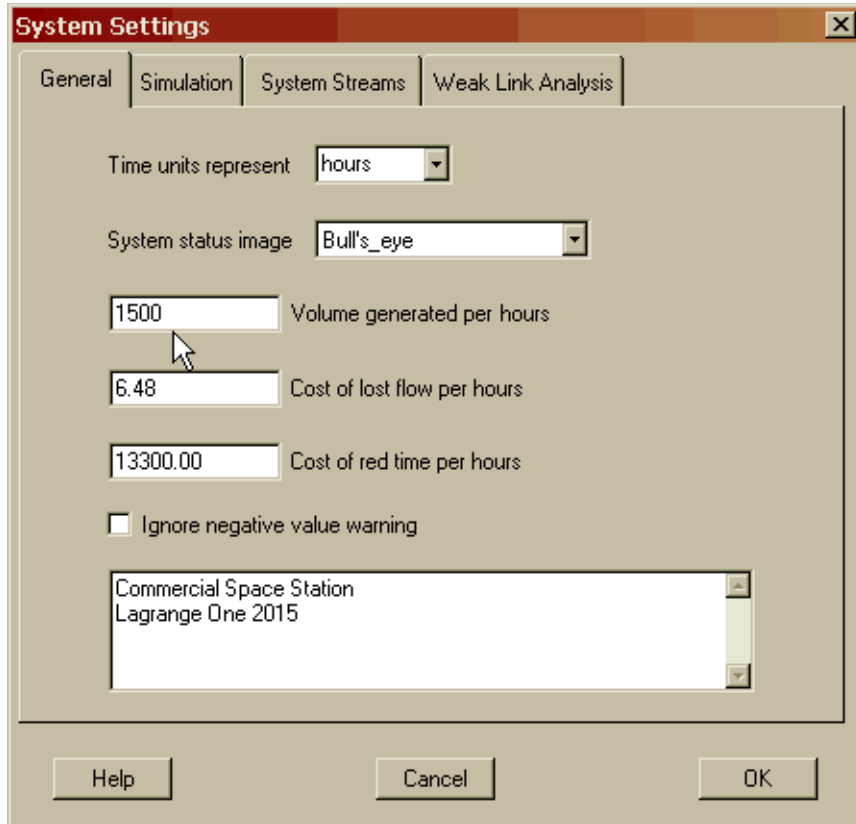


Figure 14.15 Flow Generated Less Than System's Capacity

Although this system is capable of producing 2,050 Watts, we only need 1,500 Watts to maintain the essential functions of the space station. Any additional electricity will be used for onboard experimentation as it becomes available. Let us further specify that under nominal conditions, the three strings that comprise the power subsystem can each process 500 Watts of throughput. Open the Node Properties dialog box for the Power node, select its Capacity tab and modify its attributes to appear as shown in Figure 14.16. Thus, the 650 Watt string nominally processes 500 Watts but can surge an additional 150 Watts to 650 Watts if called upon by the system. Make a similar change to the 900path node as shown in Figure 14.16. Its nominal flow should be set to 500 Watts and its maximum flow should be set to 900 Watts. Changes to the 500path node are not required since this path's maximum flow is 500 Watts. Hence, the 500 Watt string does not have any reserve capacity since its nominal and maximum flows are identical.

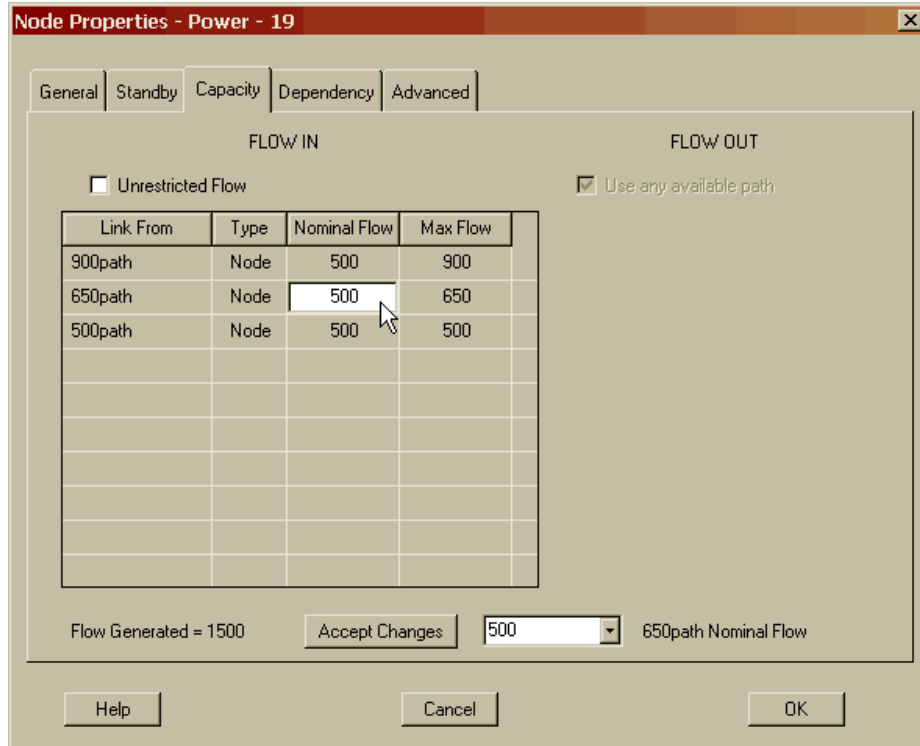


Figure 14.16 Flow for the 650path and 900path Nodes

Select the *Options – Tables – View Node Inputs* menu item. The Capacity tab of the Node Input Tables dialog box should appear as displayed in Figure 14.17.

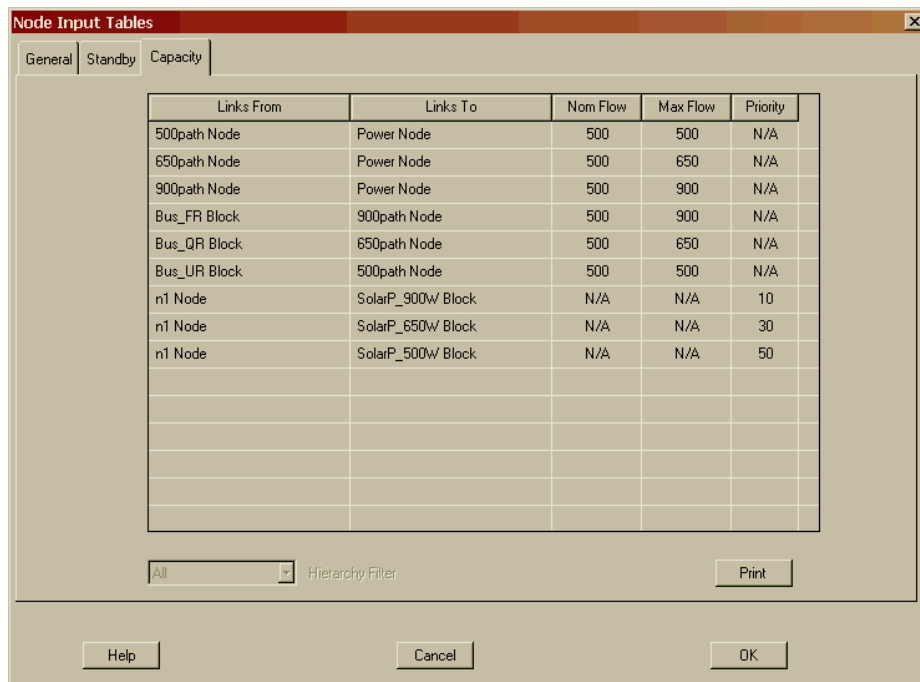


Figure 14.17 Capacity Tab of the Node Input Tables Dialog Box

Simulate this RBD starting in step mode and note the initial flow conditions as shown in Figure 14.18.

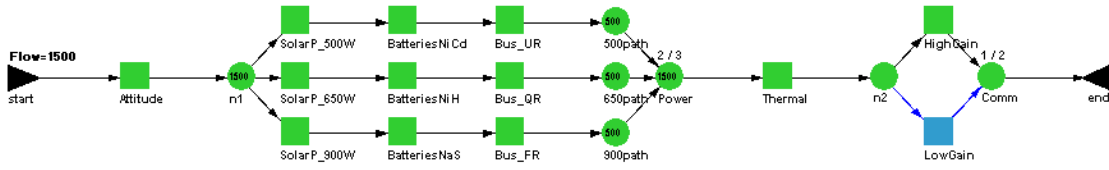


Figure 14.18 Initial Conditions with Excess Capacity

The Start marker indicates 1,500 Watts of flow is being generated and each of the three strings in the power subsystem is handling an equal share of 500 Watts. Take four steps and as shown in Figure 14.19, the 500path power string is down due to preventive maintenance. The 900path has accepted 400 Watts of the 500path's flow and the 650path has taken the 100 Watts. The 900path node receives the lion's share of the excess since its priority is higher (recall Figure 14.4). Even though the 500path power string has lost flow, there is sufficient reserve capacity in the remaining two operating power strings to ensure no system loss of flow.

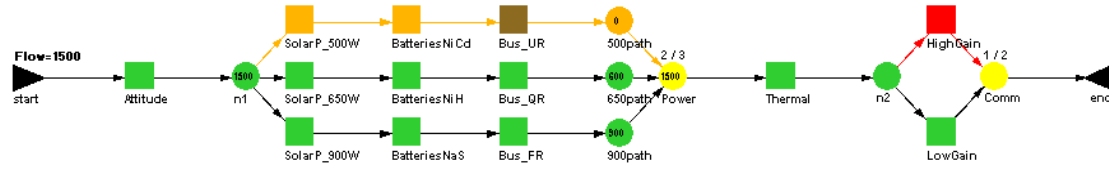


Figure 14.19 Flow Redirection to the 650path and 900path Nodes

Take three more steps and note that the 650path power string is down due to preventive maintenance (Figure 14.20). The 900path has accepted 400 Watts of the 650path's flow but the 500path was unable to accept the remaining 100 Watts since it possesses no reserve capacity (i.e., its nominal and maximum flows are identical). Thus, the loss of 500 Watts of flow from the 650path power string only results in a system loss of 100 Watts due to the 900path power string's ability to handle surplus capacity.

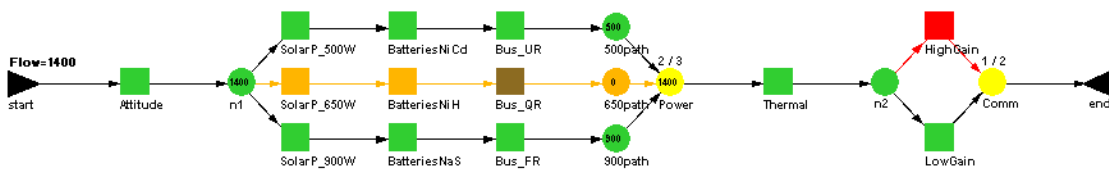


Figure 14.20 Flow Redirection to the 900path Node

Take three more steps and note that the 900path power string is down due to preventive maintenance as shown in Figure 14.21. The 650path has accepted 150 Watts of the 900path's flow but again the 500path was unable to accept the remaining 350 Watts since it possesses no reserve capacity. Hence, the loss of 500 Watts of flow

from the 900path power string only results in a system loss of 350 Watts due to the 650path power string's reserve capacity.

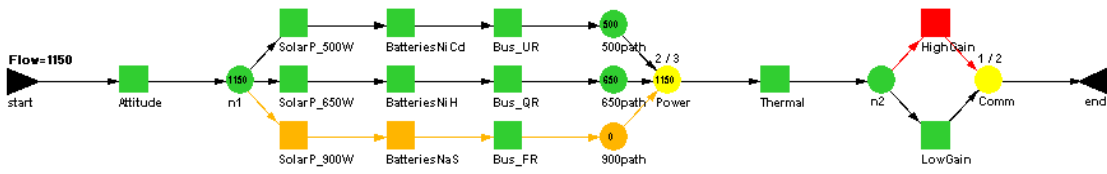


Figure 14.21 Flow Redirection to the 650path Node

Click on the depressed Pause button to allow the simulation to complete. As indicated in Figure 14.22, the mean capacity no longer equals the mean flow. This system has the ability to handle an average flow of 1,794 Watts, but is only producing 1,397 Watts.

Output Tables

Summary | Logistics | Sparing | Block Costs | Capacity

Results from 10 runs of sim time 730.500000:

Parameter	Minimum	Mean	Maximum	Standard Dev	SEM
Total Cost	\$20,875,356.28	\$21,950,812.79	\$25,408,634.23	\$1,310,460.19	\$414,403.90
Units Produced	815860.87	1020368.03	1087440.29	76870.14	24308.47
Mean Flow	1116.852658	1396.807712	1488.624620	105.229480	33.276483
Mean Capacity	1434.830073	1793.880177	1997.542636	149.483218	47.270744
Availability	0.750202498	0.950309773	0.998631075	0.075212951	0.023784424
MTBDE	137.005731	275.099797	721.460040	167.780399	53.056821
MDT	0.500000	11.012703	45.619269	13.785903	4.359485
Reliability	0.000000000	0.000000000	0.000000000	0.000000000	0.000000000
Conditional Reliability	0.000000000	0.000000000	0.000000000	0.000000000	0.000000000
System Failures	1	3.000000	4	0.942809	0.298142

Print

Help OK

Figure 14.22 Summary Results for Excess Capacity System

The value of 1,794 Watts should not surprise us since that is the quantity we obtained in the previous simulation where we determined that this is the maximum flow that this system can process. The 1,397 Watts of mean flow indicates that the system does not have enough reserve capacity to consistently maintain the desired level of 1,500 Watts. If the requirement of 1,500 Watts is the minimum needed for daily operations, a RAM versus capacity versus cost tradeoff study would have to be conducted to

achieve all the aforementioned goals simultaneously. We have reserved this task for the reader as a problem at the end of this chapter.

The total cost of this model is significantly less than the previous model by more than seven-hundred and twenty-three thousand dollars since fewer lost flow charges are assessed. The lower throughput requirement of this simulation allowed excess capacity to compensate for component failures and thus provided more consistent power output at a much lower cost.

Select the Capacity tab of this dialog box as shown in Figure 14.23 and note the usage rates for the various nodes. The 500path power string was used almost to its fullest ability with a usage rate of 98%. In contrast, the 900path power string was only utilized to 62% of its maximum ability. These results should not be surprising since the 900path power string possesses significant reserve capacity.

Average capacity flow across 10 runs of sim time 730.500000:

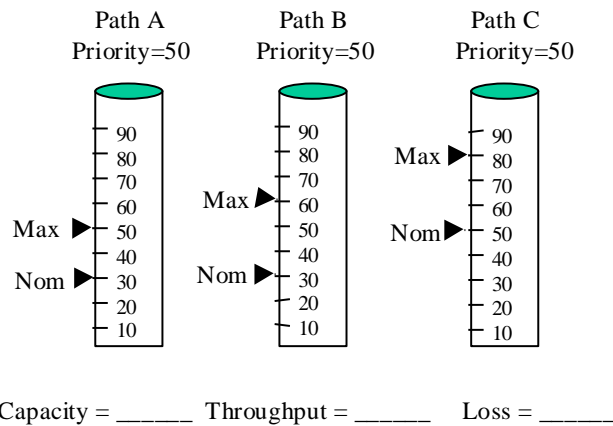
Node Name	Min Flow	Avg Flow	Max Flow	Usage	Min Cap	Avg Cap	Max Cap	Capability
System	0.00	1396.81	1500.00	0.779	0.00	1793.88	2050.00	0.875
500path	0.00	421.08	500.00	0.980	0.00	428.75	500.00	0.858
650path	0.00	461.64	650.00	0.802	0.00	575.88	650.00	0.886
900path	0.00	514.08	900.00	0.620	0.00	828.85	900.00	0.921
end	0.00	1396.81	1500.00	N/A	N/A	N/A	N/A	N/A
n1	0.00	1396.81	1500.00	N/A	N/A	N/A	N/A	N/A
Power	0.00	1396.81	1500.00	0.778	230.00	1796.00	2050.00	0.876

At the bottom of the dialog box, there is a 'Hierarchy Filter' dropdown menu set to 'All' and a 'Print' button. At the very bottom, there are 'Help' and 'OK' buttons.

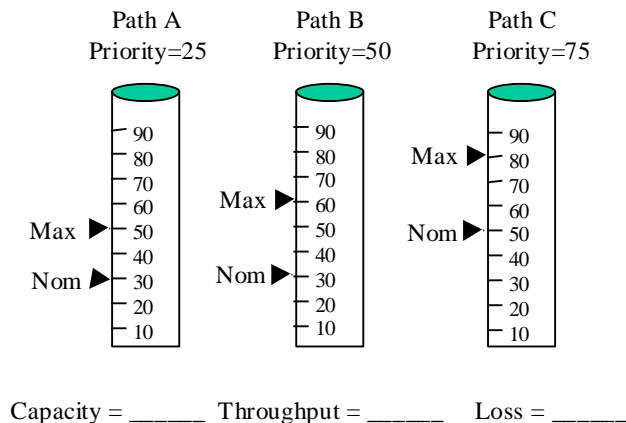
Figure 14.23 Capacity Tab of the Output Tables for Excess Capacity System

Problems

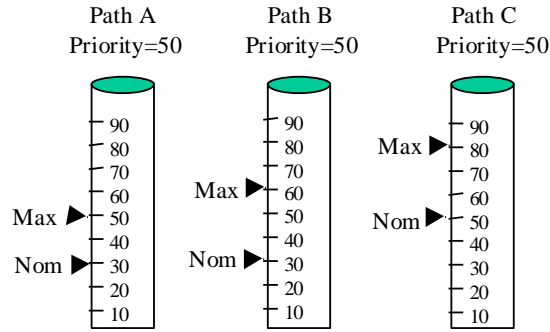
1. What is the difference between throughput and capacity?
2. What analysis features of Raptor must be activated to obtain the cost of lost system flow?
3. A system has a flow of ten units per hour for two hours, and then forty units per hour for an hour. Calculate the total units produced and the mean flow value.
4. Fill in the columns to show the proper distribution of flow if the flow generated is sixty. Calculate the capacity, throughput and loss.



5. Fill in the columns to show the proper distribution of flow if the flow generated is sixty. Calculate the capacity, throughput and loss.

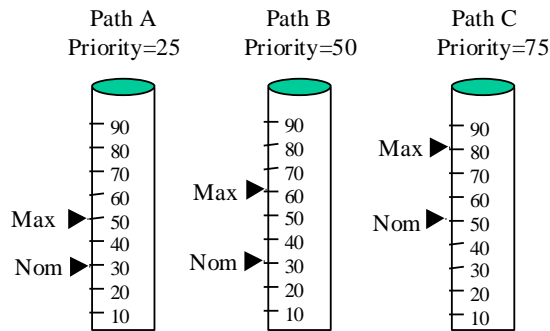


6. Fill in the columns to show the proper distribution of flow if the flow generated is 140. Calculate the capacity, throughput and loss.



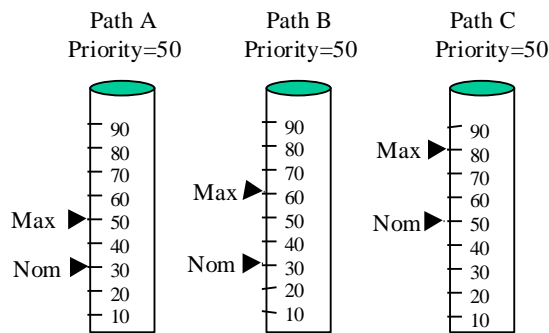
Capacity = _____ Throughput = _____ Loss = _____

7. Fill in the columns to show the proper distribution of flow if the flow generated is 140. Calculate the capacity, throughput and loss.



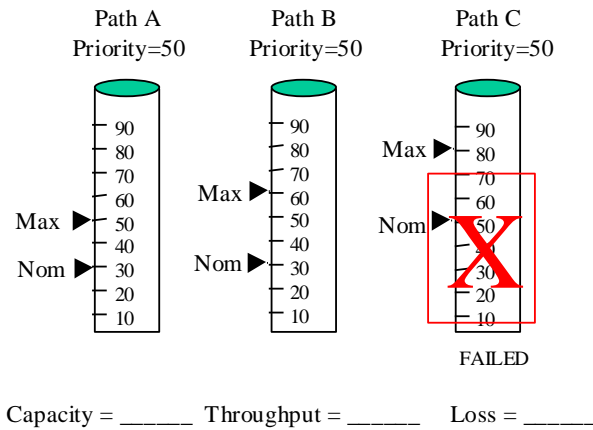
Capacity = _____ Throughput = _____ Loss = _____

8. Fill in the columns to show the proper distribution of flow if the flow generated is 200. Calculate the capacity, throughput and loss.

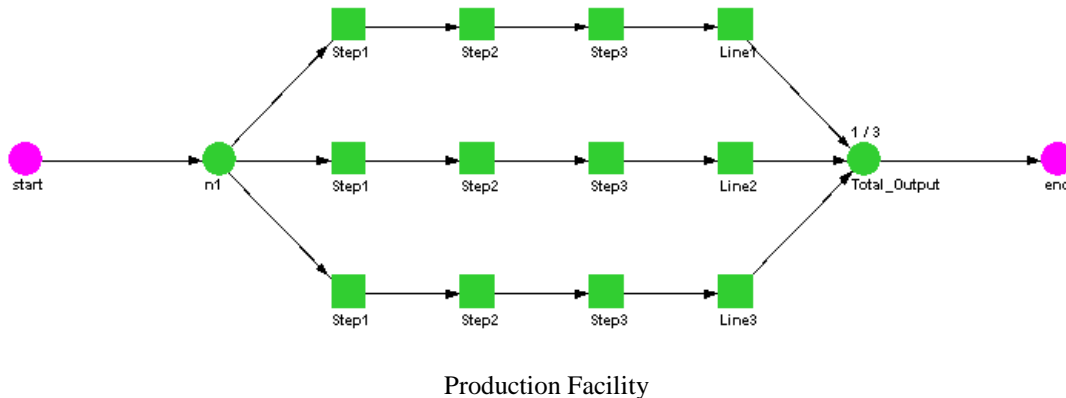


Capacity = _____ Throughput = _____ Loss = _____

9. Fill in the columns to show the proper distribution of flow if the flow generated is 200. Calculate the capacity, throughput and loss. Path C is not available to accept flow.



10. The production facility shown below has three production lines, each of which produces 100 widgets per hour. When all three are running, the factory produces 300 widgets per hour. Each component is independent, fails exponentially (mean of 100 hours) and repairs lognormally (mean of ten hours and a standard deviation of two hours). Determine the average amount of widgets per hour that the system produces.



11. Modify the **Capacity3.rbd** model by adding similar redundant power strings to achieve an availability of 0.960, a mean flow of 1,450 Watts and a first month cost less than 28 million dollars.

12. How can a node be specified to be a choke point or a flow restrictor?

Chapter 15



Objectives

1. Understand when and why you would use an event element.
2. Be able to simulate constant reliability models.
3. Understand the meaning of a “complex” RBD.

Recommended Problems

2, 4 and 6

Events

Often there are situations in the building of an RBD where you need to represent an instantaneously occurring binomial event. A binomial event is discrete, and can be in one of two states: “Success state” or “Failed state”. In the world of reliability simulations, there are traditionally two types of models that require binomial events. The first type is constant reliability models, which are elaborated upon in this chapter. The latter type is a phased-RBD model that will be discussed in Chapter 17. Raptor allows representation of a binomial event by using a diamond-shaped element known as an “Event”.

We want to complete the building of the RBD file labeled **Events1.rbd** in such a manner that upon completion of this chapter it looks like the RBD in Figure 15.1. Let each event represent a piece of equipment of a system that can be modeled with a constant reliability. Thus, we will assume that the system components do not change their reliability significantly during the system’s operations (at least during the period of operations with which we are concerned).

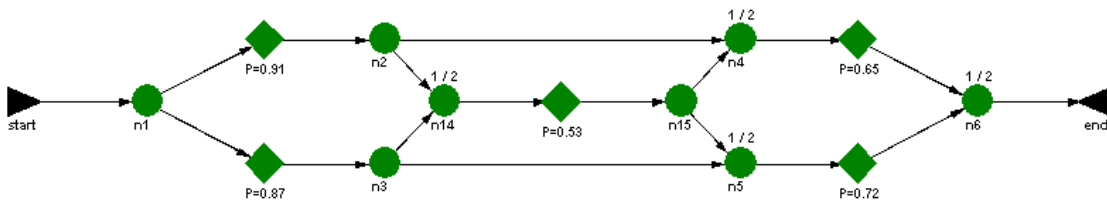


Figure 15.1 Completed RBD for Chapter 15

Events1.rbd starts with the following structure already built for you:

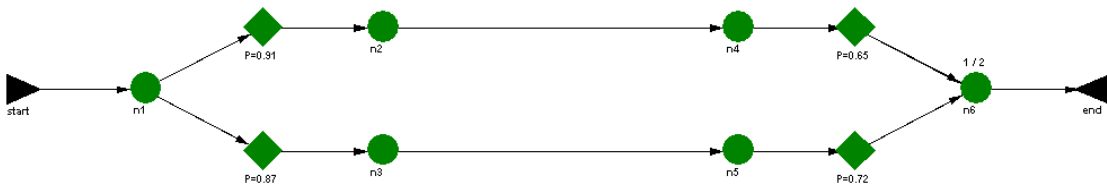


Figure 15.2 Initial RBD for Chapter 15

Let us add an event to the Workspace View, which Figure 15.3 illustrates via the *Edit – Add – Event* menu item. Events can also be added by using the right mouse click or from the add Event button of the toolbar. When an event is placed within the Workspace View, the Event Properties dialog box appears awaiting input. Set this event’s name to “P=0.53” and its “Probability of event success” value box to 0.53 as shown in Figure 15.4.

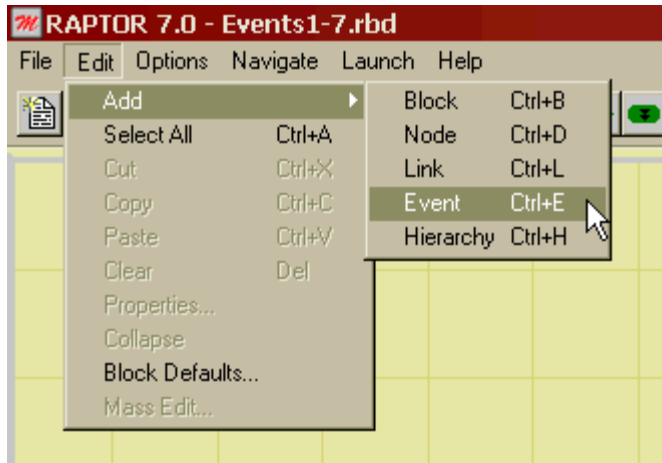


Figure 15.3 Add Event from Edit Menu Option

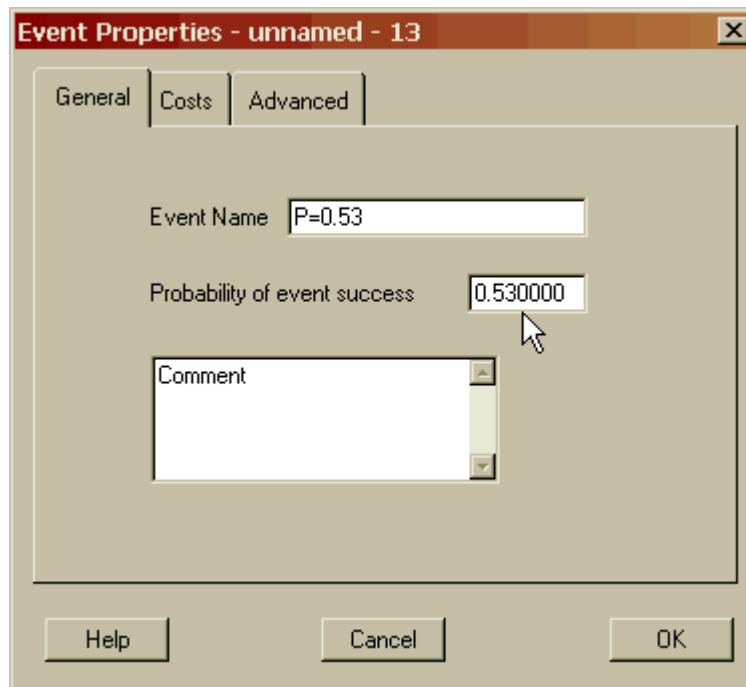


Figure 15.4 Event Properties Dialog Box

Add two nodes and the remaining links until your RBD matches the one shown in Figure 15.1. When you attempt to simulate this RBD the Node Paths dialog box may appear before the simulation starts. The Node Paths dialog box will appear for any *k-out-of-n* node not previously specified, or whose topology has been modified since the last time the RBD was simulated. This dialog box, shown in Figure 15.5, highlights the node whose *k*-information is absent and causes the screen to zoom close to that particular node. It will continue to appear until all unspecified *k-out-of-n* nodes have been designated.

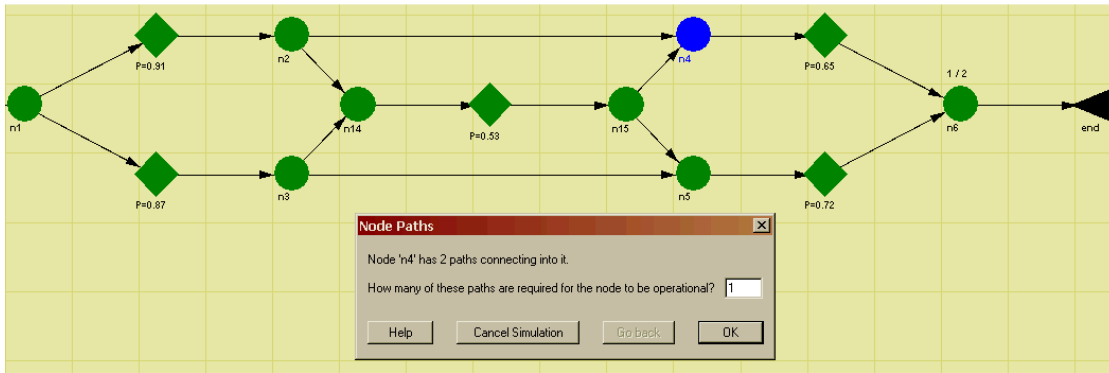


Figure 15.5 Node Paths Dialog Box

If the Node Paths dialog box appears, set the k -values of each of the k -out-of- n nodes in this RBD to a value of one. The Simulation Options dialog box will appear as soon as the Node Paths dialog box is no longer required. Conduct a failure-truncated simulation, allowing one system failure to occur per trial. Allow the simulation to run 200 trials with the graphics engaged. The results for this simulation are shown in Figure 15.6.

Results from 200 runs to 1 failure:

Parameter	Minimum	Mean	Maximum	Standard Dev	SEM
Availability	1.000000000	1.000000000	1.000000000	0.000000000	0.000000000
MTBDE	N/A	N/A	N/A	N/A	N/A
MDT	N/A	N/A	N/A	N/A	N/A
Reliability	0.000000000	0.850000000	1.000000000	0.357967462	0.025312122
Conditional Reliability	0.000000000	0.850000000	1.000000000	0.357967462	0.025312122
Ending Sim Time	0.000000	0.000000	0.000000	0.000000	0.000000

Figure 15.6 Results for 200 Trials

The reliability of this system is 0.85 (Note: This is not known by the term “mission reliability” since zero time passes for this type of simulation). Raptor calculates the

reliability of a system by dividing the number of successful trials by the total number of trials simulated. Thus, for the simulation conducted earlier, we can deduce that 170 of the 200 trials were successful ($170/200 = 0.85$). In order to obtain greater accuracy in the system reliability parameter, we must simulate more trials. Repeat the last simulation, but for this situation conduct 10,000 trials without graphics. The results of this simulation are shown in Figure 15.7.

Parameter	Minimum	Mean	Maximum	Standard Dev	SEM
Availability	1.000000000	1.000000000	1.000000000	0.000000000	0.000000000
MTBDE	N/A	N/A	N/A	N/A	N/A
MDT	N/A	N/A	N/A	N/A	N/A
Reliability	0.000000000	0.872200000	1.000000000	0.333883674	0.003338837
Conditional Reliability	0.000000000	0.872200000	1.000000000	0.333883674	0.003338837
Ending Sim Time	0.000000	0.000000	0.000000	0.000000	0.000000

Figure 15.7 Results for 10,000 Trials

The system reliability for 10,000 trials is 0.8722. This RBD is sufficiently simple such that a mathematical derivation can yield a closed-form solution for the reliability of the system. Thus, it can be shown that the true reliability of this system is 0.8707. The 10,000 trial simulation yielded a reliability value that deviates only 0.2% from the known value. More trials would result in even greater accuracy.

By definition, events happen instantaneously and thus are not functions of time. These types of event-only simulations are usually implemented when it is known that the reliability of the components of the system will not vary much during the time of interest. Another use of this type of simulation is to evaluate whether one system configuration is better than another, all other aspects of the simulation being equal. For example, you might wish to know which of the designs shown in Figure 15.8 is more reliable. By building two RBDs using only events, you can quickly determine which system is more reliable.

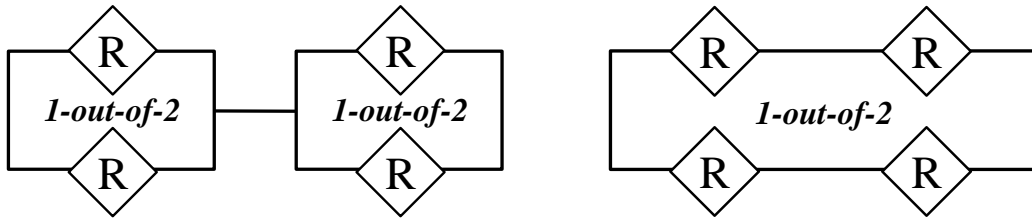


Figure 15.8 Comparisons of Reliability Designs

• • •

You may have noticed during these simulations that the system status indicator was no longer a bull's eye icon. As shown in Figure 15.9, this RBD used a circuit card as its system status indicator.

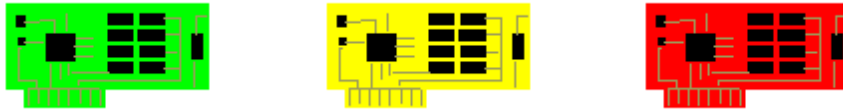


Figure 15.9 Circuit Card Status Symbols

Any one of nineteen status images can be chosen from the *Options – System Settings* menu. Figure 15.10 shows the General Tab, which allows a user to change the status image. Alternatively, during a simulation you can use the right mouse click over the status indicator to cycle through each of the nineteen images.

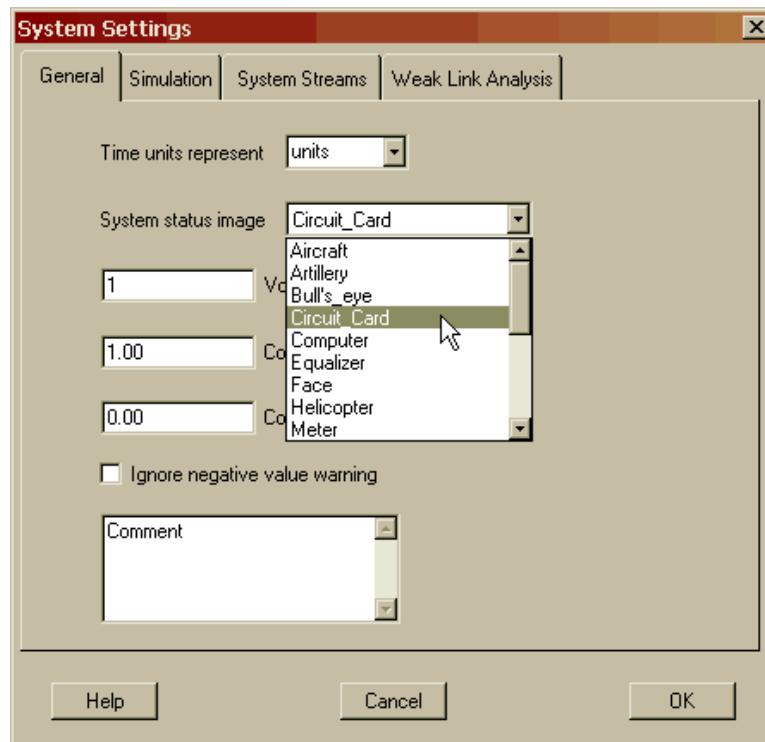
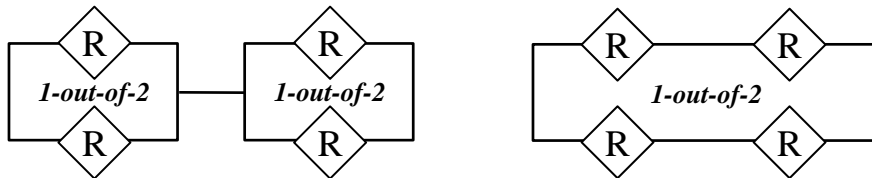


Figure 15.10 System Settings Dialog Box

An interesting aspect of the RBD built for this chapter is that it cannot be simplified into series or parallel sub-structures. The event in the center of the RBD in Figure 15.1 causes the RBD to be called “complex”. The term “complex” in reliability theory does not reflect the number of elements comprising an RBD, but rather that an RBD’s topology cannot be reduced to more simple series or parallel sub-structures. Traditionally, complex RBDs could only be solved by hand and only for systems that did not contain too many elements. The reason for this is that the methods used to solve complex RBDs are based on the mathematics of combinations. Since mathematics of combinations is itself based on the factorial function, it does not take many elements to cause an RBD to be unsolvable by hand. Raptor does not use combinatorial techniques so it is not susceptible to the weaknesses of the traditional methods. In fact, Raptor can resolve complex RBDs of any size. Complex RBDs, however, can be difficult to construct as evidenced by the number of nodes required for this rather unobtrusive RBD.

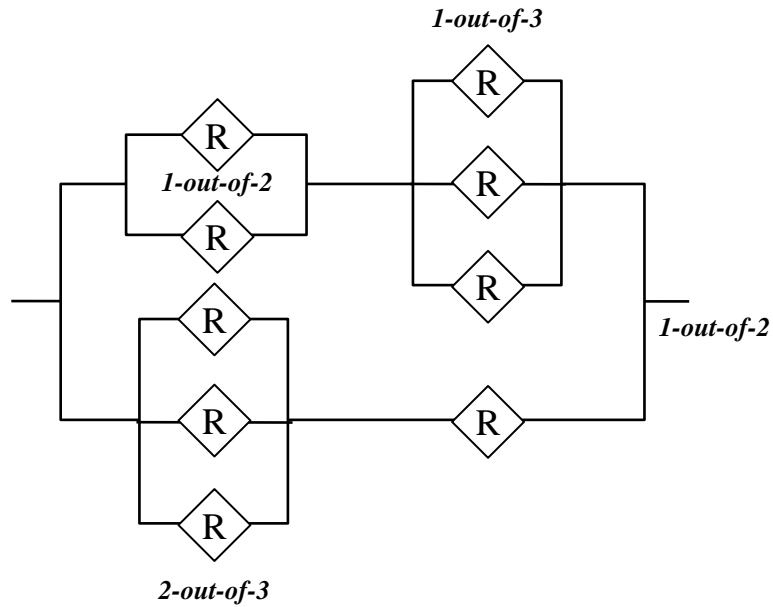
Problems

1. What are the purposes of the zooming feature and the Go Back button of the Node Paths dialog box?
2. Which configuration is more reliable if $R = 0.85$?

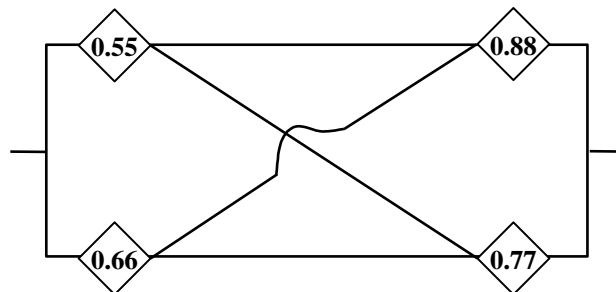


3. Derive a general expression that proves that the reliability of one of the systems in Problem 2 is always greater than the other, regardless of the value chosen for R .
4. What is the minimum number of trials you must run to have confidence in the fourth digit of a system reliability value (e.g., 0.9995, where we want to be sure that the 5 is truly a 5 and not a 4 or 6).

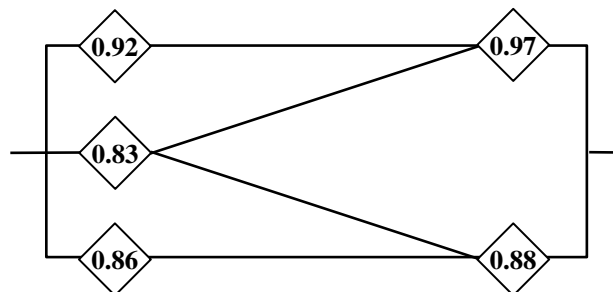
5. If the reliability of all the components shown in the RBD below is 0.76, what is the reliability of the system?



6. What is the reliability of the RBD shown below? All redundant paths are in a 1-out-of-2 configuration.



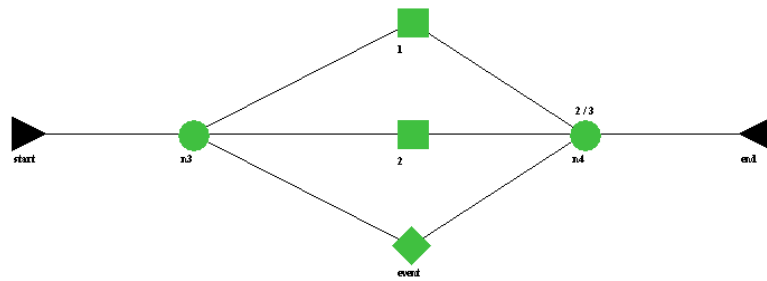
7. What is the reliability of the RBD shown below? All redundant paths are in a 1-out-of-2 configuration.



8. Determine how the standard deviation and standard error of the mean were determined for the reliability parameter shown in Figure 15.6.

9. What is the two-sided 95% confidence bound of the mean reliability for the results shown in Figure 15.6? Use both the traditional t or Z confidence equations as well as the more precise F equations for binomial events. Compare the two methods.

10. If the 2-out-of-3 node shown below is standby, why are all three paths up and running? How would you correct this?



Chapter 16



Objectives

1. Be able to add a hierarchy element.
2. Be able to collapse an existing topology into a hierarchy element.
3. Understand the hierarchy navigation tools.
4. Be able to import an existing RBD as a hierarchy element.
5. Understand the function of the Descendent Status Indicator (DSI)

Recommended Problems

1, 2 and 3

Hierarchies

Often there are situations in the construction of an RBD when the model becomes cluttered. While there are no limitations to the number of components that can be placed within the Raptor Workspace View, eventually too many elements will make for a poor presentation of your model. A crucial part of model development is the verification and validation of a model to those often not knowledgeable in the ways of simulation and reliability theories. Raptor allows the representation of a subsystem by a single element known as a hierarchy. Thus, a large model can be reduced in appearance to its major subsystems to help ease the understanding of a system. As you will see, no information or elements are lost by using hierarchies; the information has been merely hidden below the main or “home” Raptor Workspace window. A hierarchy is shaped like an elongated oval and is the last of the Raptor elements that were briefly mentioned in Chapter 1. Representative hierarchy elements are shown in Figure 16.1.

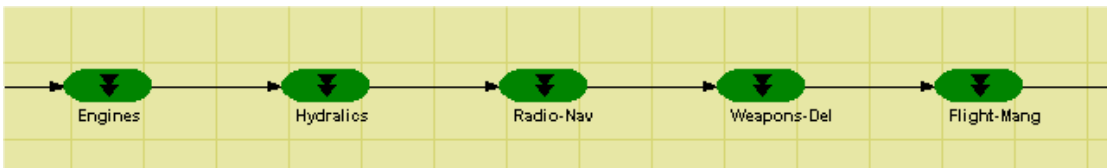


Figure 16.1 Hierarchy Elements

Let us take a look at an aircraft system that could benefit from the use of the hierarchy element. Open the **Hierarchy1.rbd** file and notice in Figure 16.2 that this model is significantly larger than any of the models we have built or assessed thus far.

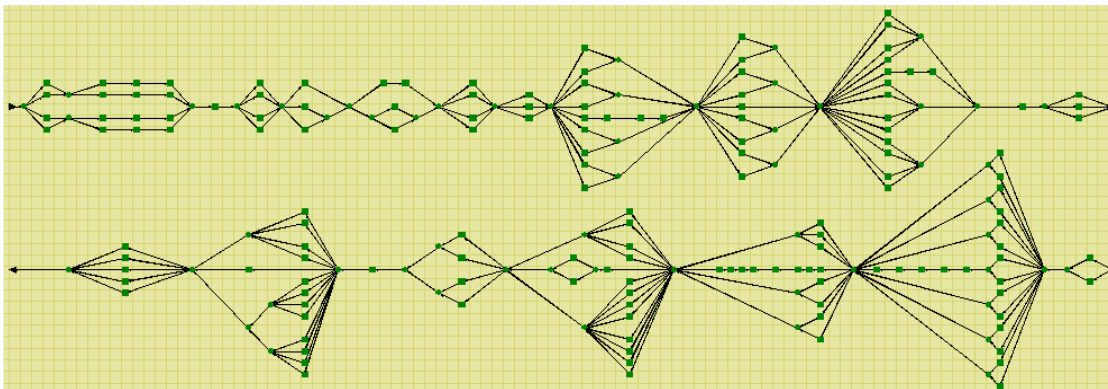


Figure 16.2 Aircraft System

Because of the size of this model it may be difficult to present. The analyst who built this model was wise enough to group the components by subsystems in such a manner that even from a zoomed-out view as shown in Figure 16.2, the subsystems are recognizable as triangular shapes. Yet, to view the entire model, the zoom setting has to be such that the components appear quite small. Because of this some colors will be difficult to distinguish during simulation, especially those with similar hues (e.g., the

blue, red and green shades). Presenting this model to management might create doubt that your model accurately represents the genuine aircraft. Let us make this model more presentable by including some hierarchical structures and in the process learn how to employ the hierarchy element. Right click in the first cluster of components near the top left side of the RBD and select the *Center-zoom* menu item as indicated in Figure 16.3. Change the zoom setting to 20% once the Center Zoom dialog box appears. Your model should look similar to Figure 16.4.

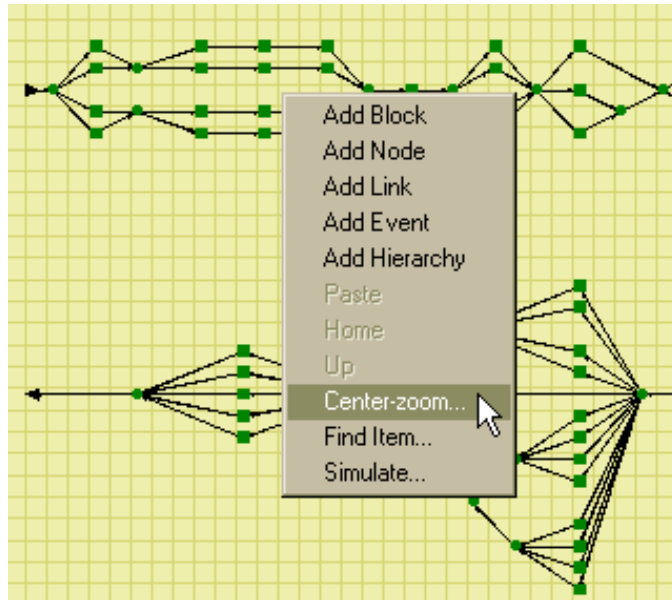


Figure 16.3 Zooming Using the Right Click Menu Option

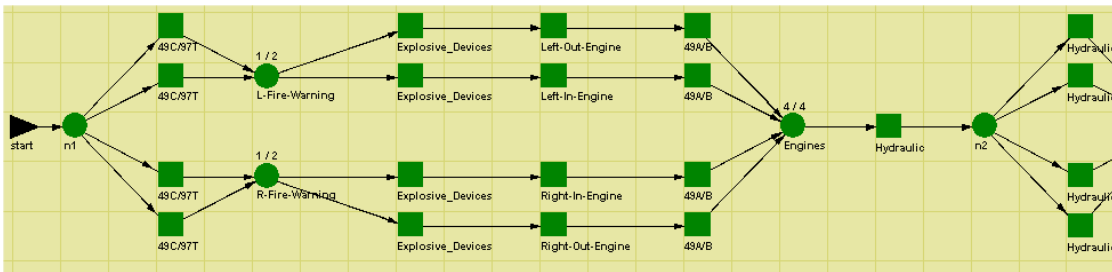


Figure 16.4 Zoomed-in View of the Aircraft System

Now that we have zoomed-in, we can distinguish that the first cluster of elements represents the four engines of the aircraft system. The next cluster of components represents the hydraulics of the system. We will convert both of these clusters of components into hierarchies using two different techniques.

Use the rubber band feature of Raptor (i.e., left click – hold – drag and then release) as shown in Figure 16.5 to highlight the engine subsystem. Select the *Edit – Collapse* menu item and notice that a hierarchical element has been attached to the cursor. Place this element within the Workspace View near the Start marker and once the Hierarchy Properties dialog box appears, modify it as shown in Figure 16.6.

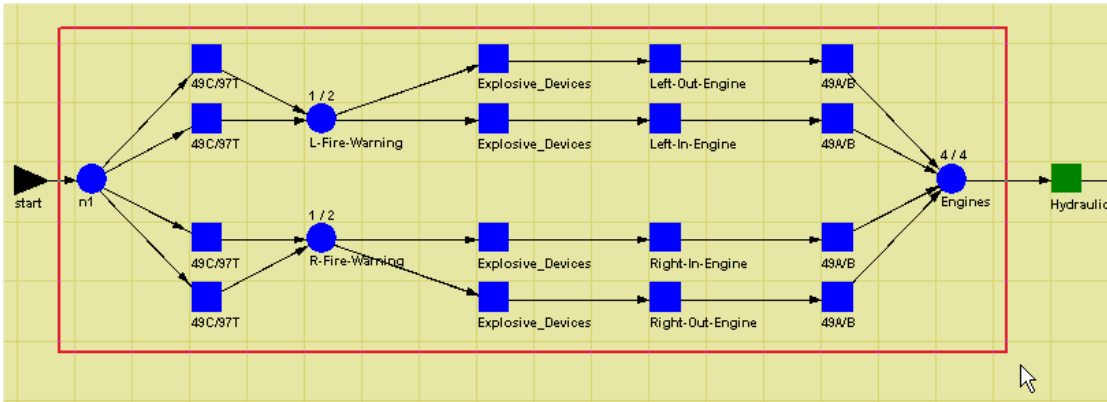


Figure 16.5 Selecting the Engine Subsystem of the Aircraft System

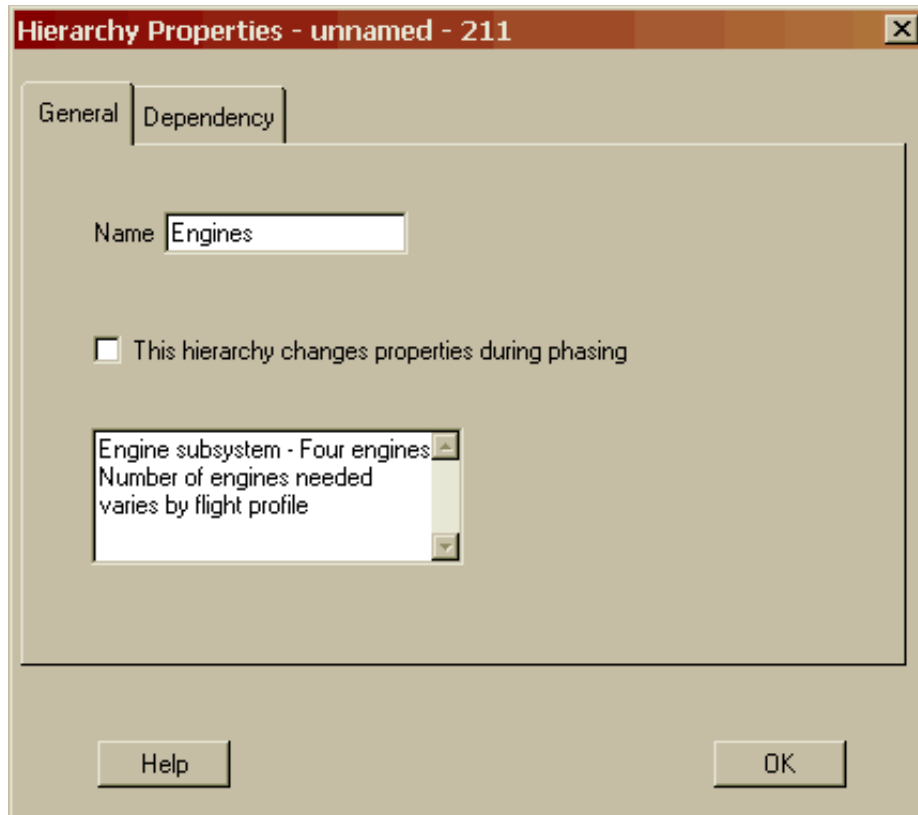


Figure 16.6 Engine Hierarchy Properties Dialog Box

Select the OK button of this dialog box and then double click on the Engines hierarchy element. Notice that this action drops you into a new window (see Figure 16.7) that is one level below the original or home level. The engine components that were previously located on the home level have now been collapsed into a lower or child window of the main system.

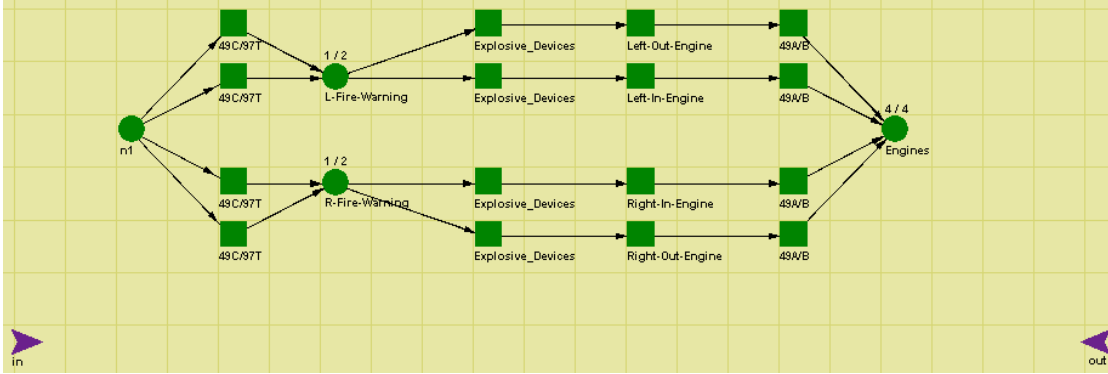


Figure 16.7 Within the Engine's Hierarchy Window

In and Out markers are always included within a hierarchy window and like the Start and End markers cannot be deleted. Connect the engine subsystem with links as shown in Figure 16.8. In essence, this RBD is now embedded within the main RBD.

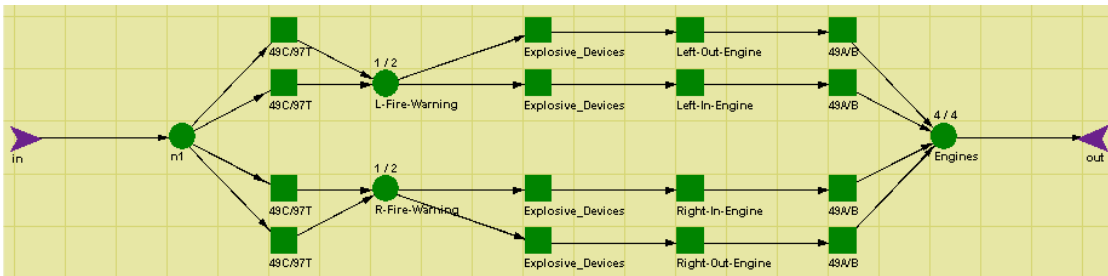


Figure 16.8 Completed Engine Hierarchy Subsystem

There are several ways to navigate back to the home level within a hierarchical RBD. The Home or Up buttons shown in Figure 16.9 can be used or you can simply double click on an Out marker. This action will always take you up one level. This version of Raptor allows for five levels of hierarchical indenturing. Use one of the navigation methods to get back to the home level.

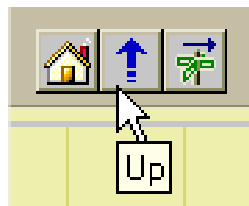


Figure 16.9 Home and Up Navigation Buttons

Add a hierarchy element by using the *Edit – Add – Hierarchy* menu item as shown in Figure 16.10. Place this element near the engine subsystem and modify its properties as indicated in Figure 16.11. If you were to drop into this hierarchy element, you would see that it is empty except for the required In and Out markers. Like other elemental cursor modes, be sure to right click to exit from the hierarchy cursor mode.

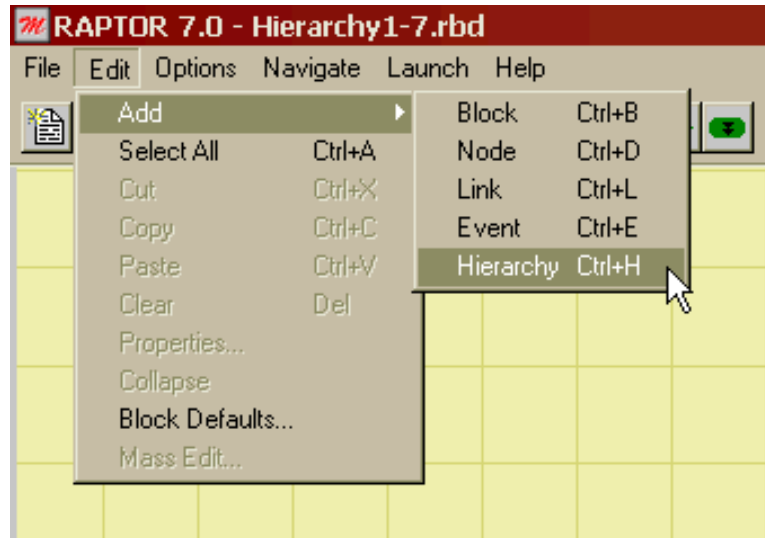


Figure 16.10 Adding a Hierarchy via the Edit Menu Item

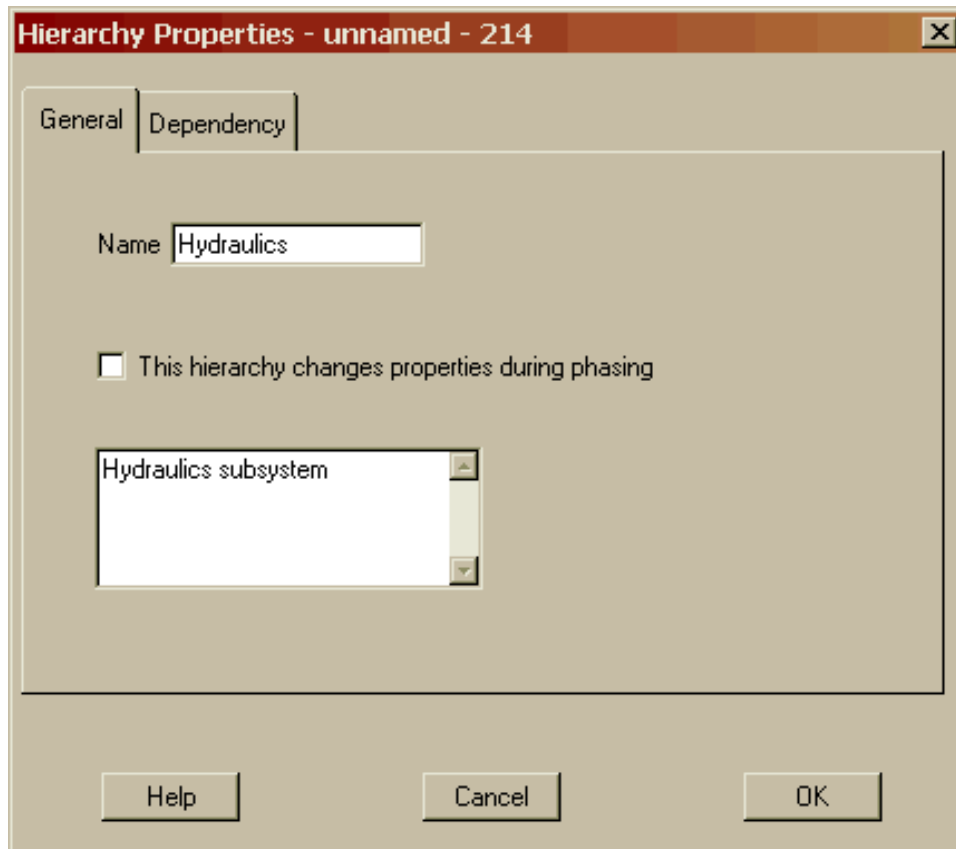


Figure 16.11 Hydraulics Hierarchy Properties Dialog Box

Highlight the hydraulics components and drag them over the newly placed hydraulics hierarchy element as shown in Figure 16.12. Position the cursor over this element for a few seconds until the highlighted components are automatically transported into the Hydraulics window as shown in Figure 16.13.

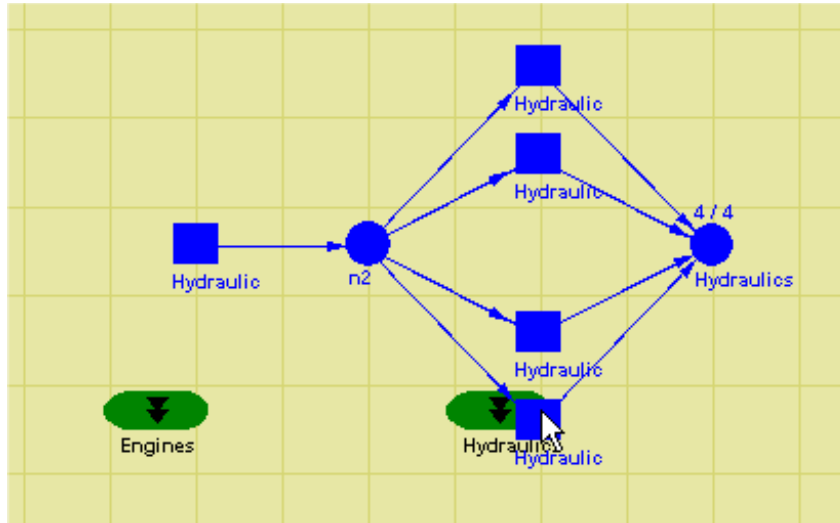


Figure 16.12 Highlighted Components Hovering Over a Hierarchy Element

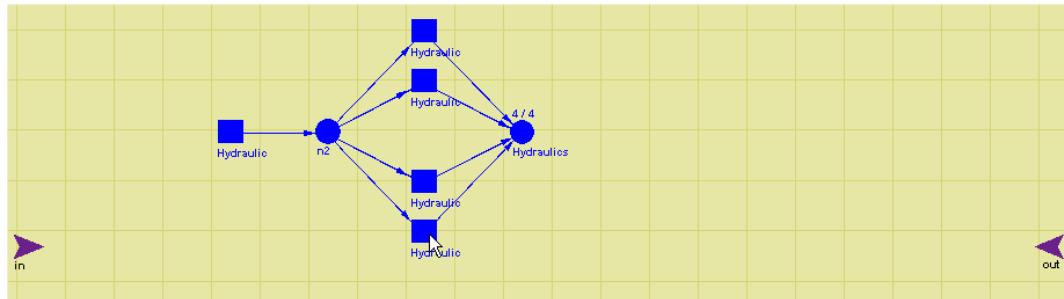


Figure 16.13 Hovered Components being Dropped into a Hierarchy

Add a link from the In marker to the hydraulic component that is in series and one from the serial hydraulics node to the Out marker. Continue collapsing (or creating new hierarchies and dragging the appropriate elements into them) the major subsystems of this aircraft model into hierarchies as shown in Figure 16.14. Be sure to descend into each hierarchy and construct a valid RBD. This could take some time so if you are less adventurous than most, open the **Hierarchy2.rbd** file and note that all of this work has already been completed.

This aircraft system is now comprised of fourteen major subsystems of which thirteen are hierarchies. Additionally, this model contains 149 components, 95 nodes and 343 links. Knowing what is specifically contained within a hierarchical RBD can be at times beneficial. From the *Options – Tables – View System Inputs* menu item, a summary of the number of elements contained within an RBD is available on the first tab of the System Settings Input Tables dialog box as shown in Figure 16.15.

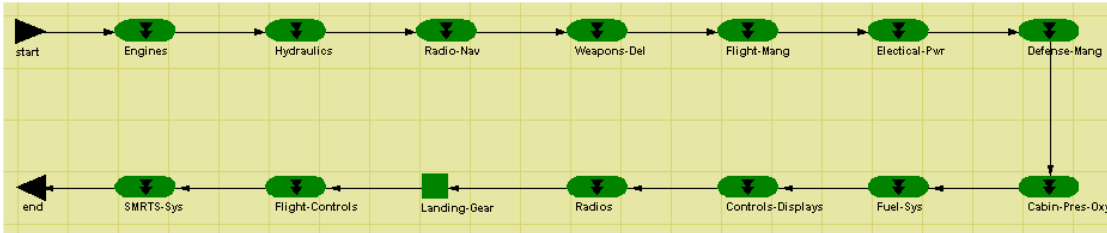


Figure 16.14 Hierarchical Aircraft Model

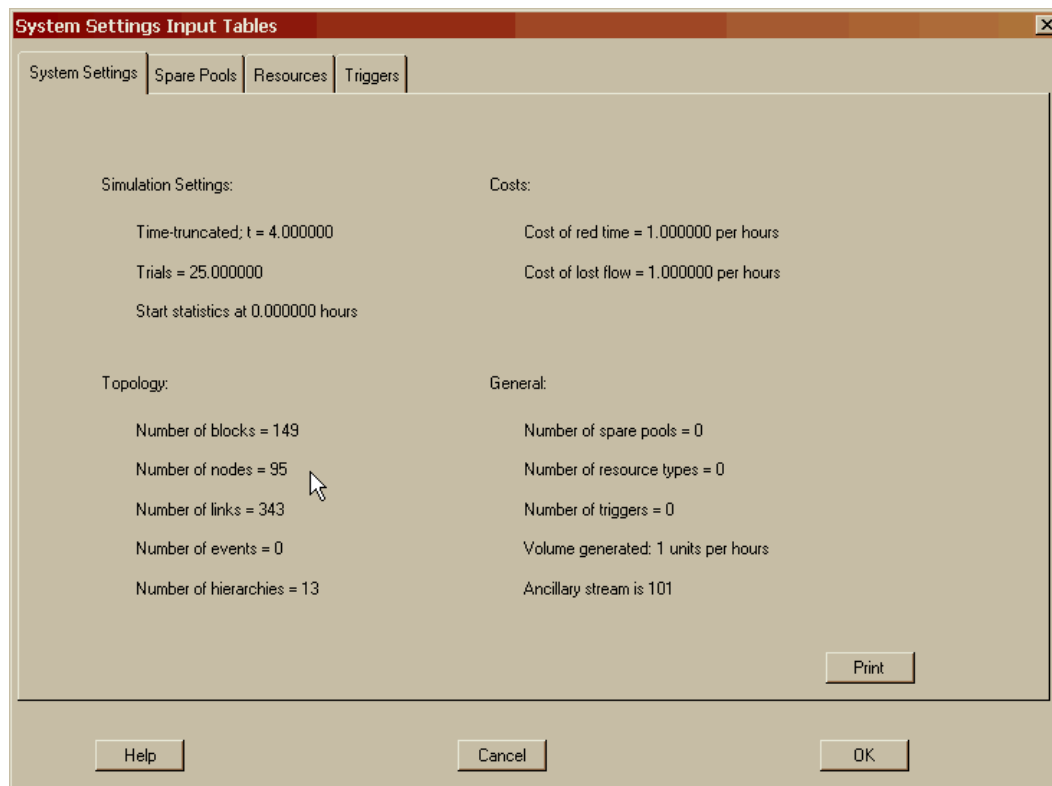


Figure 16.15 Overview of the Elements within the Aircraft Model

In figure 16.14, notice that one block still remains at the home level, that is, the landing gear component. When this model was originally built, it was assumed that the landing gear subsystem was so reliable that little fidelity beyond a single component representing the entire subsystem was needed. What if more detailed information is now available concerning the landing gear subsystem from the manufacturer? If a Raptor RBD model exists of this subsystem, it can be imported in its entirety into the current aircraft model. Remove the landing gear component from the current model and then select the *File – Import RBD* menu item. Choose the **ImportMe.rbd** file from the *Examples* subdirectory as shown in Figure 16.16,

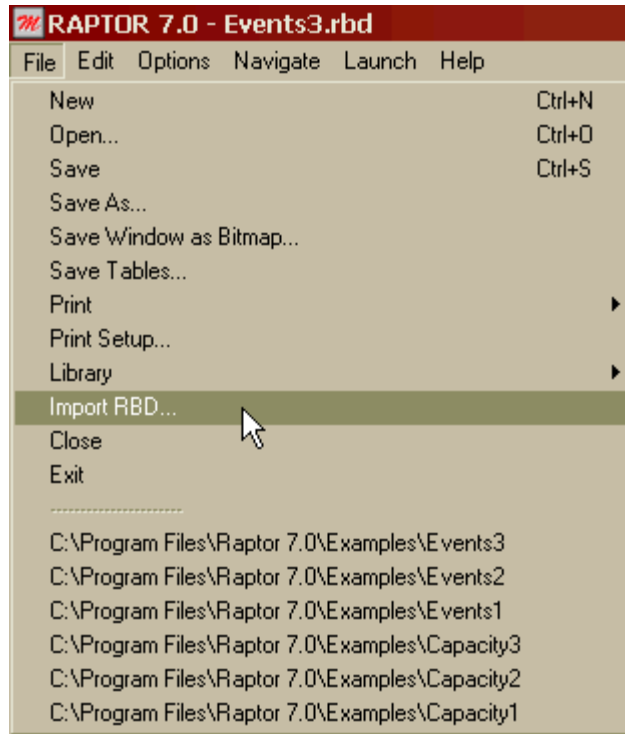


Figure 16.16 Importing an RBD File Menu Item

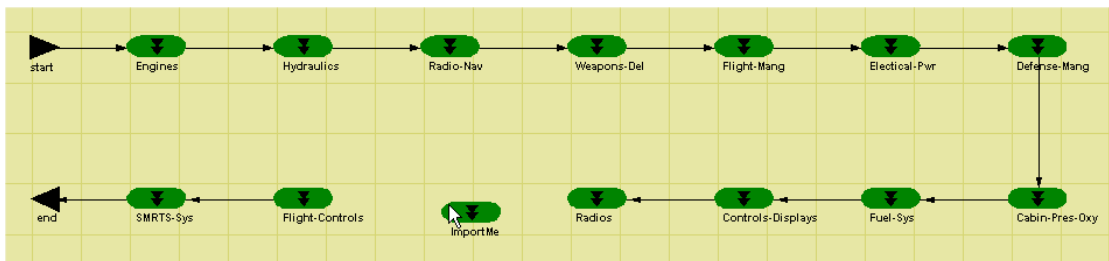


Figure 16.17 Placing an Imported RBD in the Workspace View

Notice that this entire RBD materializes into Raptor as a hierarchy element attached to the cursor as indicated in Figure 16.17. Place the hierarchy element into the Workspace View near the former location of the landing gear component and modify its properties as shown in Figure 16.18. Connect the Radios hierarchy to the LandingGear hierarchy and then to the Flight-Controls hierarchy. Double click on the LandingGear hierarchy and confirm that the landing gear RBD model has been successfully imported as shown in Figure 16.19.



Figure 16.18 Landing Gear Hierarchy Properties Dialog Box

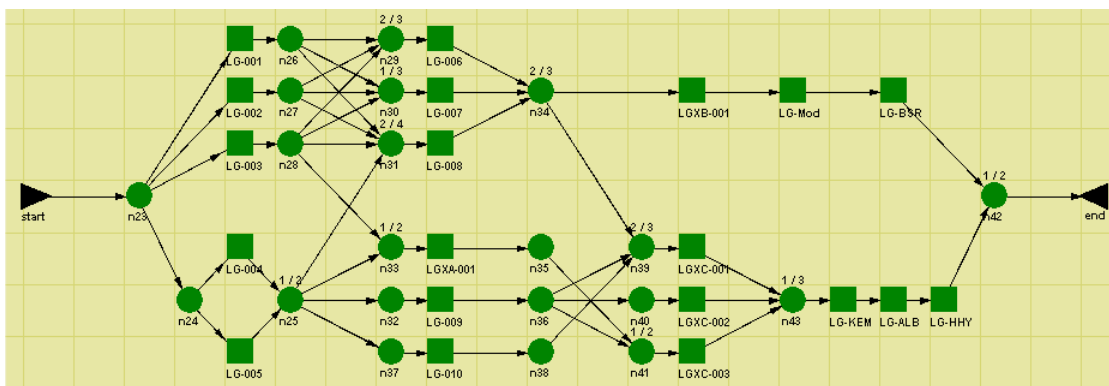


Figure 16.19 Imported Landing Gear RBD

The final and now more presentable RBD is shown in Figure 16.20. This is the type of model that can be presented to management. It appears high-level and simplistic but during a presentation or verification activities, an analyst can demonstrate that the model possesses considerable depth.

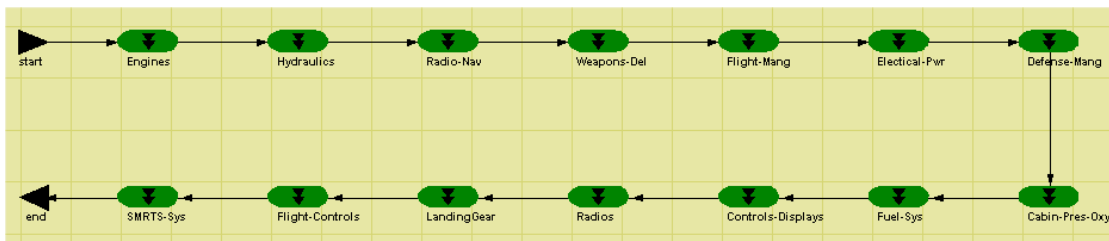


Figure 16.20 Aircraft Model with a Less Complex Appearance

Start the simulation in step mode and take a single step and note the status of the system as indicated in Figure 16.21. During a simulation the hierarchies have the potential to display all the colors of a node. As shown in this figure, the landing gear subsystem is weakened (i.e., yellow hierarchy) but not enough to take the system down so a yellow system status indicator is displayed. During a simulation, you can drop into whichever hierarchy window you desire and watch the localized simulation effects. Double click on the landing gear subsystem and note what is causing the yellow status of this hierarchy element (see Figure 16.22).

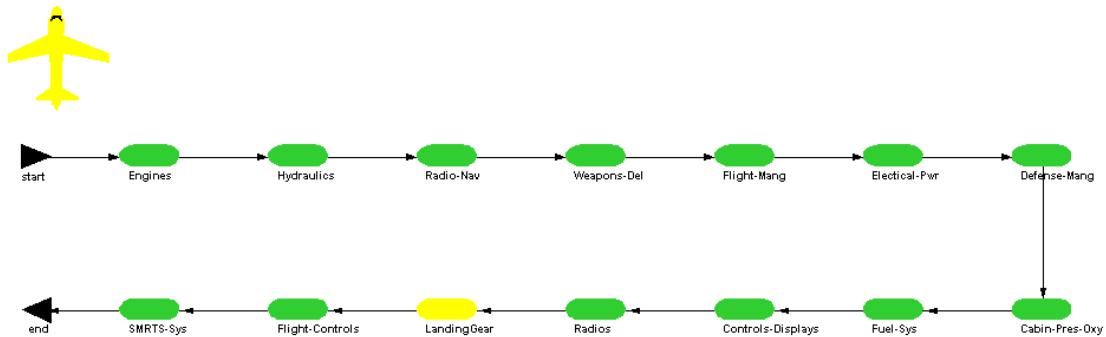


Figure 16.21 Simulated Hierarchy Model

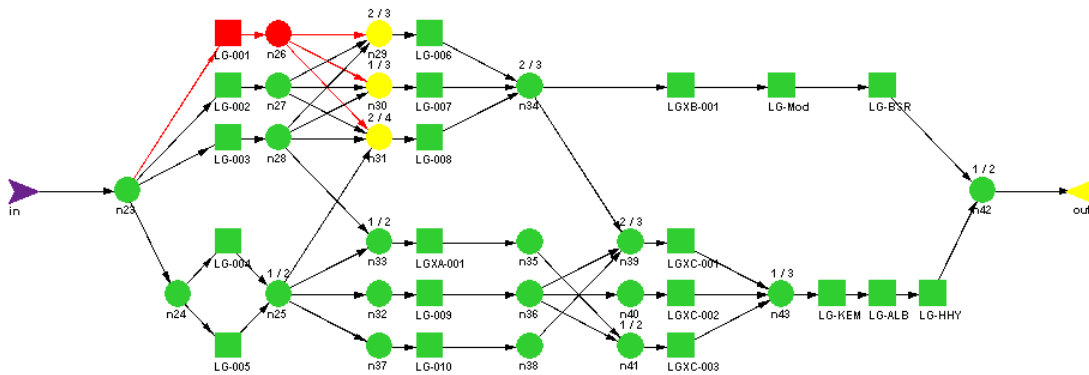


Figure 16.22 Cause of Landing Gear's Yellow Status

Click on the depressed Pause button and allow the simulation to complete. At the end of a simulation the weak link analysis values are displayed on the hierarchies just like they are for the other elements of Raptor. Figure 16.23 shows the weak link reliability values for the major aircraft subsystems. Further weak link analysis insight can be gained by drilling down into a hierarchy to see the weak link values for its subcomponents. Open the Output Tables dialog box and notice on the first tab the reliability of this model is 92%. Thus, this aircraft has a probability of 0.92 of completing a four hour sortie or mission without a system-level failure.

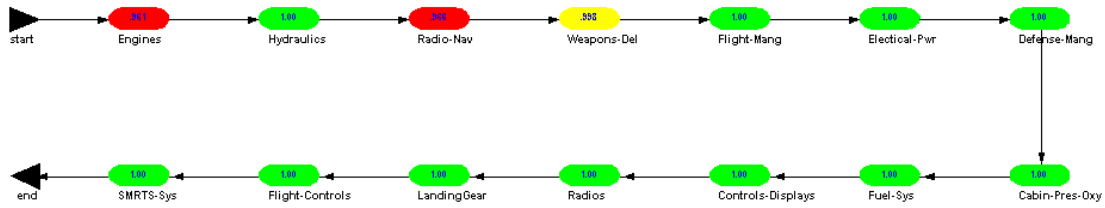


Figure 16.23 Hierarchy Weak Link Analysis Values

Select the Block Analysis tab of this dialog box and notice that the table’s hierarchy filter, which appears on many input and output tables, is now activated (see Figure 16.24). The table filter defaults to “All”, which displays all possible data for a particular tab. Selecting a specific hierarchy filter will only display information in a table for the components contained within that hierarchy element.

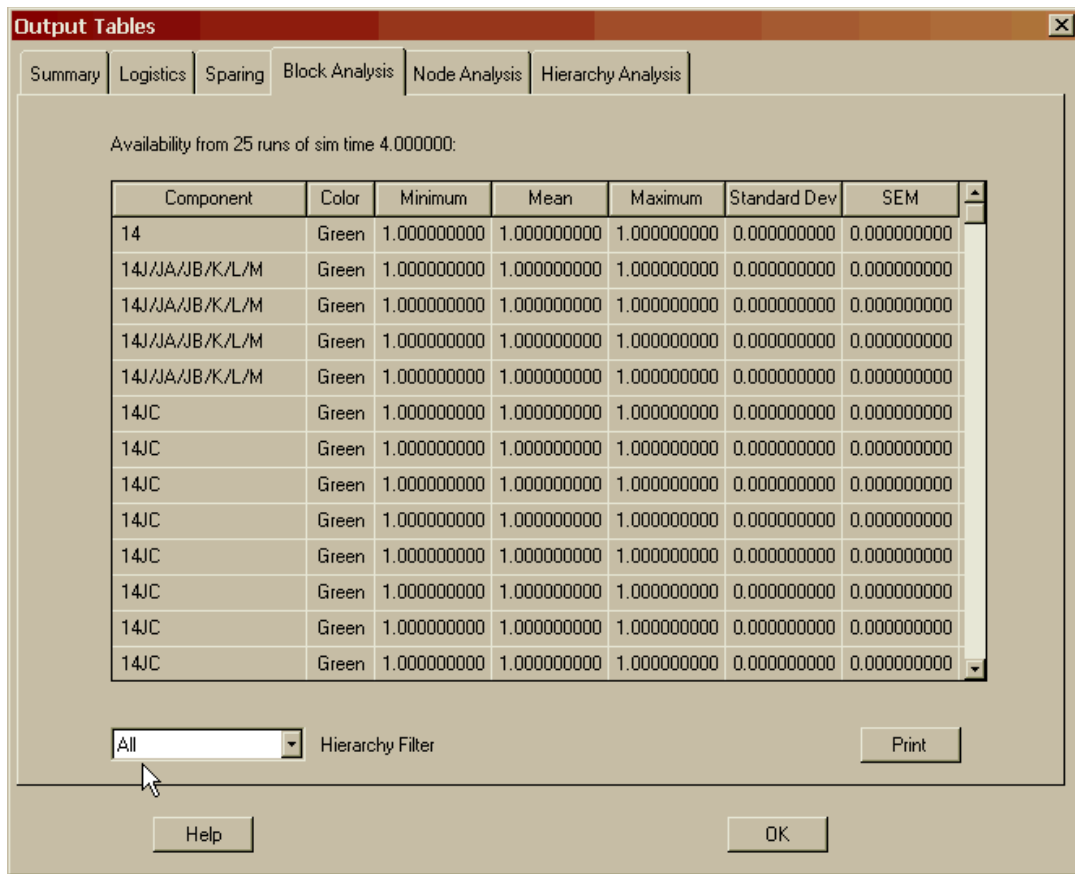


Figure 16.24 Block Analysis Values for Aircraft Model

Change the table’s hierarchy filter to Engines and then sort the mean value from low to high. As indicated in Figure 16.25, the right inboard engine seems to be less reliable than the other engines. In this model the engines are identical and exponentially distributed so a better estimate of an engine’s mission reliability would be the average reliability of the four engines or 0.9902 (i.e., 3.9606/4).

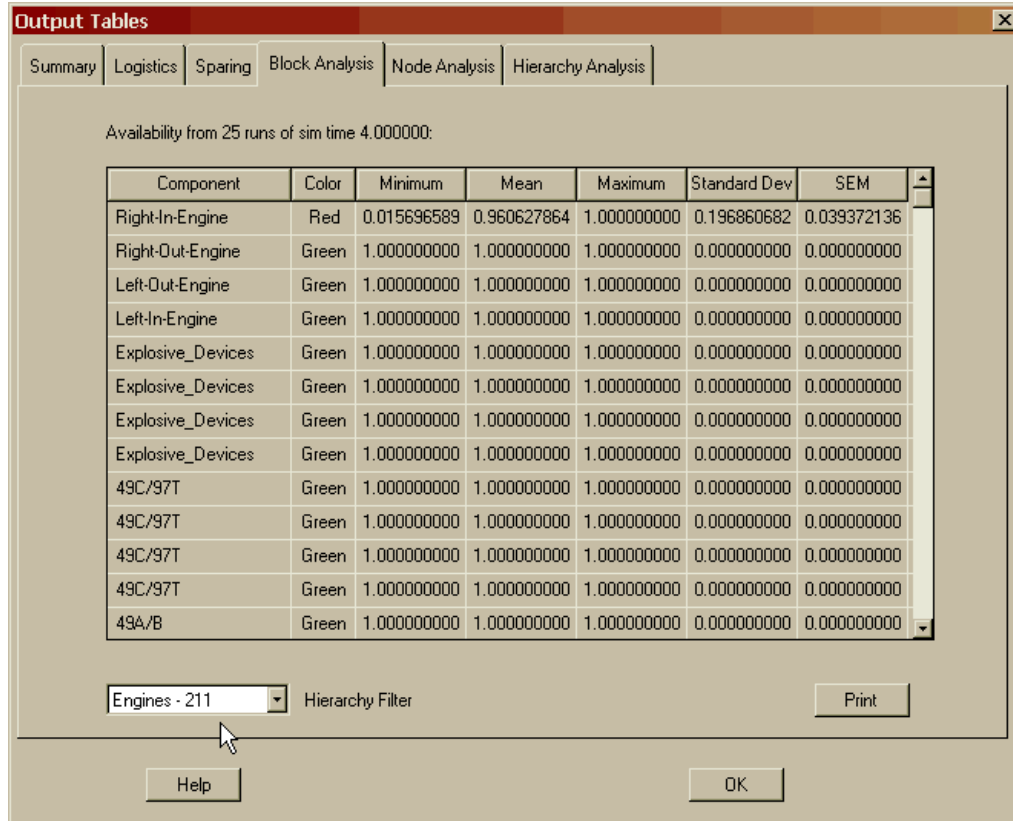


Figure 16.25 Hierarchy Filter and Sorted Block Analysis Values

Close the Output Tables dialog box and return to the Workspace View. There are some additional navigational aids associated with hierarchies that should be reviewed. Select the Display tree button as indicated in Figure 16.26 and note that a dialog box appears entitled with the file name of the currently open file. Figure 16.27 is an indented tree diagram showing the hierarchical structure of the RBD.

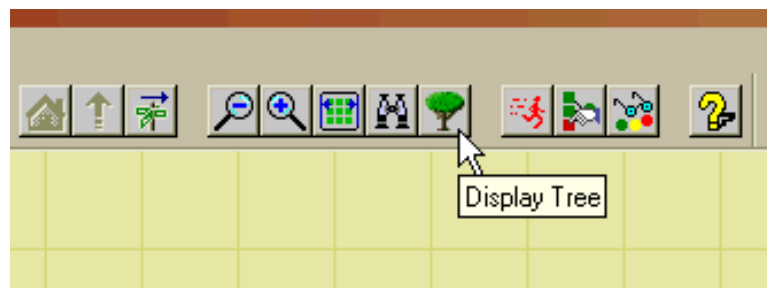


Figure 16.26 The Display Tree Button

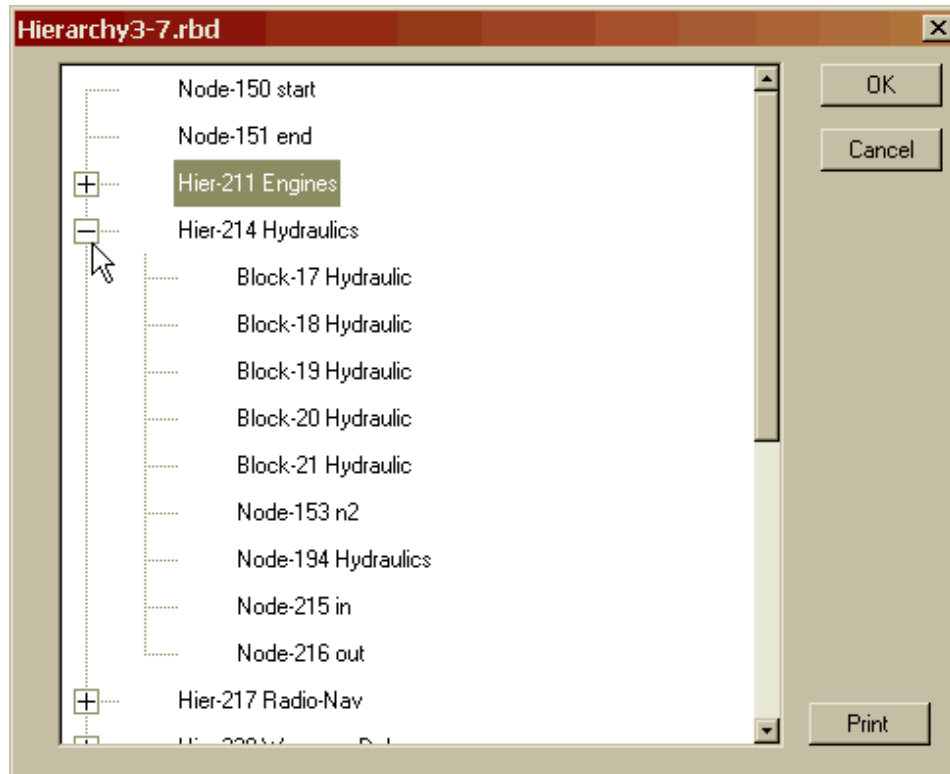


Figure 16.27 RBD Hierarchical Tree

This tree diagram can be printed to help a user see a global map of all the components within a system and indicate their memberships to one hierarchy or another. A printed tree diagram has all of its indentures completely expanded. There exists another global view of the hierarchical elements that can be viewed by selecting the *Options – Tables – View Hierarchy Inputs* menu item and displaying the Hierarchy Input Table dialog box. Figure 16.28 shows a listing of all hierarchies contained within this model as well as how many of each element types the hierarchies themselves contain.

The second column of this table indicates the level of the RBD in which the hierarchy resides. Level zero is the home level so all of these hierarchies with a level of one are children of the home level. A hierarchy with a level of two would be a grandchild to the home level and a child to another hierarchy. The maximum level number is five and a hierarchy with such a level value would be a great-great-great-grandchild of the home level. Any attempt to exceed Raptor’s limit of five levels of indenturing would be fruitless. For example, attempting to import an RBD with three levels of indenturing into the second level will be allowed but if attempted at the third level, it would be blocked. The second column indicates that a hierarchy element can be dependent on other elements. Thus, the flight controls hierarchy could be dependent on the engine hierarchy and when the engines fail, the flight control hierarchy passes this information to all of its descendants. This might cause elements within this hierarchy to invoke their dependency rules and thus ripple back up to the hierarchy itself. It is possible for an element to be dependent on another for which neither are

contained within the same hierarchy. Thus, if the reader was wondering if Raptor could perform trans-hierarchical dependency, the answer is a definite yes.

Hierarchy Name	Level	Dependency Type	Phased	Blocks	Nodes	Events	Hiers
Cabin-Pres-Oxy	1	Independent	False	21	13	0	0
Controls-Displays	1	Independent	False	15	8	0	0
Defense-Mang	1	Independent	False	4	4	0	0
Electical-Pwr	1	Independent	False	15	7	0	0
Engines	1	Independent	False	16	6	0	0
Flight-Controls	1	Independent	False	13	8	0	0
Flight-Mang	1	Independent	False	9	8	0	0
Fuel-Sys	1	Independent	False	15	7	0	0
Hydraulics	1	Independent	False	5	4	0	0
LandingGear	1	Independent	False	20	23	0	0
Radio-Nav	1	Independent	False	14	10	0	0
Radios	1	Independent	False	4	6	0	0
SMRTS-Sys	1	Independent	False	5	4	0	0
Weapons-Del	1	Independent	False	12	8	0	0

Figure 16.28 Hierarchy Input Table

There is one more navigation button that will help facilitate movement around the multiple window environment of a hierarchical RBD. The Goto Hierarchy button shown in Figure 16.29 brings up the Goto Hierarchy dialog box (see Figure 16.30), which allows a user to jump to a particular hierarchical window.



Figure 16.29 Goto Hierarchy Button

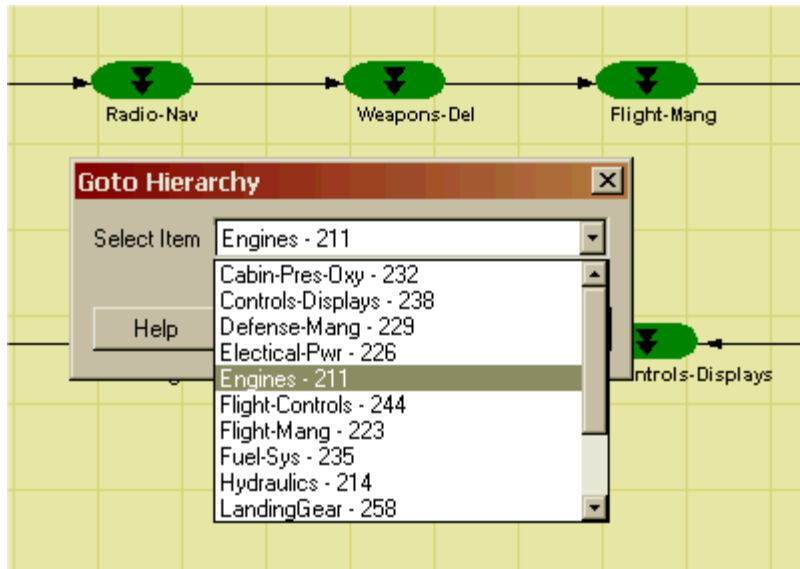


Figure 16.30 Goto Hierarchy Dialog Box

For models with more than one level of indenture, it can be difficult to determine how deep the problem component lies within a hierarchical subsystem. Located in the center of a hierarchy image within the Workspace view (see Figure 16.31), is a symbol of two black triangles. For any hierarchy with two or more levels of indenturing, that symbol has the potential to change color within the Simulation and Failure Effects Views. This symbol is called the Descendent Status Indicator (DSI) and provides insight into the status of elements below the immediate hierarchy window. Open the **Hierarchy4.rbd** file and start the simulation in step mode. Take one step and notice the DSI of the Electronics hierarchy in Figure 16.32.

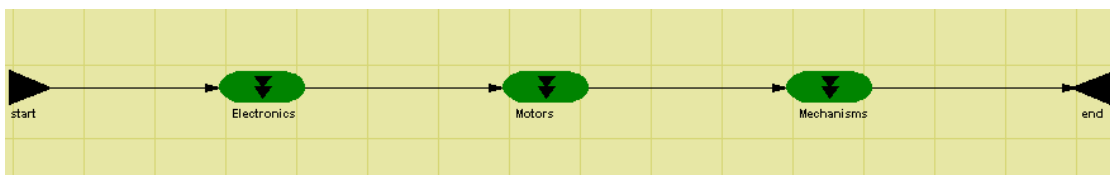


Figure 16.31 Model with Numerous Levels of Indenture



Figure 16.32 Yellow DSI on the Green Electronics Hierarchy

As shown in Figure 16.32, the Electronics hierarchy element has a green body with a yellow DSI. This means that the Electronics subsystem has some maintenance ongoing beneath the current window but the subsystem itself is still functioning. In fact given that the DSI is yellow, we know that it is the result of some sort of redundancy failure and not a serial loss. Double click on the Electronics hierarchy and note that the 1553 Card is a solid yellow with no DSI as shown in Figure 16.33. Also

note that the Out marker changes color to indicate the color status of the subsystem being displayed.

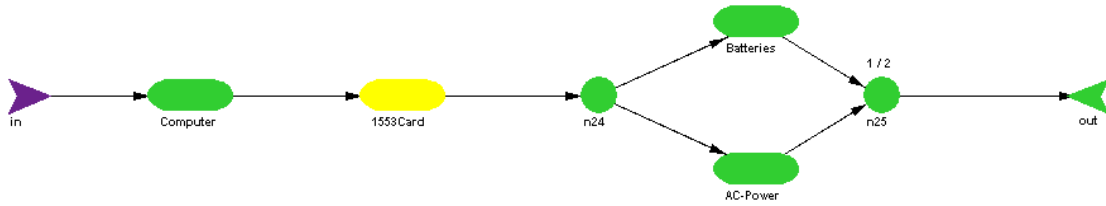


Figure 16.33 Yellow Body of the 1553 Card Hierarchy

You must have two or more levels of indeture below a hierarchy for its DSI to be displayed. Double click on the 1553 Card hierarchy as indicated in Figure 16.34. We now realize the cause of the yellow color of the Electronics' DSI is the failure of the Data-Bus-15 component.

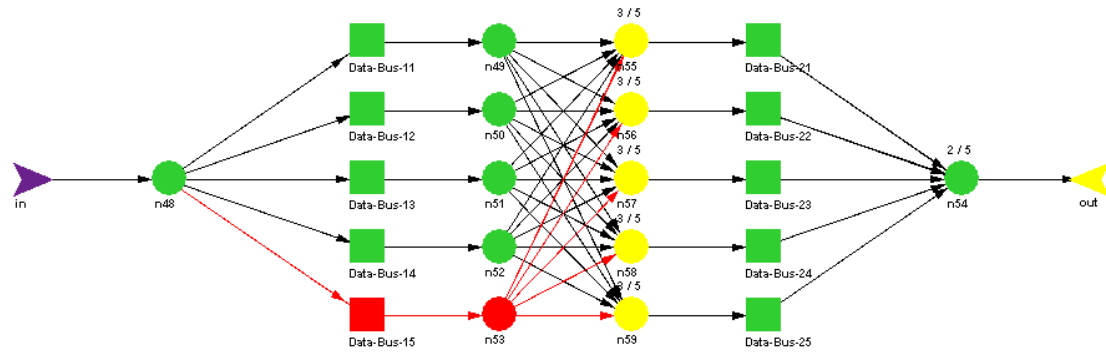


Figure 16.34 Failed Data Bus Component

Return to the home level and take another step such that your RBD matches the one shown in Figure 16.35.



Figure 16.35 Red Body of the Motors Hierarchy

Double click on the Motors element and note that the motor's armature has failed (see Figure 16.36) and thus has brought down the entire system.

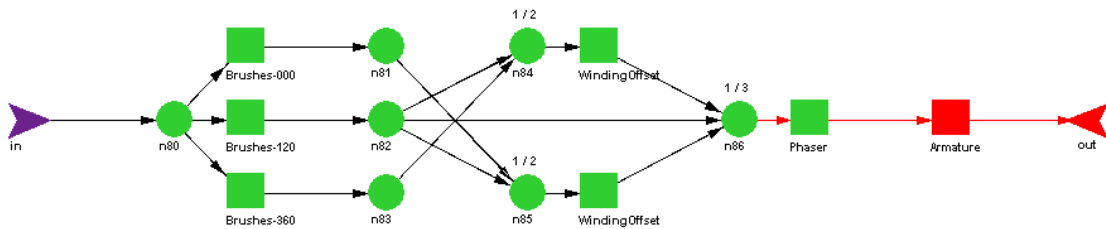


Figure 16.36 Failed Armature Component

Return to the home level and take four more steps such that your RBD appears as the one in Figure 16.37



Figure 16.37 Red DSI on the Yellow Electronics Hierarchy

In this case, the Electronics hierarchy has a yellow body with a red DSI. The yellow body indicates that there is a loss of redundancy one level down and the red DSI indicates that at some level further down there is a failed hierarchy element. Double clicking on the Electronics hierarchy indicates that the AC-Power hierarchy has failed as shown in Figure 16.38.

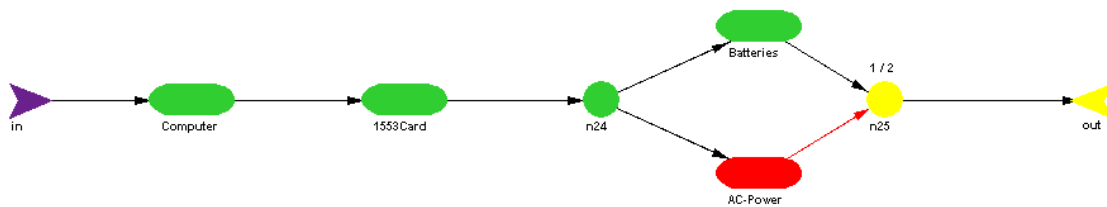


Figure 16.38 Red AC-Power Hierarchy

Double click on the AC-Power hierarchy to determine the ultimate problem with the Electronics hierarchy. As shown in Figure 16.39, the Rectifier component has failed causing the AC-Power hierarchy to be failed, which in turn causes the Electronics hierarchy to have a yellow body with a red DSI.

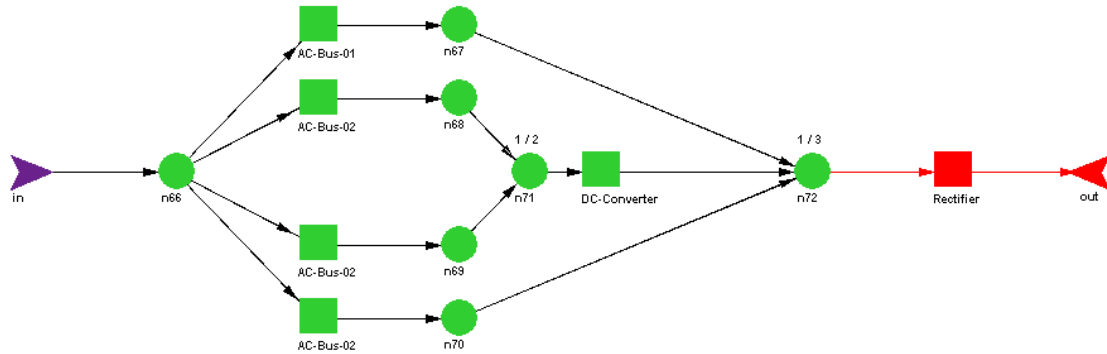
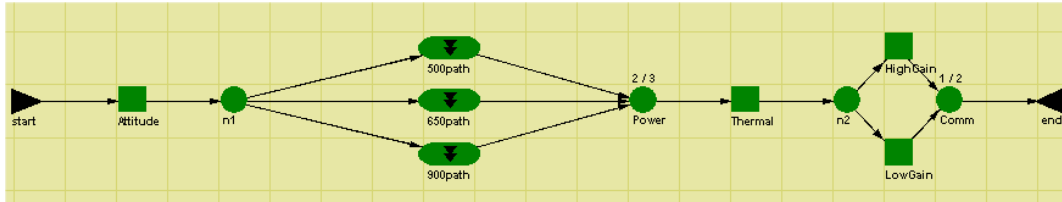


Figure 16.39 Failed Rectifier Component

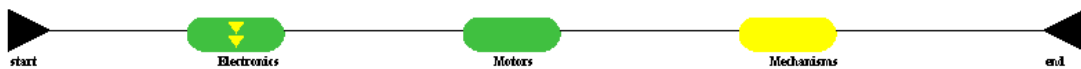
In summary, the body of a hierarchy will be red if it cannot function, yellow if it is weakened and green if there are no downed components beneath this element. The DSI on a given hierarchy always indicates the worst case status of all of its internal hierarchies. A DSI has the potential to take on any color that a node itself can assume.

Problems

1. Modify the Capacity3.rbd file such that the three power strings are converted to hierarchical structures as shown in the figure below. Determine the availability and MDT parameters. Does it match those in Figure 14.22? Should the results match?



2. Explain the color of each hierarchy shown in the figure below.



Can you recreate this situation in Failure Effects view?

3. Explain the color of each hierarchy shown in the figure below.



Can you recreate this situation in Failure Effects view?

4. Explain the color of each hierarchy shown in the figure below.



Can you recreate this situation in Failure Effects view?

5. Explain the color of each hierarchy shown in the figure below.



Can you recreate this situation in Failure Effects view?

Chapter 17



Objectives

1. Understand which block, node, event and hierarchy attributes can be modified from one phase to another.
2. Understand the phasing nomenclature.
3. Be able build a phased RBD.

Recommended Problems

1, 3, 4, 6, 8, 9, 10, 11, 13, 14, 16, 18, 19 and 20

Recommended Readings

Determining the Ballistic Missile Defense Shield's Effective Reliability (Murphy)
Reliability Analysis of Phased-Mission Systems: A Correct Approach (Murphy, Carter & Malerich)

Phasing

Often a system changes its configuration as time passes. These discrete configuration changes at defined times are known as “phases”. These segments of time could represent different employment shifts or portions of a system’s mission. For example, an aircraft’s flight profile might be separated into three distinct phases, namely, takeoff, cruise and landing. Each of these phases might require different equipment or levels of redundancy for the system to function properly. A four engine aircraft might need all its engines during the takeoff phase, but only *3-out-of-4* during the cruise phase, and *2-out-of-4* during the landing phase. Whenever it is desirable to change the attributes or configuration of a system as time passes, you will want to build a “phased RBD”.

Raptor allows simulations to be segregated into at most 999 phases. Blocks, event, nodes and hierarchies all can be designated as phased elements and thus are eligible for their attributes to be modified from one phase to another. This lesson will instruct you on how to build a phased RBD, and how to determine which attributes of the different elements can be changed within a phase. Open **Phasing1.rbd** and note the new elements that have been added to the space station system and that the power strings are now represented by hierarchies. Figure 17.1 shows that two new power sources have been added to this system. The top string provides nuclear power to the system while the bottom string provides electrical power via redundant tether devices.

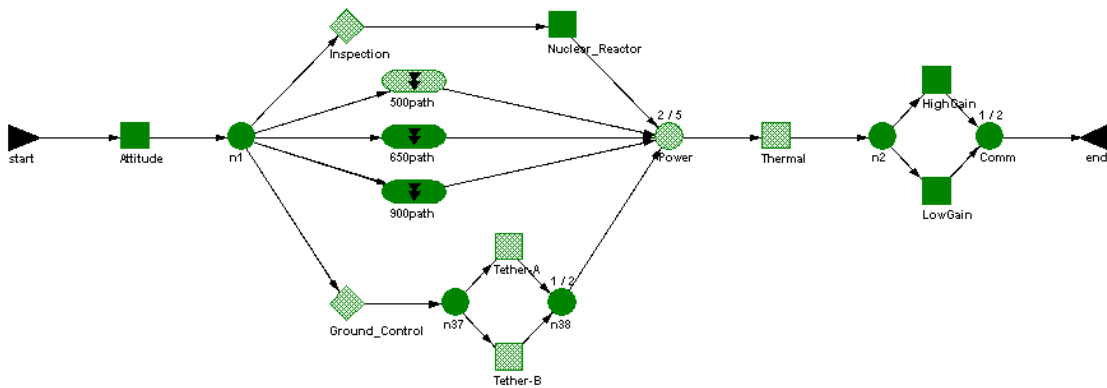


Figure 17.1 Partially-Phased RBD

Operations of the space station will be subdivided into four shifts of five hours and one maintenance shift of four hours in duration. Only the ten elements that are displayed as hatched will change their characteristics as the system operates from one phase to another. These elements are the Tether and Thermal components, the Power node, the Ground_Control and Inspection events, the 500path hierarchy and its three subcomponents. We will first indicate how Raptor knows that only these elements are phased elements (this has already been done for you but let us review how it was accomplished).

Double-click on the Tether-A component and select the Advanced tab of the Block Properties dialog box as shown in Figure 17.2. The first checkbox on this tab instructs Raptor to consider this component to be a phased element.

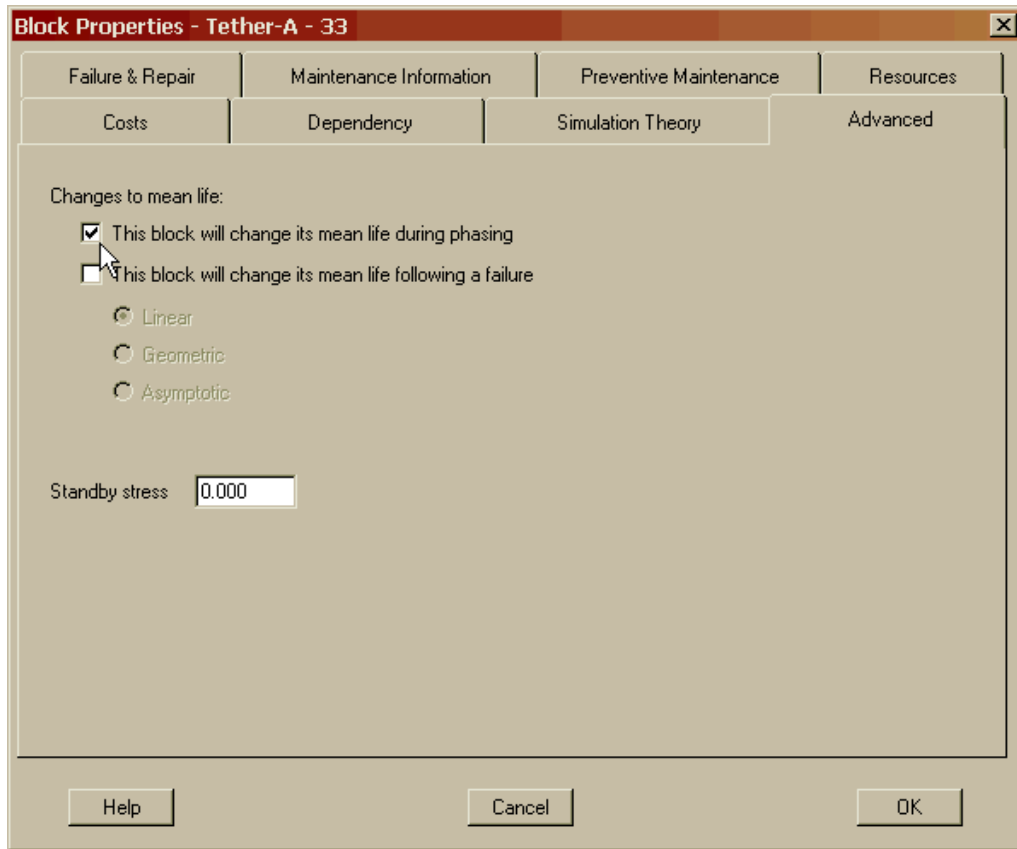


Figure 17.2 Phased Tether-A Component

Of the more than eighty attributes that can be assigned to a component, its most crucial attribute is its mean life. In the previous models illustrated in this workbook, all components possessed a static mean life. For example, an exponential component with a mean life of 100 hours never changed to an exponential component with a mean life of 50 hours within a single simulation. The mean life of a component might change by being diminished due to less stress on the component when it is not used or needed. Some components actually wear out faster (i.e., higher mean life) when not used since they are not being lubricated (e.g., pumps in the petrochemical industry). By designating a component as phased, Raptor will recognize such a component to be dynamic in nature and allow its mean life to vary within a simulation trial.

Often components are not needed for a portion of a simulation trial. For example, a bomber aircraft does not need its bomb release mechanism during its return flight. If this mechanism fails during ingress this would be considered a failure of the system, but if this component fails during egress it would not be considered a system-level failure. To accommodate components that are occasionally not needed, Raptor allows components to be “cut” or “linked” within a system. A cut or linked component is not

considered a part of the system and thus a failure of this component will have no impact on the system. The concepts of cut and linked components will be discussed in greater detail later in this chapter. For now just realize that a phased component can change its mean life and be designated as part of its system or not.

Leave this dialog box and double click on the Inspection event to open its Event Properties dialog box. This event has been declared to be a phased element since its phasing checkbox on the Advanced tab has been marked as shown in Figure 17.3.

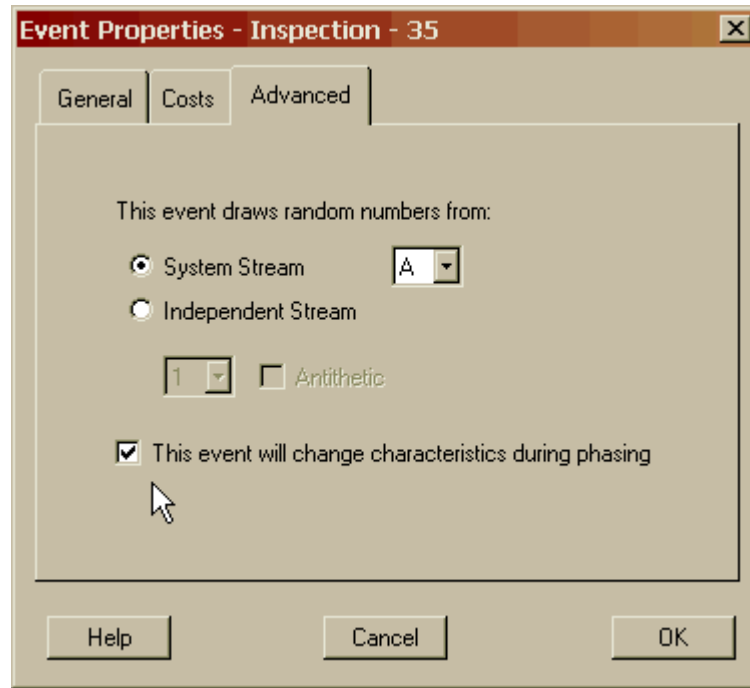


Figure 17.3 Phased Inspection Event

The key attribute of the event element is its probability of success. A non-phased event always fires once at simulation time zero with a constant probability. A phased event can fire multiple times, at different times and can vary its probability of firing. As will be demonstrated later in this chapter, events can also be cut or linked from a system.

Leave this dialog box and double click on the Power node to open its Node Properties dialog box. This node has been declared to be a phased element since its phasing checkbox on the Advanced tab has been marked as shown in Figure 17.4. The most important attribute of a node is its *k-value* or the number of paths into the node that must be functioning for that node itself to be considered functioning. A non-phased node possesses a static *k-value* while a phased node can vary its *k-value* and thus vary the node's redundancy requirement. A node can also be cut or linked from a system resulting in a localized string of components (i.e., the components of its domain) that are not needed or considered a part of a system.

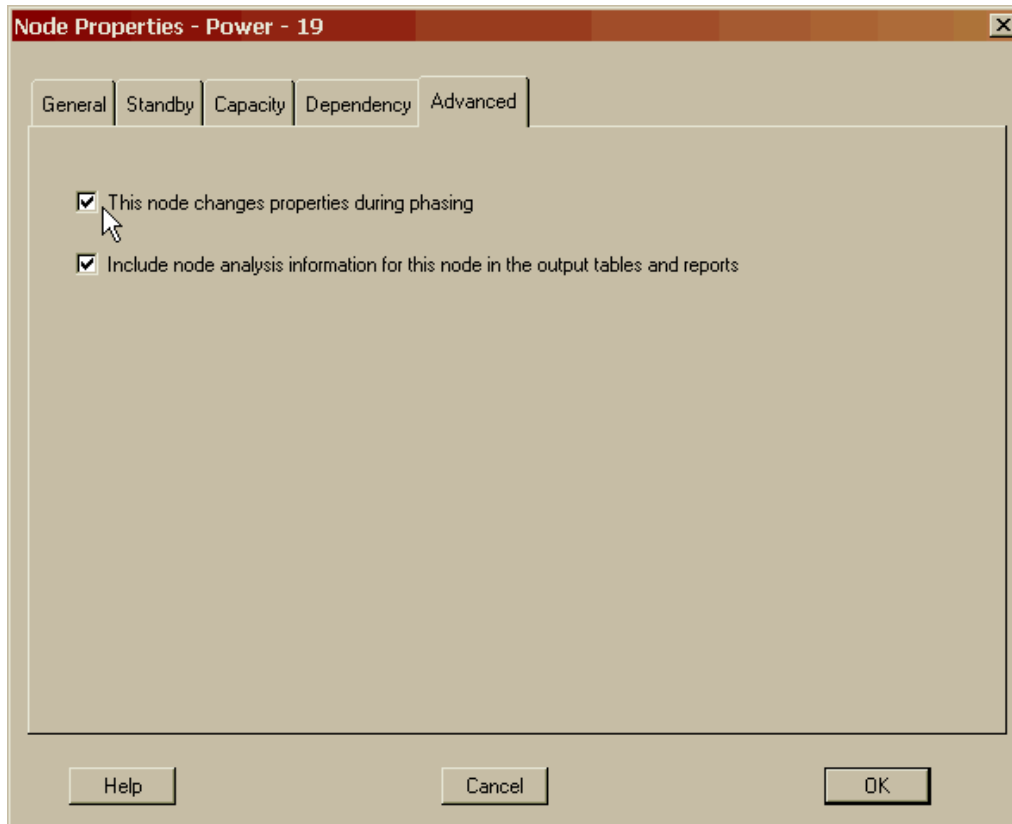


Figure 17.4 Phased Power Node

Common reasons for changing a node's *k-value* usually involve a varying demand on a system's redundancy requirements. For example, a ship with multiple engines might need more engines during rough sea conditions compared to the number needed to perform more benign actions such as docking maneuvers. An electrical system comprised of say ten power sources might need to deliver more power during peak demand. During low demand a *5-out-of-10* system is sufficient but during its greatest demand *9-out-of-10* power sources are needed. A phased node can model this common characteristic of varying *k-values*.

Leave the current dialog box and right click on the 500path hierarchy element and select the *Properties* menu item. Recall that double clicking on a hierarchy takes you into a new window as opposed to opening its properties dialog box like the other elements. Although there are no truly key attributes of a hierarchy element since it merely defines a sub-window within Raptor, a hierarchy element can be phased as well (see Figure 17.5). A phased hierarchy allows an entire subsystem to be linked or cut from a system without the need to define all of a hierarchy's subcomponents as being phased.

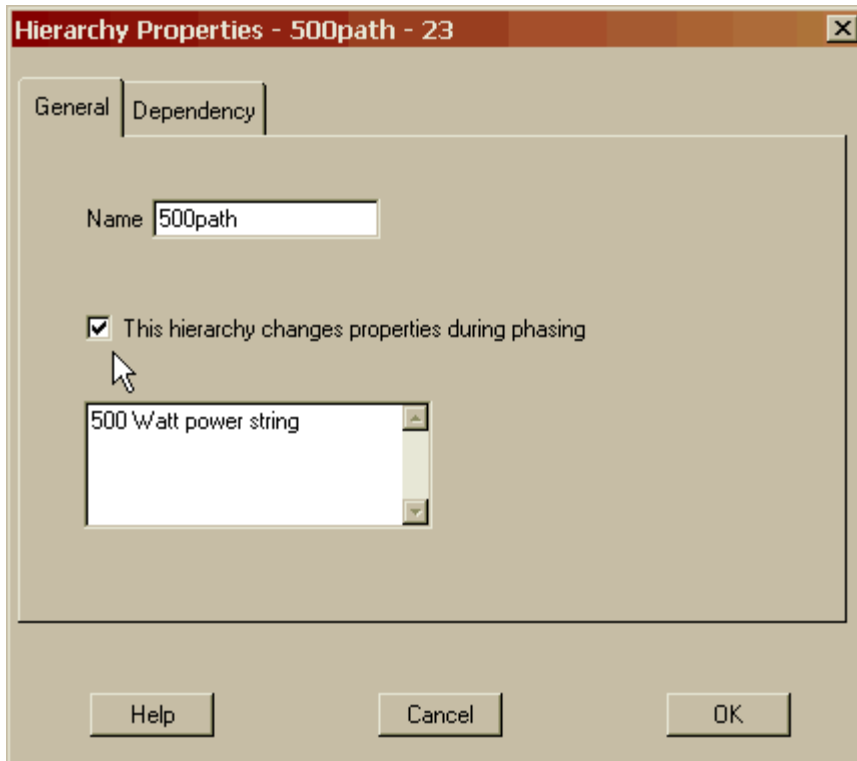


Figure 17.5 Phased 500path Hierarchy

Having now explored how to declare each element type as phased, the process of creating a phased RBD is only half accomplished. Phases themselves will need to be defined before a phased simulation can be conducted. By default, all non-phased RBDs, those without any phases defined, can in essence be considered a single phased RBD. Thus far in this workbook, only single or non-phased simulations have been conducted. A single phased RBD is considered a non-phased RBD because without multiple phases, variation in the elements' key attributes cannot be specified.

Let us break up the operations of the space station model by defining a day to be approximately comprised of five segments; the first four phases are each five hours in duration and the last is four hours in length. These segments or phases will represent different topologies of the system by varying the key attributes of the phased elements. Thus, we will essentially model five different RBDs within the same simulation and account for the effects of traversing from one phase to another. To define phases, we can either select the *Options – Phases* menu item or the Phasing button of the Workspace toolbar (see figure 17.6).

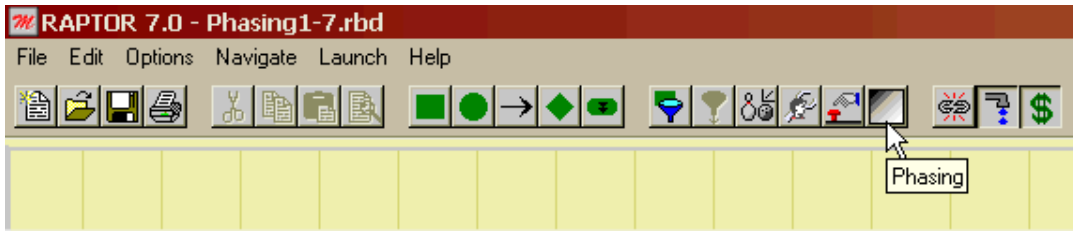


Figure 17.6 Phasing Button of the Workspace View

Figure 17.7 displays the appearance of the Phasing dialog box with three phases previously defined.

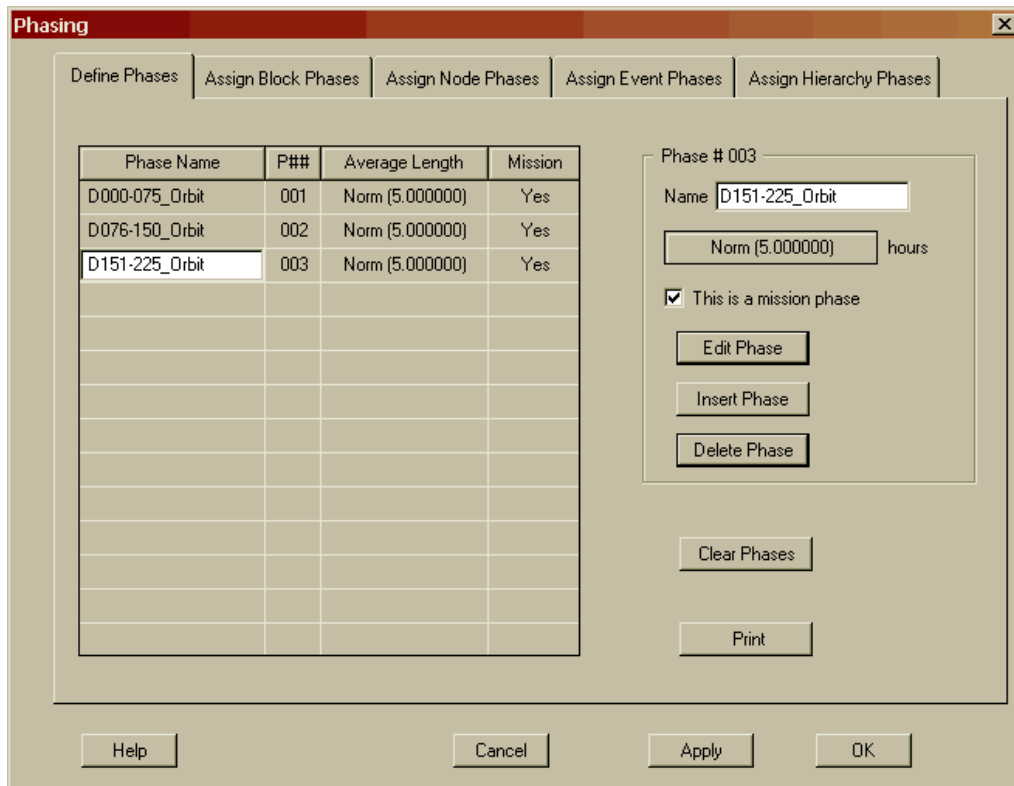


Figure 17.7 Initial Phasing Dialog Box

The upper right section of the Define Phases tab allows for the three attributes of a phase to be defined. A phase can possess a name, a fixed or random length and a designation as to whether the phase is considered a part of a mission or not. Our phases shall be named according to the space station’s location as it orbits the Earth. Thus, a phase name of “D000-075_Orbit” designates that the space station is within the first seventy-five degrees of its orbit. As a reference, the line intersecting the sun and the Earth’s centroids shall be considered the zero degree line. The length of the first four phases will be based on random draws from a normal distribution with a mean of 5.0 hours and a standard deviation of 0.1 hours. The last phase will possess a length that is defined by the normal distribution with a mean of 4.0 hours and a standard deviation of 0.3 hours. Thus, the sum of the five phases will be

approximately equal to the length of a day. The combined length of all phases defined within a model is known as its “cycle”. Once Raptor reaches the end of a cycle (i.e., the end of the last defined phase) and there exists additional simulation time, Raptor will loop back to the first phase and continue this pattern until a simulation trial is completed.

Edit the current dialog box such that it matches the one shown in Figure 17.8 (i.e., add the fourth and fifth phases). Notice that the fifth phase is the only one designated as a non-mission phase. We will discuss more about this issue later in this chapter but for now we are stating that phase five is not as important as the other phases. Be sure to click on the Insert Phase button after you define all of the properties of phase four and then again for phase five. If you make a mistake while entering the data shown in Figure 17.8, select the errant phase and use the Edit Phase button to correct the mistake.

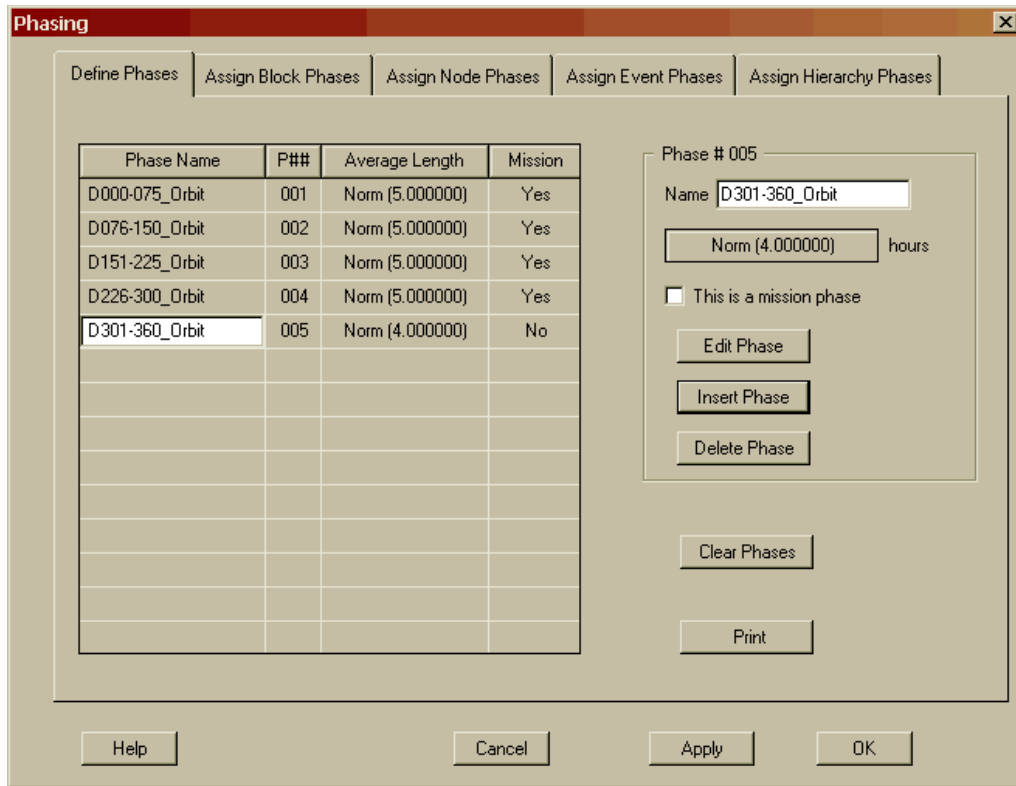


Figure 17.8 Phases Defined

Now that the day and the orbit of the space station have been divided into five phases, we can modify the key attributes of the elements that have been declared to be phased. Select the Assign Block Phases tab and set the attributes of the components using the information in Figure 17.9 such that your Phasing dialog box appears as the one shown in Figure 17.10. Only the cells in Figure 17.9 that are highlighted need to be changed from the dialog box default values.

	P001	P002	P003	P004	P005
BatteriesNiCd	C:0.500	A:1.000	A:1.000	A:1.000	A:1.000
Bus_UR	C:0.500	A:1.000	A:1.000	A:1.000	A:1.000
SolarP_500W	C:0.500	A:1.000	A:1.000	A:1.000	A:1.000
Tether-A	A:1.000	A:1.000	A:1.000	A:1.000	L:1.000
Tether-B	A:1.000	A:1.000	A:1.000	A:1.000	L:1.000
Thermal	A:1.000	A:1.500	A:1.500	A:1.000	A:0.500

Figure 17.9 Block Phasing Data

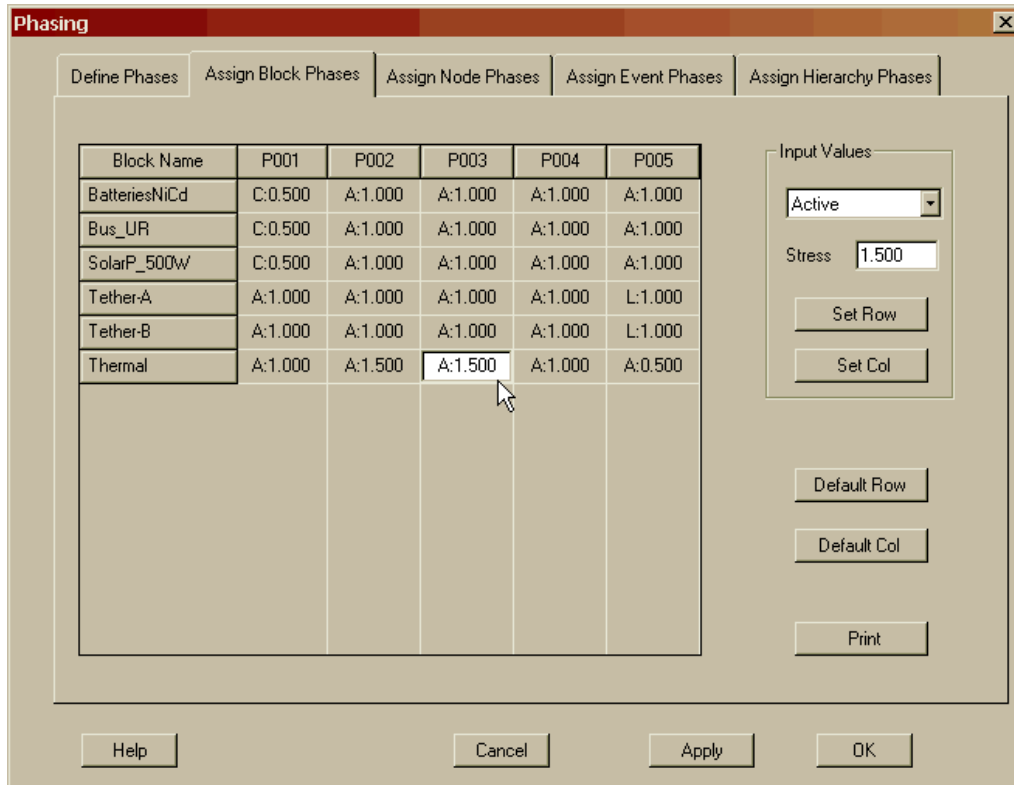


Figure 17.10 Block Phase Attributes

The “L” code used in Figure 17.10 represents a component that is linked in a particular phase. A linked component is not considered to be a part of the system. A linked component’s localized path within the RBD is by definition deemed to be a good path regardless of the state of the component itself. The “C” code represents a component that is cut from a particular phase. A cut component is not considered to be a part of the system. A cut component’s localized path within the RBD is by definition deemed to be a bad path regardless of the state of the component itself. While there is no difference with respect to the component itself between being linked or cut, there is a significant difference with respect to the system. A link is a good path for the system while a cut is a bad path. The “A” code represents an active component, which is by definition not a linked or cut component. An active component will progress through its life cycle of failing and repairing. All of the components utilized in the previous models were active components.

The link, cut and active codes possess a number after them that represents the stress on a component during a particular phase. Nominal or no stress condition is equal to 1.000. Stress values can range from 0.000 to 999.999. A component with a value of zero is under no stress and will not accumulate any life. A component with a stress greater than zero but less than one is under-stressed and accumulates life at a slower rate than normal. A component with a stress greater than one is over-stressed and accumulates life at a faster rate than normal.

As indicated in Figure 17.9, a user can interpret the Thermal component to be under a stress of 1.0 during phases one and four, a stress of 1.5 for phases two and three and a stress of 0.5 during the last phase. This stress profile represents the stress on the Thermal component as it proceeds through its orbit. As shown in Figure 17.11, this component's stress is a function of its orbit location due to the intensity of solar radiation. That is, when the space station is between the Earth and the sun, its thermal shield is under greater than normal stress. In contrast, when the space station is behind the Earth's shadow, the thermal shield is not stressed very much since the damaging solar radiation effects are lower.

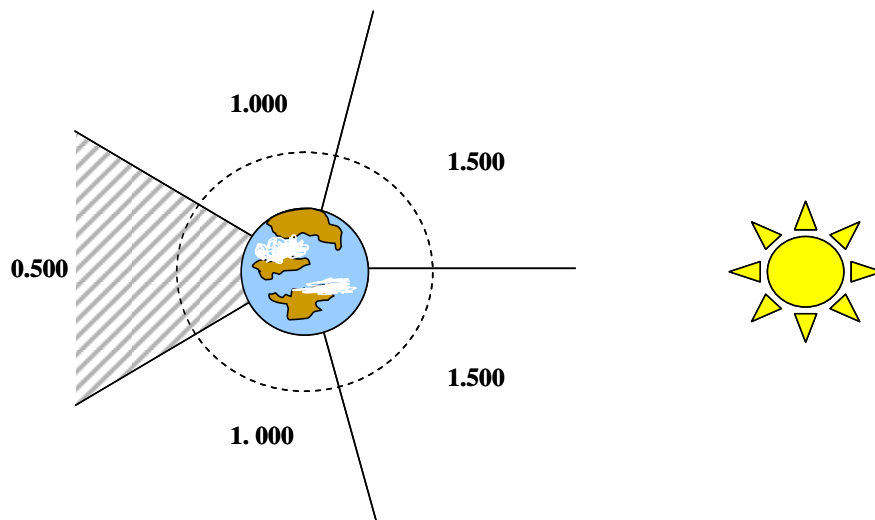


Figure 17.11 Thermal's Stress as Defined by the Orbit Location

The Tether components operate normally during phase one through four but are not a part of the system during phase five since they are linked. This represents the situation that the tethers are not in use during the early morning and late hours of a day due to ground control restrictions. The three components of the 500path hierarchy are all cut (not available for use) during the first phase with a stress value that is half of its normal value. These stress values affect the mean life of a component by the following relationship:

$$\theta_{resulting} = \frac{\theta_{original}}{stress}$$

This equation states that the mean life of a component for a particular phase (i.e., resulting mean life) is its original mean life (i.e., the mean of the component's failure distribution) divided by the stress defined for a phase. For example, a 100 hour component with a stress of two is going to act like a component with a mean life of 50 hours and will fail twice as often. A 100 hour component with a stress of 0.5 is going to act like a component with a mean life of 200 hours and last twice as long. The astute reader should realize that a component designated as A:1.000 across all defined phases is equivalent to a non-phased component. It is fundamentally imperative that a user understands that stress values only affect those components that are running (i.e., displaying one of the three green shades).

During a simulation, a phased component that is functioning may display the colors shown in Figure 17.12 based on the stress values defined for this component. The Thermal component is such a component and when we eventually simulate this model, be sure to notice the color changes it undergoes.




Stress Type	Stress	Color
Nominal	$\pi_f = 1$	
Under-stress	$0 \leq \pi_f < 1$	
Over-stress	$\pi_f > 1$	

Figure 17.12 Component Functioning State Colors

A linked component will be displayed with a white square overlaying the component. In contrast, a cut component will be displayed with a black square overlaying the component. Although a linked or cut component will possess an overlaying square, the component's state can still be determined. As shown in Figure 17.13, the linked and cut scenarios are considered to represent components that are offline and thus not a part of a system.

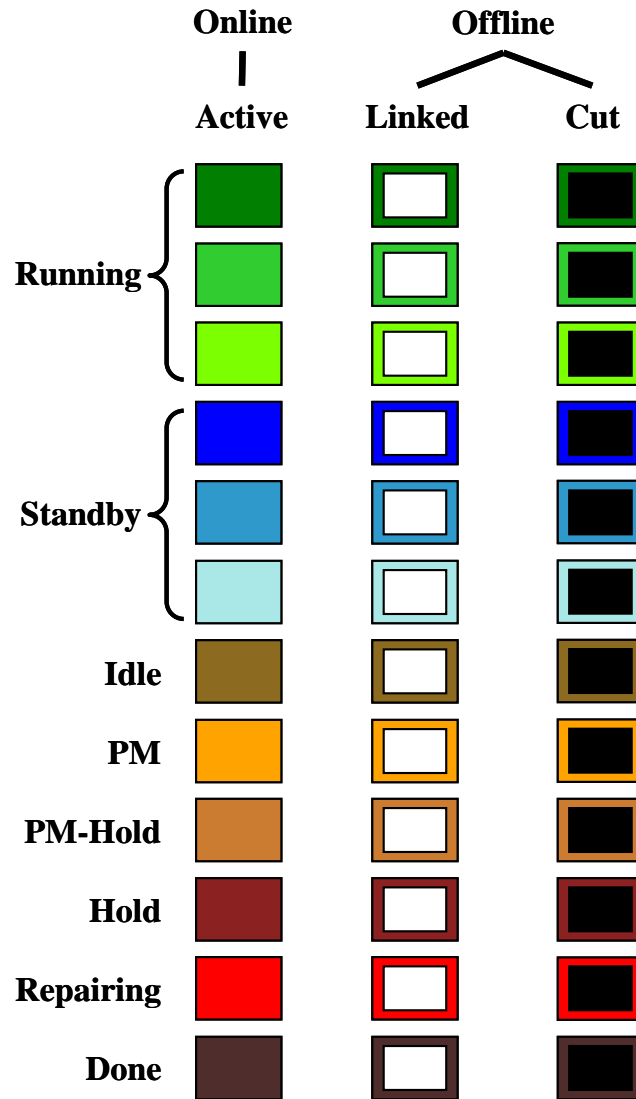


Figure 17.13 Link and Cut State Colors

Notice that a cut or linked component can still be in a failed state, a standby state or any other state of an active online component.

Let us now progress to discussing the next type of phased component. Select the Assign Node Phases tab of the Phases dialog box. Recall that the only attribute of a node that can vary by phase is its k -value and it can range from one to n . As indicated in Figure 17.14, the base k and n values are shown in the second and third columns, respectively and are provided as a reminder of the node's k -out-of- n attribute when the node is not phased. Fill in this tab such that it mimics Figure 17.14 by specifying the Power node's k -value to be a value of one for the first and last phases and a value of two for the other phases. We have now defined the Power node to be a 1-out-of-5 redundant node during one portion of the day and a 2-out-of-5 redundant node during the other portion. This represents the varying need for electricity during different phases of a day.



Figure 17.14 Node Phase Attributes

Similar to a component, a node can be cut or linked. A linked node is in essence a *0-out-of-n* redundant node, which implies that no paths are needed so the node and its entire domain are not a part of the system and thus not needed. A cut node is fundamentally an *(n+1)-out-of-n* redundant node, which means that the node and its domain cannot fulfill the require *n-value* and thus creates a bad path for the system.

Select the Assign Event Phases tab of this dialog box and populate the event phasing table with the information shown in Figure 17.15 (recall that only the highlighted cells need to be modified). Notice that the second column states the base *p-value* for each event. As indicated in this figure, the base *p-value* for the Ground_Control event is 0.800 and for the Inspection event it is 0.940. The base *p-value* is the probability of firing successfully when the event is not phased; it is a reference point for defining an event’s phased firing information.

	P001	P002	P003	P004	P005
Ground_Control	C	F	P	P	P
Inspection	0.960	P	0.940	P	0.920

Figure 17.15 Event Phasing Data

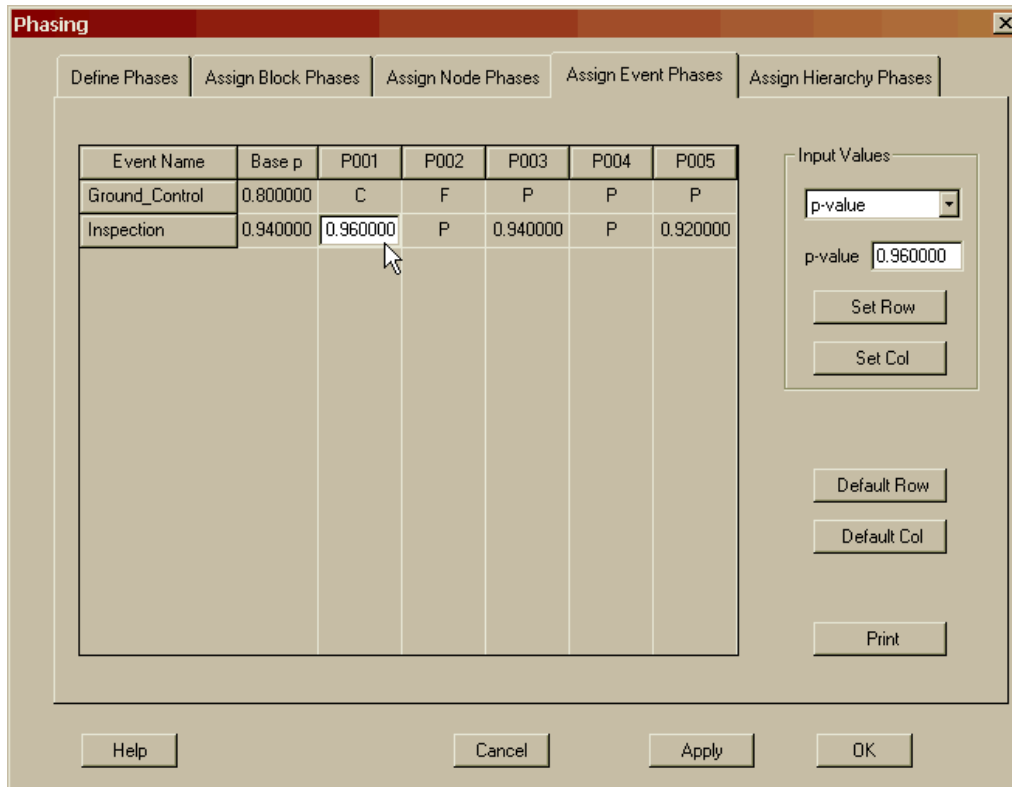


Figure 17.16 Event Phase Attributes

Let us first interpret the data entered for the Inspection event. This event will fire three times every day, drawing a different random number from a uniform real distribution that ranges from zero to one. Recall that a non-phased event can only be fired once at the start of each trial (i.e., simulation time zero). A phased event can be fired as many times as there are phases defined. The Inspection event possesses an additional level of detail in that its *p-value* is varying each time the event is fired. This may represent the variability in the inspection process itself. A non-phased event always fires with the same *p-value* for each trial of a simulation. The Inspection event also has been encoded with the letter “P” for “prior” in the second and fourth phases. An event that is marked as prior in a particular phase will take on the attributes of the previous phase that was fired. For example, if the Inspection event fires successfully in phase one, then this event will also be considered fired successfully in phase two. If the Inspection event misfires in phase three, then this event will also be considered a misfire in phase four. Consecutive prior settings always refer back (i.e., to the left) to a previous phase that has been fired.

If the *p-value* does not change from the base *p-value* during a particular phase, a letter “F” code can be used to fire an event. For the Ground_Control event, the F in phase two could have been replaced with a *p-value* of 0.80 and there would be no change in this event’s behavior. An event can be cut or linked much like a component. A linked event will be displayed with a white diamond overlaying the event. In contrast, a cut event will be displayed with a black diamond overlaying the event. Although a linked

or cut event will possess an overlaying diamond, an event's state of fired, misfired or armed can still be determined. The armed state of an event is one in which the event has yet to fire. The cut code marked in the first phase of the Ground_Control event represents a time when the tether power string is unavailable for use by the system. The tether equipment will not be employed during the early morning phase due to safety concerns of the ground control staff and therefore cannot provide tether power to the system during this phase.

If it is decided not to use the tether equipment for a particular day (event misfires in phase two), then it is not used for the entire day since phase two is followed with three prior markings. Likewise, if the Ground_Control event fires successfully within a particular day, then the tether equipment can be utilized for the remainder of the day. Since the Ground_Control event is in series with the tether components, it will dictate whether the tether power string may produce electricity for the space station. Finally, select the last tab of this dialog box labeled Assign Hierarchy Phases. Modify you tab to appear as the one shown in Figure 17.17. The 500path hierarchy is cut in phase one, linked in phase five and active for all of the other phases.

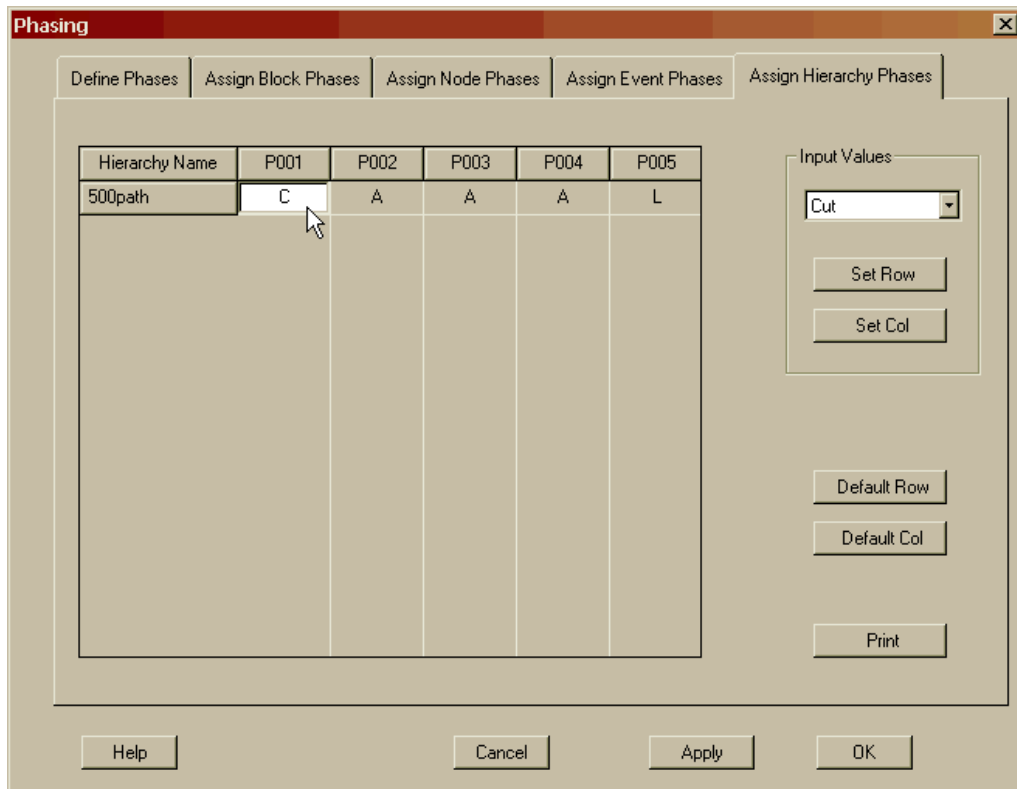


Figure 17.17 Hierarchy Phase Attributes

A hierarchy's phase attributes are limited since the hierarchy merely represents a subsystem. A hierarchy can be active, cut or linked. An active hierarchy is nothing more than a non-phased hierarchy. It monitors all its descendants then expresses the overall effect of their states by changing the color of its body and possibly its DSI. A

cut hierarchy represents an entire subsystem that is not a part of the system and represents a bad localized path. A linked hierarchy represents a subsystem that is not a part of the system but represents a localized good path. A hierarchy will change the color of its DSI to indicate whether it is linked or cut. As shown in Figure 17.18, a black DSI represents a cut hierarchy while a white DSI represents one that is linked.



Figure 17.18 Phased DSI Colors

Be sure to click the OK button of this dialog box to save all the changes that you have made. Return to the Workspace View and notice again that all phased elements are hatched as shown in Figure 17.19. Recall, that if phases are defined, then any element marked to be phased will be displayed as hatched within the Workspace View. This hatching helps the user to distinguish the dynamic elements from the static ones. Recall that hatching implies a different concept of induced dependency in Failure Effects View.

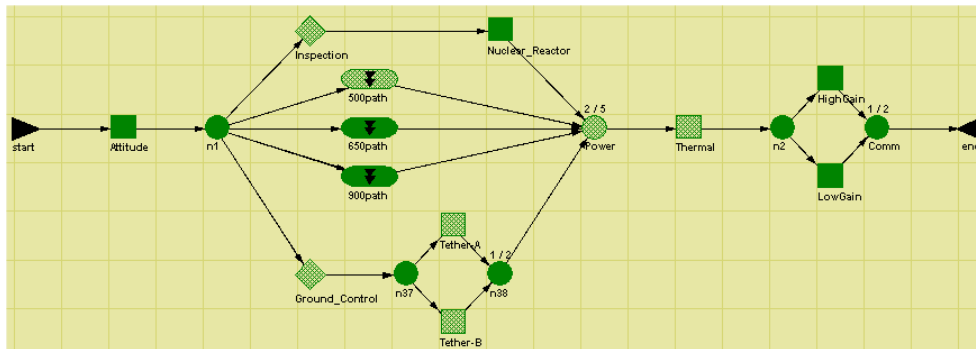


Figure 17.19 Phased RBD within the Workspace View

Before we begin to simulate this model, let us recall that we specified the fifth phase to be a non-mission phase. The mission designator indicates whether the user considers a particular phase to be critical or not. By this, we mean that a failure during a mission phase is more critical to the operations of the space station than one during a non-mission phase. Failures during non-mission phases are still a concern but not as much compared to those failures occurring during a mission phase. For example, suppose a single engine aircraft lost the use of its engine during a non-mission phase of pre-flight inspection. The engine failure would bring the system down and cause a mission abort but it is not as critical as a failure of the engine during flight. Raptor delineates between mission failures and non-mission failures by computing two distinct readiness parameters for a system. First, the system-level availability is determined without regards as whether a failure occurs during a mission or non-mission phase. That is, all failures effect the system-level availability calculation. This is the traditional availability definition we have been using throughout this workbook. The second readiness parameter is called dependability and is only concerned with system-level failures that occur during mission phases. The system-level dependability

parameter measures the effect of system failures while the system is actually employed to perform its critical tasks as oppose to its mundane tasks. The definitions of the system-level availability and dependability parameters are show below:

$$A_o = \frac{\mathit{uptime}_{(mission+non-mission)}}{\mathit{uptime}_{(mission+non-mission)} + \mathit{downtime}_{(mission+non-mission)}}$$

$$A_o = \frac{[\text{😊😊😊😊}]}{[\text{😊😊😊😊}] + [\text{😞😞}]}$$

$$D_o = \frac{\mathit{mission\ uptime}}{\mathit{mission\ uptime} + \mathit{mission\ downtime}}$$

$$D_o = \frac{[\text{😊😊}]}{[\text{😊😊}] + [\text{😞}]}$$

Let us simulate this system and notice that within the Simulation View, the phased elements are displaying the phase attributes as indicated in Figures 17.20 through 17.24.



Figure 17.20 Thermal Component's Phase Appearance

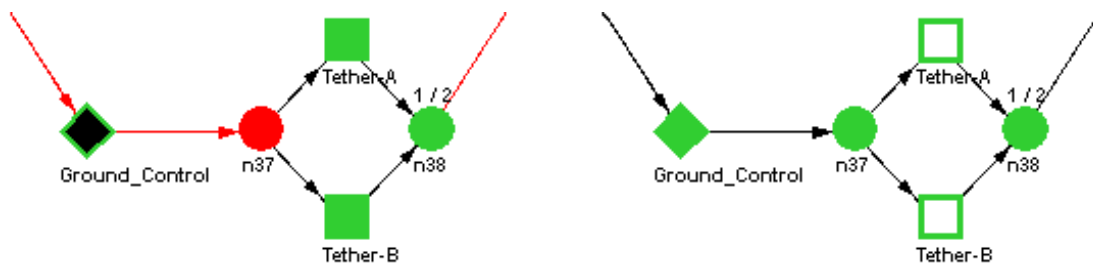


Figure 17.21 Tether and Ground_Control Elements' Phase Appearance

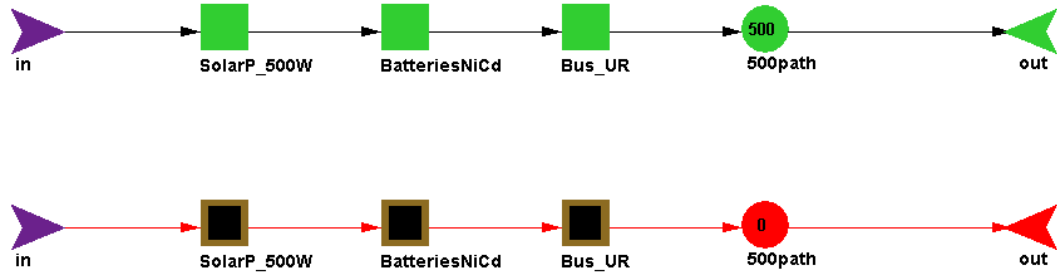


Figure 17.22 500 Watt Power String Components' Phase Appearance

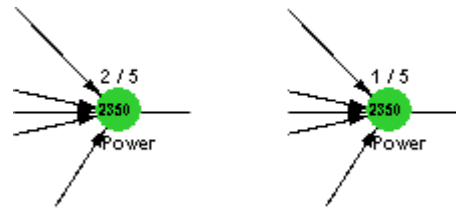


Figure 17.23 Power Node's Phase Appearance

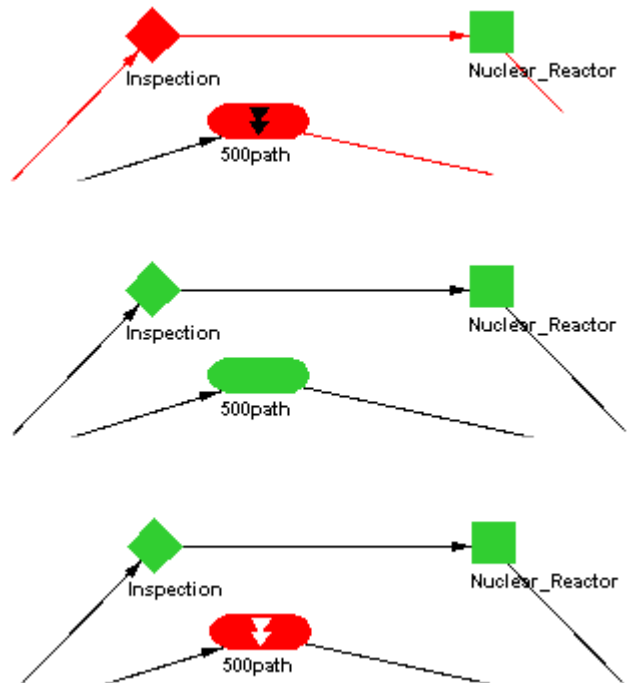


Figure 17.24 Inspection and 500path Hierarchy Elements' Phase Appearance

During a simulation, note that the status bar at the bottom of the Simulation View indicates not only the cumulative results, the percent completion of a trial and the trial number, but also the phase currently being simulated and whether it is a mission phase or not. Figure 17.25 is a partial view of the Simulation View's status bar during the simulation of the current model.

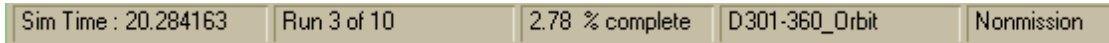


Figure 17.25 Phased Status Bar Information

After each phase is simulated, the simulation cycles back to the first phase and this process repeats until the end of the simulation is reached. If all the phasing information was entered correctly, the simulation results should match those shown in Figures 17.26 and 17.27.

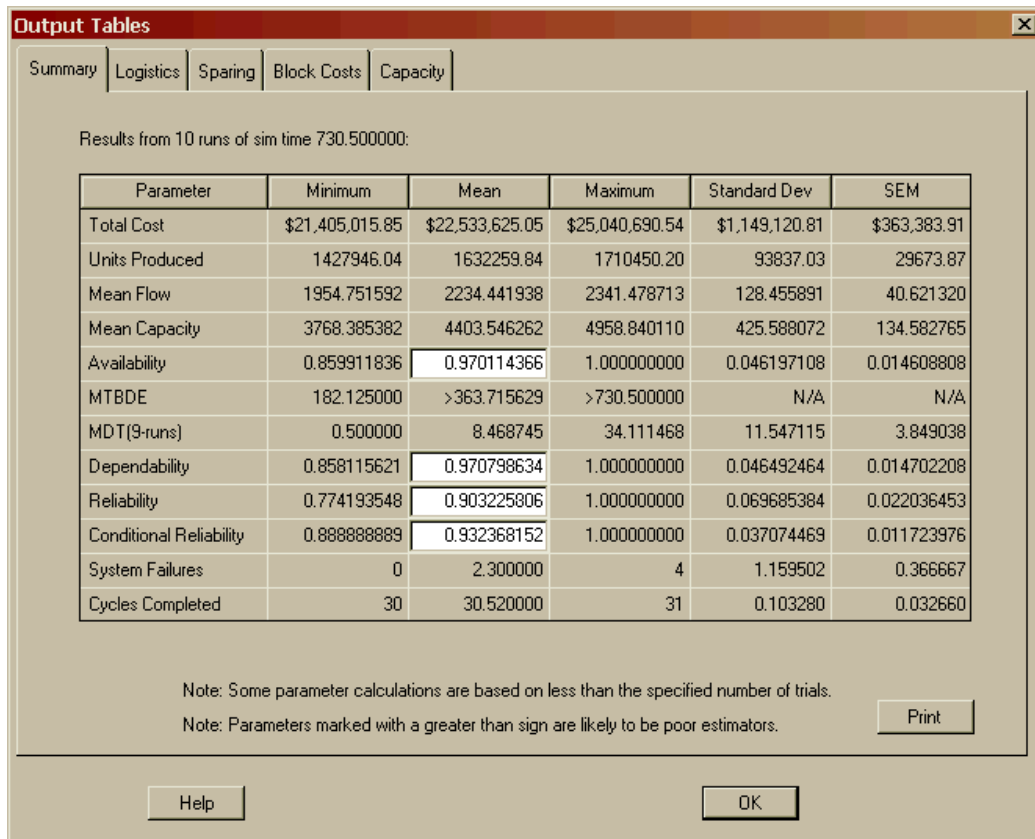


Figure 17.26 Phased Summary Results

While the availability and dependability values are very similar they differ as a result of any system-level failures occurring during the fifth phase; the only non-mission phase.

For a phased simulation, reliability is no longer either a one or zero for a particular trial. Reliability is now the number of missions successfully completed divided by the

number of missions attempted. A mission is a collection of contiguous mission phases. For this model the mission length is approximately twenty hours long and is comprised of phases one through four (i.e., all mission phases that are contiguous). The beginning of phase five is the end of one mission and the end of phase five is the beginning of another. Within the simulation length, approximately 31 missions are performed. The conditional reliability parameter is new and only appears for phased and delayed statistics gathering simulations.

The conditional reliability parameter is determined by dividing the number of missions started in an up state divided by the number of these types of missions that were attempted. That is, since phase five is the only non-mission phase, if the system was up at the end of phase five and then completes phases one, two, three and four without a system-level failure, then this mission has a reliability of one. If coming off the end of phase five the system was down, then the next mission does not count towards the conditional reliability parameter, although it counts against the reliability parameter. If at the end of phase five the system is up but cannot make it all the way to the beginning of the next phase five, then that mission has a reliability of zero. All of these mission reliabilities are averaged together to produce the mean reliability parameters shown in Figure 17.26.

Results from 10 runs of sim time 730.500000:

Parameter	Minimum	Mean	Maximum	Standard Dev	SEM
MTBM	21.616844	25.148449	30.437500	2.792759	0.883148
MTBMu	47.865869	65.987639	121.750000	21.924416	6.933109
MTBMs	34.898089	42.364522	54.832067	5.402899	1.708547
MMT	12.790444	20.005087	23.406685	3.089975	0.977136
MMTu	39.751897	47.636799	58.692946	6.122559	1.936123
MMTs	3.383155	3.658125	3.873069	0.173346	0.054817
Green Percent	4.098107	16.602704	41.556277	10.494087	3.318522
Yellow Percent	58.443723	80.408733	95.628108	11.031113	3.488344
Red Percent	0.000000	2.988563	14.008816	4.619711	1.460881
Mission Green Percent	4.064143	15.642849	39.492879	9.911237	3.134208
Mission Yellow Percent	60.507121	81.437014	95.772110	10.654890	3.369372
Mission Red Percent	0.000000	2.920137	14.188438	4.649246	1.470221

Figure 17.27 Phased Logistics Results

Figure 17.27 shows that a mission red, yellow and green percentage can now be calculated. These values are very similar to the all-encompassing red, yellow and green percentages since there is very little non-mission time in this model. Let us run one more phased model but we shall change the termination conditions. Open **Phasing3.rbd** and begin the simulation by opening the Simulation Options dialog box. As shown in Figure 17.28, we are going to terminate this system after 100 cycles have been simulated. Recall that one cycle is the sum of all defined phases, both mission and non-mission phases.

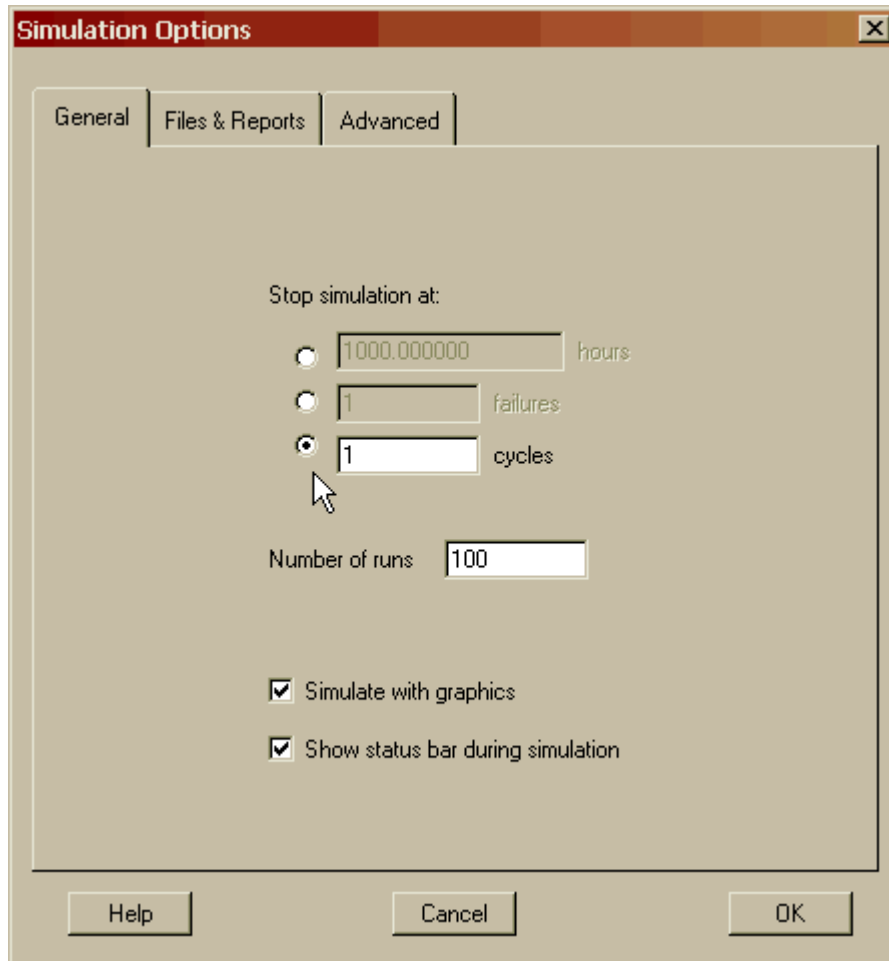


Figure 17.28 Cycle-Terminated Simulation Option

This type of simulation will proceed to simulate the five phases of this model and then stop the current trial and begin another trial. Thus, this simulation uses one hundred cycles to assess just the first orbit or day of operations. The results are shown in Figure 17.29. As indicated in this figure, the ending simulation time is 24.02 hours, which represents the average cycle length. Recall that the cycle is not exactly 24 hours since the cycle is comprised of random phase lengths. The reliability and conditional reliability are identical merely because the cycle length is so short. This means that we did not simulate enough cycles to see a failure that occurred in phase five and that was not fixed by the time phase one started.

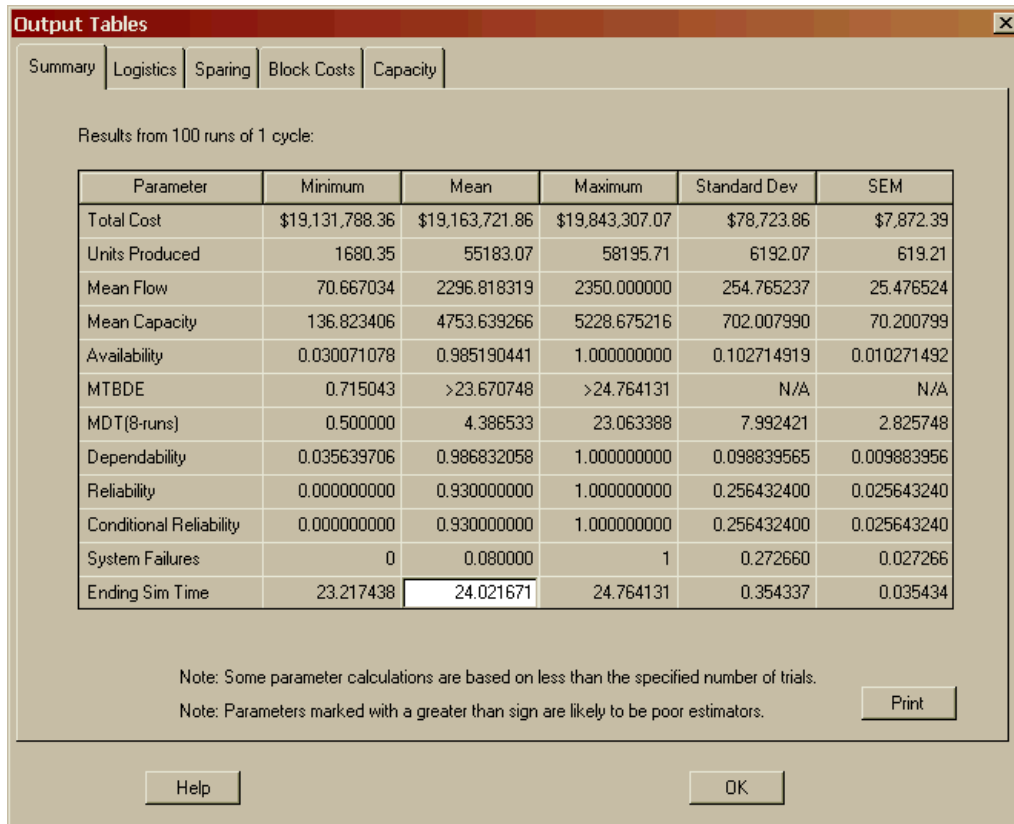


Figure 17.29 Cycle-Terminated Phased Results

Problems

1. Can failure-truncated simulations be phased?
2. What is the stress value for components that are not phased?
3. If a component has an average mean life of 100 hours and its stress is 5.00 in a particular phase, what value best describes its mean life during this phase?
4. If a component has an average mean life of 100 hours and its stress is 0.20 in a particular phase, what value best describes its mean life during this phase?
5. If a component has been stressed such that its mean life is 50 hours during a particular phase, does this mean that the component will fail 50 hours after the initiation of this phase? Why or why not?

6. A component's mean life is 100 hours. Let us say that a particular random draw resulted in the component's time to first failure to be 120 hours. The component enters phase two having not failed during the 40 hours of phase one in which it was under a stress of one. The component is designated with a stress value of four during phase two. Phase two lasts for 80 hours and then the simulation is cycled back to phase one. Will the component's first failure occur during phase one or two and at what simulation time value?

7. What does a node that is linked represent?

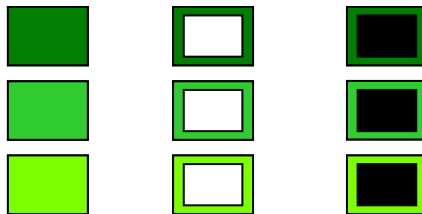
8. Explain the state of an event for each of seven phases if it was designated as follows:

F(*p-value* = 0.90) P P F(*p-value* = 0.50) P L P

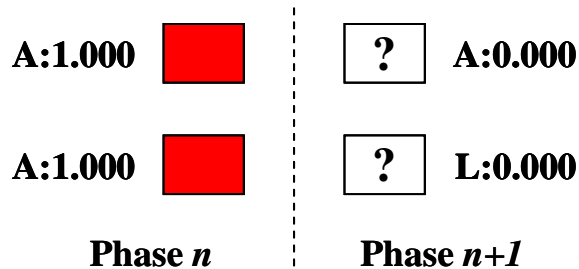
Assume the first time that an event fires it does so successfully, but the second time it fires failed.

9. Are ongoing repairs completed when a component enters a new phase that does not include this component (i.e., the component is either linked or cut)?

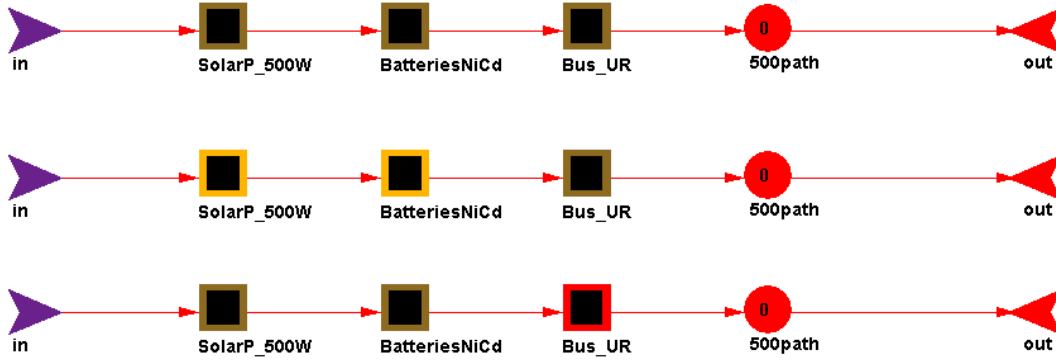
10. A component has a running cost of \$45/hour, a cut stress of 2.0 and a linked stress of 0.5. What is the per unit time cost for each of the nine conditions a running block can be in during a phased simulation?



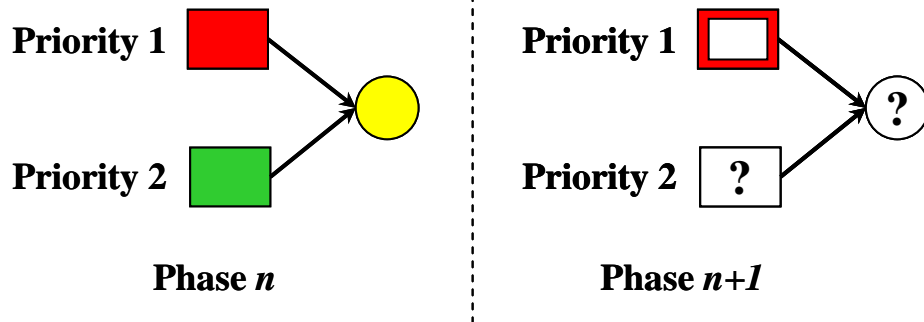
11. Two identical components that are actively repairing in a particular phase traverse into a new phase. The first component is set to A:0.000 and the second to L:0.000. Describe the condition of both components as they enter the new phase. Be sure to state which component creates localized good or bad paths and which component is accumulating life during this new phase.



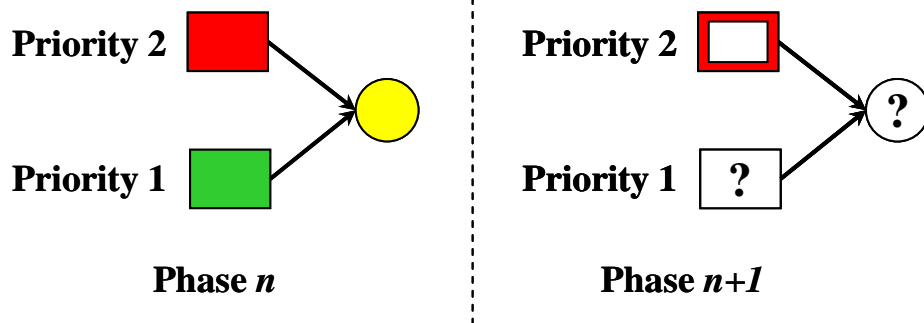
12. Three different phased conditions for the elements of the 500path hierarchy are shown below. Each case occurred when the 500path hierarchy was cut during a phase of the simulation of the **Phasing2.rbd** file. Develop a valid explanation as to why the components are displayed as the figure indicates.



13. What is the color of the blank elements shown below if the node is a standby node with the first component having a priority of one and the second a priority of two?



14. What is the color of the blank elements shown below if the node is a standby node with the first component having a priority of two and the second a priority of one?



15. Why is the total cost shown in Figure 17.29 approximately eighty-five percent of the total cost of Figure 17.26 when the former represents a single day of simulation and the later more than thirty days of simulation?

16. Can the *k-value* of a standby node be changed during phasing?

17. What two methods could be implemented to ensure whole cycles are simulated with respect to a phased simulation?

18. For the phasing information shown below, what are the average cycle and mission lengths for a simulation of near infinite duration?

Name	Length (hrs)	Type
Phase 1	Expo(3000)	Non-mission
Phase 2	Fixed(1)	Mission
Phase 3	Fixed(0.3)	Mission
Phase 4	Fixed(0.3)	Mission
Phase 5	Fixed(0.3)	Mission
Phase 6	Fixed(1.1)	Mission
Phase 7	Fixed(2.5)	Mission
Phase 8	Fixed(2)	Mission
Phase 9	Fixed(0.5)	Mission
Phase 10	Fixed(2)	Mission

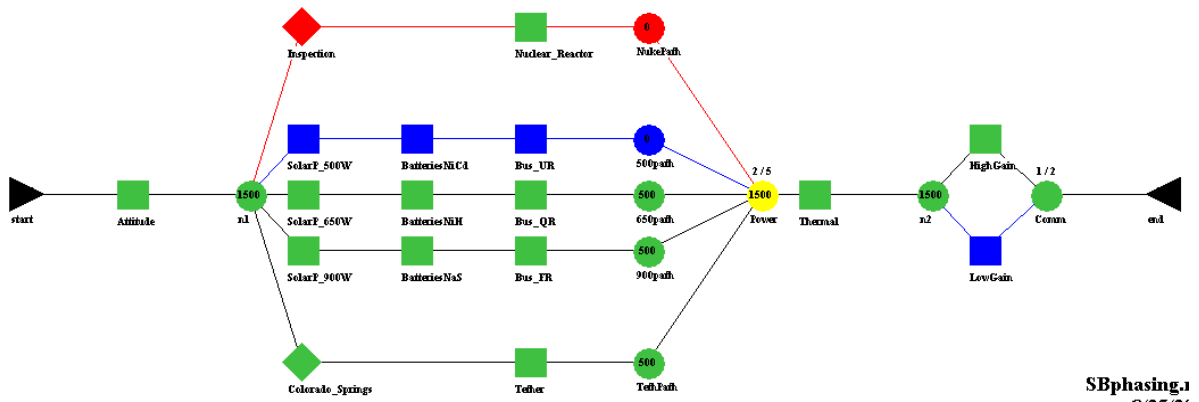
19. For the phasing information shown below, what are the average cycle and mission lengths for a simulation of 280 hours in duration?

Name	Length (hrs)	Type
Phase 1	Fixed(1)	Mission
Phase 2	Fixed(2)	Mission
Phase 3	Fixed(3)	Non-mission
Phase 4	Fixed(4)	Mission
Phase 5	Fixed(5)	Mission
Phase 6	Fixed(6)	Non-mission
Phase 7	Fixed(7)	Mission

20. For the phasing information shown below, what are the average cycle and mission lengths for a simulation of 300 hours in duration?

Name	Length	Type
Phase 1	Norm(5,0.01)	Mission
Phase 2	Expo(10)	Non-mission
Phase 3	Norm(5,0.01)	Mission
Phase 4	Norm(5,0.01)	Mission
Phase 5	Expo(10)	Non-mission

21. If the Power node is standby, and 2-out-of-5, why are there three paths up in the following diagram? How might you correct this?



SBphasing.1

22. What is the difference between setting a block in a particular phase to active with zero stress (A;0.00) and linked with zero stress (L;0.00)?

Chapter 18



Objectives

1. Understand how to set up resource pools.
2. Understand how to use the phasing feature.
3. Gain practical experience by modifying a block diagram to improve system performance.

Hands-On Example III

Congratulations, you have just been hired as the new Facilities Engineer for Rapco Electronics. Your predecessor was fired for incompetence after an audit revealed the company was spending much more than the industry average for environmental controls needed to produce the flash memory cards.

After conducting some initial research, you discovered that the most significant contributor to the environmental expense total was air conditioning. Last year, the company spent \$70,000 on operations and maintenance on the facility's five air conditioning units.

The air conditioning units (shown in Figure 18.1) all feed into a common plenum connected to the facility's duct system.

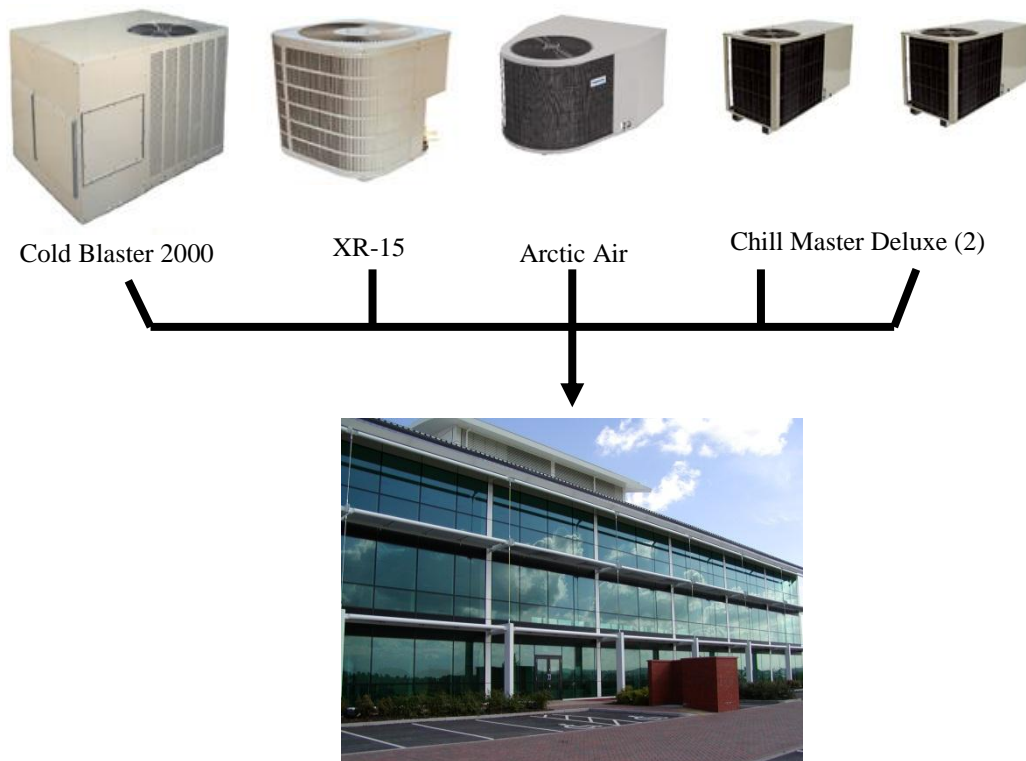


Figure 18.1 RAPCO Air Conditioning System

Currently, all of the air conditioners are configured to run as much as possible in order to insure the facility reaches the required average cooling capacity of 46,000 BTUs. Two facility repair technicians are available around the clock and there is no cost to use these personnel. Currently, there is no preventive maintenance program in place.

A. A review of available documentation (Figure 18.2) has revealed the following information on the current air conditioning equipment.

AC Unit Specifications				
Manufacturer	Model	Output (BTU)	MTBF (hrs)	MRT (hrs)
CMC Cooling	Cold Blaster 2000	25000	730	6
Fujii	XR-15	15000	1000	3
SimTec	Arctic Air	10000	1100	9
Dynastar	Chill Master Deluxe	5000	1500	5.75

Maintenance Policy	
Model	Policy
Cold Blaster 2000	Requires 3 CMC Cooling Techs, 24 delay due to travel time
XR-15	Repaired in house with 2 Rapco Techs
Arctic Air	Repaired in house with 1 Rapco Techs
Chill Master Deluxe	Repaired in house with 1 Rapco Techs

Sparing	
Model	Policy
Cold Blaster 2000	2 in stock, Order 2 when stock = 0
XR-15	2 in stock, Order 2 when stock = 0
Arctic Air	2 in stock, Order 2 when stock = 0
Chill Master Deluxe	2 in stock, Order 2 when stock = 0

Operation & Maintenance Cost Summary										
Unit	Hourly							Fixed		
	Running	Standby	Repair*	PM	Rep_Hold	PM_Hold	Idle	Spare	EmerShip	
ArcticAir_10000	\$1.00	\$0.10	\$30.00	N/A	\$1.00	\$1.00	\$0.00	\$100.00	\$20.00	
CB2000_25000	\$2.50	\$0.25	\$30.00	\$30.00	\$1.00	\$1.00	\$0.00	\$250.00	\$20.00	
CMD_5000	\$0.50	\$0.05	\$30.00	N/A	\$1.00	\$1.00	\$0.00	\$50.00	\$20.00	
CMD_5000	\$0.50	\$0.05	\$30.00	N/A	\$1.00	\$1.00	\$0.00	\$50.00	\$20.00	
XR-15_15000	\$1.50	\$0.15	\$30.00	N/A	\$1.00	\$1.00	\$0.00	\$150.00	\$20.00	

* This is the cost associated with materials needed for the repair other than the spare. CMC Cooling also charges \$30.00 per hour per person for service calls.

Figure 18.2 Conditioning Equipment Data

Open **Rapco.rbd** and complete the reliability block diagram. Only the resource and capacity data is missing. Run the RBD for five trials of one year (entering the results in Figure 18.3) and determine the baseline mean system throughput and average annual cost. Hint: The highlighted cells are the only ones that contain information not already in the RBD.

RESULTS	Avg. From 5 Trials
Mean Flow	
Total Cost	

Figure 18.3 Key Results from Rapco.rbd

B. The as is model should show an expected annual cost of \$70,012 for 58,926 average BTUs of output. Upon further research you uncover the following additional information.

Preventive Maintenance Options

- Preventive maintenance on any of the components will refresh the component like new.
- PM for the Cold Blaster 2000 takes 2 CMC personnel and takes three hours to complete; all other components take the same time and same resources as unscheduled repairs.
- Preventive Maintenance does require a spare.

Standby Options

- The Chill Master Deluxe units have the capability to be on standby mode with automatic instantaneous switching. The units accumulate life at 10% of their normal rate while in standby mode.
- None of the other units can be put into standby status.

Power Options

- SimTec sells a solar power unit specifically designed for the Arctic Air. When used with solar power, the Arctic Air can produce the same output at a cost of only \$0.10 per hour.
- The MTBF of the solar kit is 3,000 hours (scale) and its MRT is 4 hours. Assume it is Weibull distributed with a shape parameter of 2.1. The solar kit has the same failure stress rate regardless of the time of day.
- The solar kit costs \$1,500. The only cost associated with the solar kit is the initial acquisition.
- The solar unit provides power for twelve hours and then is off for twelve hours repeatedly.
- The solar unit will not work if it is cloudy in the morning, which occurs 5% of the time. If it is cloudy in the morning, it stays cloudy all day.
- The Arctic Air unit goes to an idle status without power.

Other Options

- Any of the units can be disposed of and there is no disposal fee.
- It takes three days to order a spare through normal ordering mechanisms. An emergency spare can be ordered in twenty-four hours.

Modify the baseline system to reduce the operating and maintenance cost while still meeting the overall building requirement of 46,000 BTU. Simulate the RBD for five trials of one year and determine the mean system throughput and average annual cost.

Chapter 19



Objectives

1. Understand the basic simulation algorithms for acquiring random numbers.
2. Understand the importance of an element's identification number.
3. Understand the simulation calendar.

Recommended Problems

1, 2, 4, 5, 7 and 8

Simulation Mechanics

Let us look at how Raptor uses random numbers to understand the underlying mechanics of the Raptor algorithms. We will use the simple RBD below to determine by hand what Raptor performs automatically.

Let us say that an RBD is comprised of three independent and identical components configured as shown in Figure 19.1. Each component fails using a uniform real distribution with a mean of 50 hours and repairs according to a uniform real distribution with a mean of 10 hours. This RBD uses the unrealistic failure and repair distribution of uniform real since this will make the effects of the random numbers easier to determine by hand. Let us assume that each component has infinite spares, zero-length logistic delays and does not need resources to complete a repair. Component A was placed into the Workspace View first, followed by component B and then by component C. Thus, they received identification numbers (i.e., ID#) one, two and three, respectively.

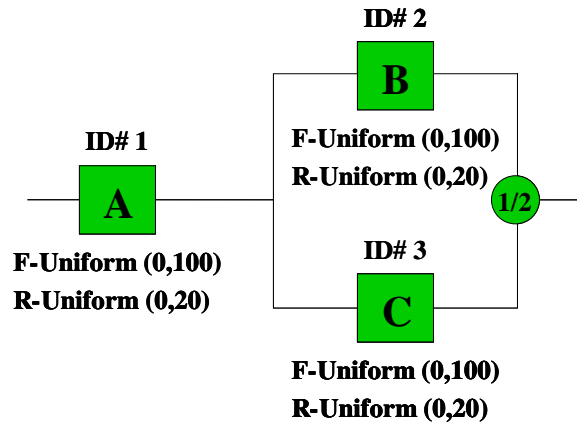


Figure 19.1 Example I Simplistic RBD

Each component will use system stream C. The first forty numbers of this system stream are shown in Figure 19.2.

1	2	3	4	5	6	7	8	9	10
0.5470	0.1898	0.5577	0.2294	0.7166	0.0864	0.2851	0.0139	0.5025	0.1144
11	12	13	14	15	16	17	18	19	20
0.6411	0.4695	0.2329	0.7211	0.9906	0.0584	0.7779	0.5647	0.2597	0.1944
21	22	23	24	25	26	27	28	29	30
0.8335	0.3364	0.0028	0.6427	0.1031	0.9010	0.6282	0.5399	0.8988	0.2446
31	32	33	34	35	36	37	38	39	40
0.0190	0.1607	0.9838	0.2560	0.9699	0.5378	0.6832	0.0444	0.8136	0.7453

Figure 19.2 System Stream C Random Numbers

Let us conduct a time-truncated simulation of 75 hours to determine which components will receive which random numbers and their effects on the system. Figure 19.3 delineates how the random numbers were assigned at the beginning of the simulation (i.e., simulation time equal to zero), and the order the components received the subsequent random numbers. Figure 19.3 also displays the random numbers rounded to the second decimal place to simplify the arithmetic involved in this illustrative example.

Sim time	Component	Random #	Fail	Repair	Will occur at	Events
0.0	A	0.55	55 hrs		55.0	55
0.0	B	0.19	19 hrs		19.0	19, 55
0.0	C	0.56	56 hrs		56.0	19, 55, 56
19.0	B	0.23		4.6 hrs	23.6	23.6, 55, 56
23.6	B	0.72	72 hrs		95.6	55, 56, 95.6
55.0	A	0.09		1.8 hrs	56.8	56, 56.8, 95.6
56.0	C	0.29		5.8 hrs	61.8	56.8, 61.8, 95.6
56.8	A	0.01	1 hr		57.8	57.8, 61.8, 95.6
57.8	A	0.50		10 hrs	67.8	61.8, 67.8, 95.6
61.8	C	0.11	11 hrs		72.8	67.8, 72.8, 95.6
67.8	A	0.64	64 hrs		131.8	72.8, 95.6, 131.8
72.8	C	0.47		9.4 hrs	82.2	82.2, 95.6, 131.8
***** Simulation Terminated at 75 hours *****						

Figure 19.3 How the Random Numbers Were Consumed for Example I

The time line shown in Figure 19.4 illustrates how events occur and graphically displays how Raptor schedules events as it proceeds through a simulation.

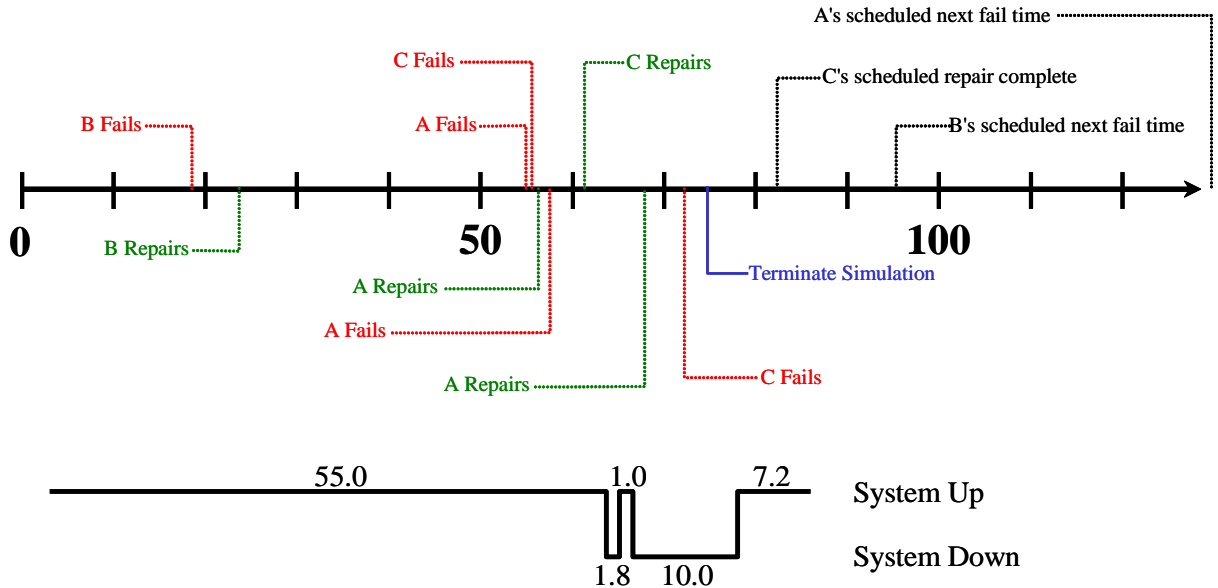


Figure 19.4 Example I Time Line

Component A gets the first random number and is placed on the simulation calendar at time 55 hours. Component B gets the second random number and component C gets the third random number. Component B is scheduled to fail at simulation time 19 hours and component C at 56 hours.

The earliest event on the calendar is the failure of component B. This component will fail and then seize another random number to determine its repair time. Component B's repair takes 4.6 hours and this repair is finished before any other event occurs on the simulation calendar. Thus, component B is now repaired and needs another failure time to be placed back on the simulation calendar. It consumes the fifth random number and gets placed on the calendar 72 hours later or calendar time 95.6 hours.

This process of components failing and repairing and only consuming random numbers when they are needed (i.e., new failure or repair times), is the basic simulation mechanics of the Raptor software.

Let us take another look at Raptor's simulation mechanics with an example that is somewhat similar to the previous example. The RBD shown in Figure 19.5 is again comprised of independent components that do not need spares or resources and use system stream C. Component B was placed within the Workspace View first followed by components A and then C.

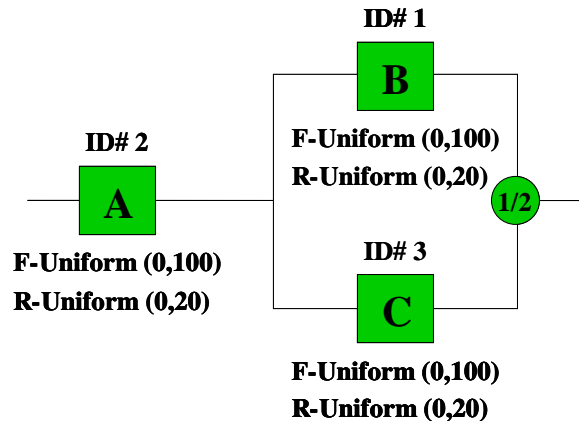


Figure 19.5 Example II Simplistic RBD

Let us conduct a time-truncated simulation of 75 hours to determine which components will receive which random numbers and their effects on the system. Figure 19.6 delineates how the random numbers were assigned at the beginning of the simulation (i.e., simulation time equal to zero), and the order the components received the subsequent random numbers. Figure 19.6 also displays the random numbers rounded to the second decimal place to simplify the arithmetic involved in this illustrative example.

Sim time	Component	Random #	Fail	Repair	Will occur at	Events
0.0	B	0.55	55 hrs		55.0	55
0.0	A	0.19	19 hrs		19.0	19, 55
0.0	C	0.56	56 hrs		56.0	19, 55, 56
19.0	A	0.23		4.6 hrs	23.6	23.6, 55, 56
23.6	A	0.72	72 hrs		95.6	55, 56, 95.6
55.0	B	0.09		1.8 hrs	56.8	56, 56.8, 95.6
56.0	C	0.29		5.8 hrs	61.8	56.8, 61.8, 95.6
56.8	B	0.01	1 hr		57.8	57.8, 61.8, 95.6
57.8	B	0.50		10 hrs	67.8	61.8, 67.8, 95.6
61.8	C	0.11	11 hrs		72.8	67.8, 72.8, 95.6
67.8	B	0.64	64 hrs		131.8	72.8, 95.6, 131.8
72.8	C	0.47		9.4 hrs	82.2	82.2, 95.6, 131.8
***** Simulation Terminated at 75 hours *****						

Figure 19.6 How the Random Numbers Were Consumed for Example II

The time line shown in Figure 19.7 illustrates how events occur and graphically displays how Raptor schedules events as it proceeds through a simulation.

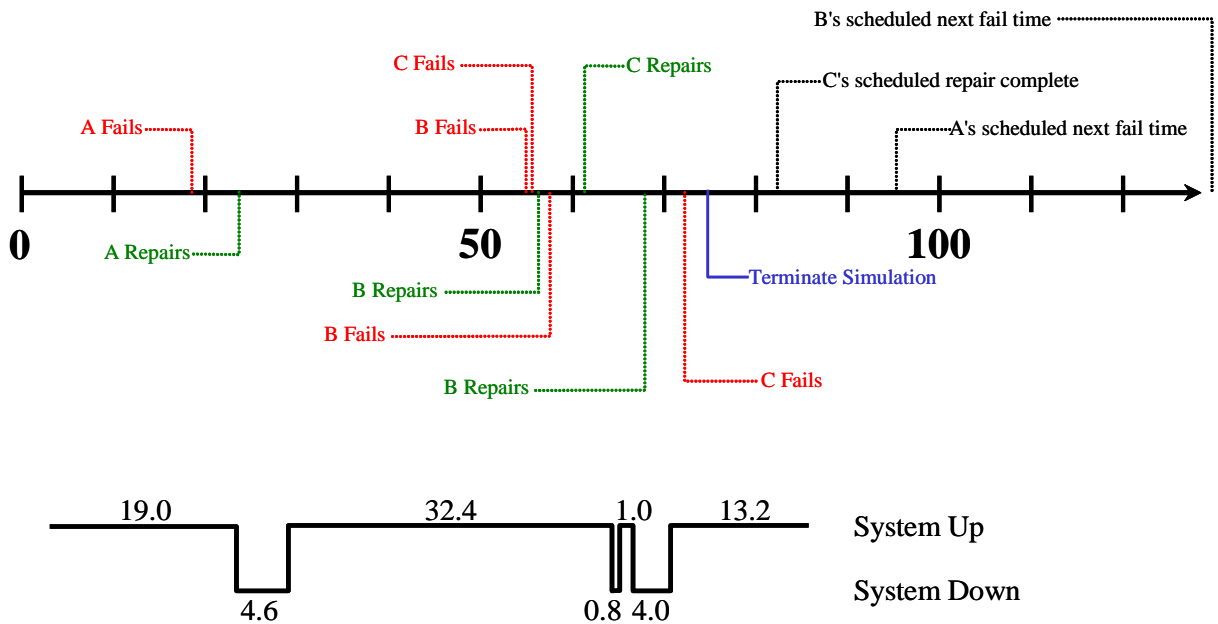


Figure 19.7 Example II Time Line

Component B gets the first random number and is placed on the simulation calendar at time 55 hours. Component A gets the second random number and component C gets the third random number. Component A is scheduled to fail at simulation time 19 hours and component C at 56 hours. Notice the differences between Figures 19.4 and 19.7 even though the same RBD appears to have been constructed in both examples. In the long term, the results of the first example and of the second example will converge upon truth, and hence upon each other. Essentially, if a sequence is formed

using the results of successively longer simulations of the two examples, this resulting sequence will be Cauchy. The convergence is assured because of the well known fact that Cauchy sequences converge in any complete metric space; intuitively we can interpret this by understanding that eventually the effects of any particular sample of random numbers will even out over long simulations of many trials.

Problems

1. Calculate the availability of the system with the event time line (i.e., simulation calendar) shown in Figure 19.3.
2. What are the MTBDE and MDT of the system described in Problem 1?
3. What is the percent yellow time of the system described in Problem 1?
4. What is the MTBDE of component B based on the data displayed in Figure 19.1?
5. Calculate the availability of the system with the simulation calendar shown in Figure 19.6.
6. What are the MTBDE and MDT of the system described in Problem 5?
7. What is the percent yellow time of the system described in Problem 5?
8. What is the MTBDE of component B based on the data displayed in Figure 19.5?

Chapter 20



Objectives

1. Understand how to set preferences.
2. Know how to mass edit.
3. Understand the purpose and implementation steps of libraries.
4. Understand the Print dialog box options.
5. Know how to find items in a large RBD.
6. Understand how to use the block growth and decay options.
7. Understand the purpose and implementation methods for choosing random number streams.
8. Know how to implement block defaults.
9. Understand the options available in the Simulation Options dialog box.

Recommended Problems

1, 2, 3, 4, 5 and 7

Miscellaneous Topics

This chapter covers miscellaneous features and options in Raptor that do not fit well into other chapters and do not themselves justify a chapter. The various topics are unrelated and are covered in their own section of this chapter. Additional help for these topics can be found in Raptor's help file.

Preferences Dialog Box

The Preferences dialog box is opened by selecting the *Options - Preferences* menu item from the Workspace View menu bar. It is comprised of two tabs in which the user preferred settings can be implemented. Preferences are saved in a separate file and are independent of any RBD files. Thus, preferences defined by this dialog box are global to all Raptor files.

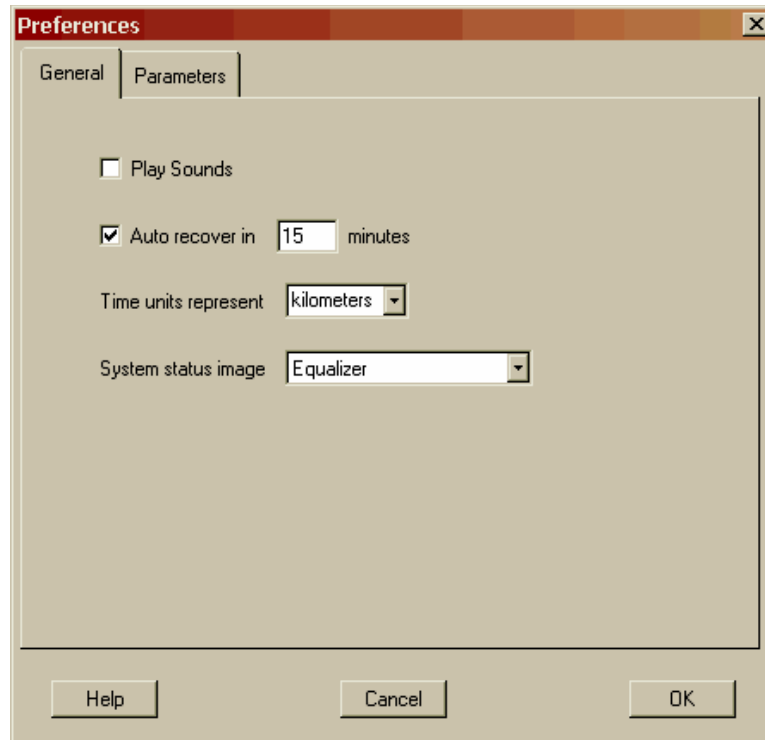


Figure 20.1 General Tab of the Preferences Dialog Box

The first tab is the General tab shown in Figure 20.1. A user can turn sounds on or off, turn the auto-save feature on or off, set the auto-save frequency, set the time units or select the system status indicator. Recall that Raptor is a unit-less tool and choosing a time unit merely places text fields of the chosen time unit on various dialog boxes. This helps a user to maintain consistency of data input by helping the user to remember which time unit is being used for a particular RBD. The determination of results by Raptor are unaffected by the selection of the time unit.

The second tab, as show in Figure 20.2 is for setting options related to the parameters of the exponential and lognormal distributions. The exponential distribution has two forms that are mathematically equivalent. One form uses the mean of the distribution as the input parameter and the other uses the failure rate, which is the reciprocal of the mean. Some reliability engineers prefer to work in failure rates when using the exponential distribution. When the option to enter failure rates is selected, a failure rate text field will appear on the Failure & Repair tab of the Block Properties dialog box instead of the traditional text field of the mean.

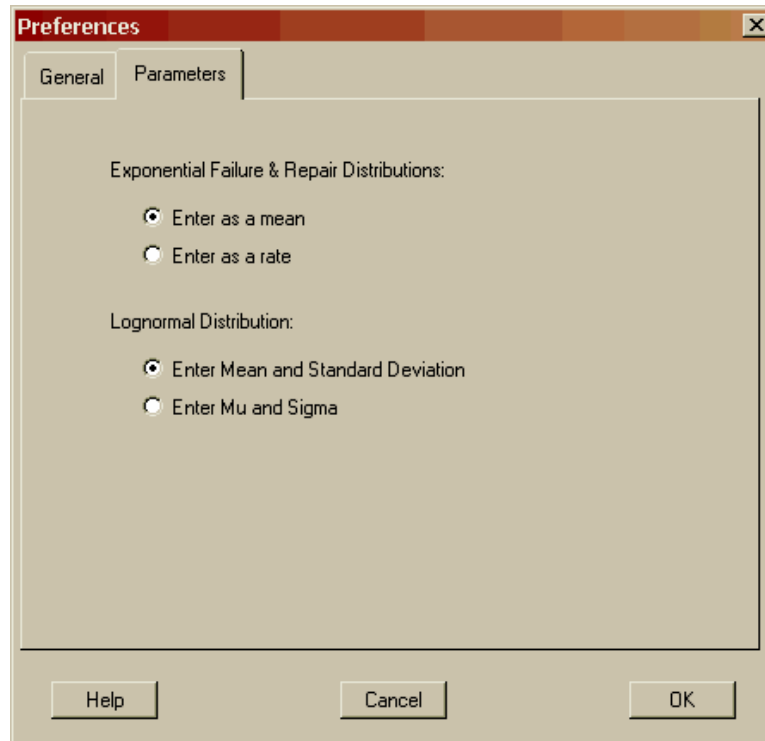


Figure 20.2 Parameters Tab of the Preferences Dialog Box

The lognormal distribution is another distribution whose input parameters can be defined in more than one way. Some people prefer to enter the mean and standard deviation of the sample data, or equivalently, the mean and standard deviation of the actual distribution. The lognormal distribution, however, is based on the normal distribution. That is, the log of data from a lognormal distribution is normally distributed. Another way to define lognormally distributed data is to define the mean and standard deviation of the data in the normal space. These values are referred to using the Greek letters μ and σ (i.e., mu and sigma). The probability density function equation for the lognormal distribution contains the parameters μ and σ and most curve fitting tools will provide analysts with the μ and σ values.

System Settings Dialog Box

The System Settings dialog box is opened by selecting the *Options – System Settings* menu item from the Workspace View menu bar. It is comprised of four tabs in which the system specific settings can be implemented. System settings are saved within the RBD file. Thus, settings defined by this dialog box are specific to a particular RBD.

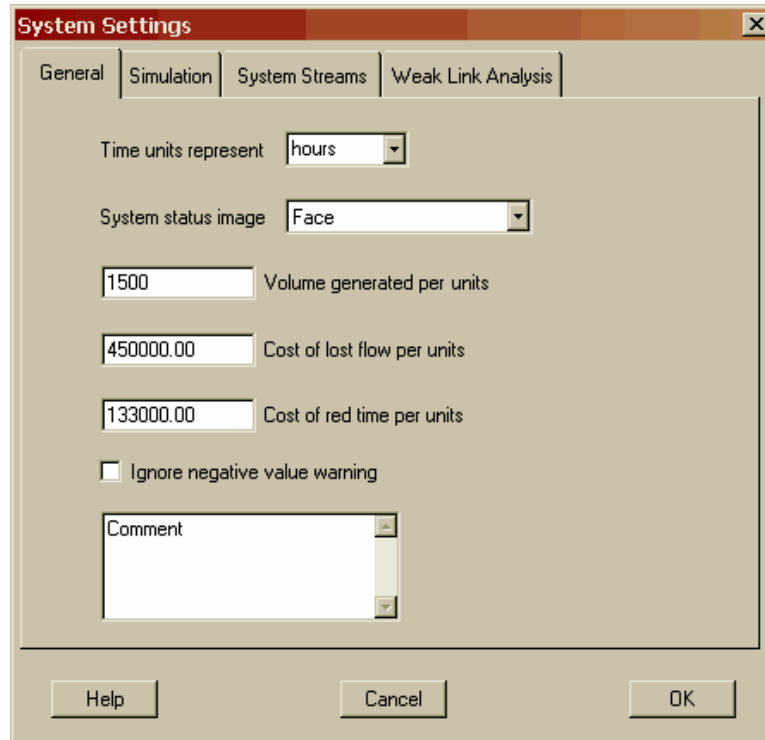


Figure 20.3 General Tab of the System Settings Dialog Box

The first tab is the General tab shown in Figure 20.3. The first two settings, time units and system status indicator, can be set here for a particular RBD, or globally through the Preferences dialog box. The volume generated, cost of lost flow, and cost of red time for the system are also set here. By checking the ignore negative value warning check box, the warning box that appears when negative values are drawn during simulation will not be displayed. A system comment can also be set.

The second tab, as shown in Figure 20.4, contains settings related to simulation. The simulation termination type (time, failure, or cycle) can be set, as well as the number of runs, the simulation time at which statistics gathering will start, and whether or not this particular RBD will be simulated with graphics.

The third tab (Figure 20.5) is for changing the designated system streams. By default, new components are tied to system stream C, which is seed number nine of the 109 built in random number streams of the underlying Raptor simulation language. In some cases, when numerous components are set to use a particular system stream and a user desires to change them all to another stream, it is easier to just change the

definition of the stream itself. A discussion of random number streams, including definitions for system streams and independent streams, and reasons why one might want to change streams is included in the Raptor help file.

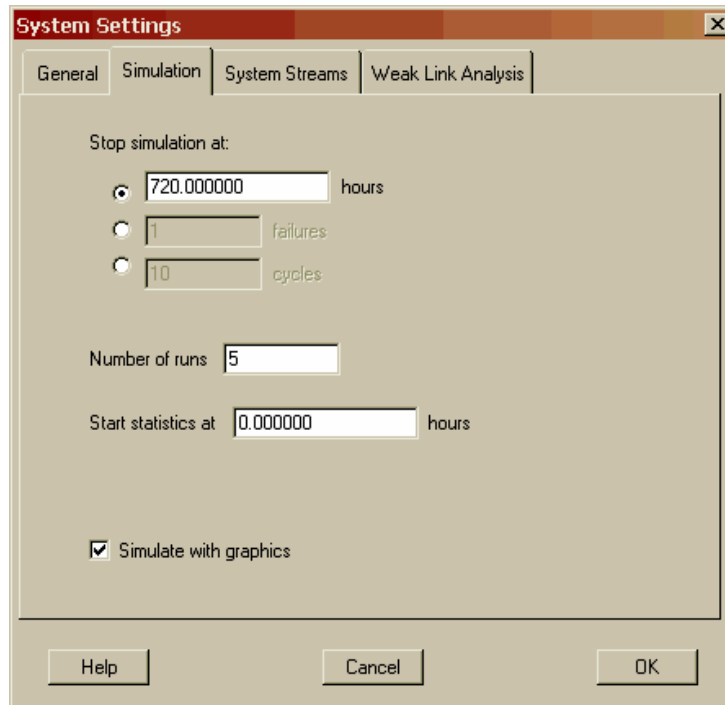


Figure 20.4 Simulation Tab of the System Settings Dialog Box

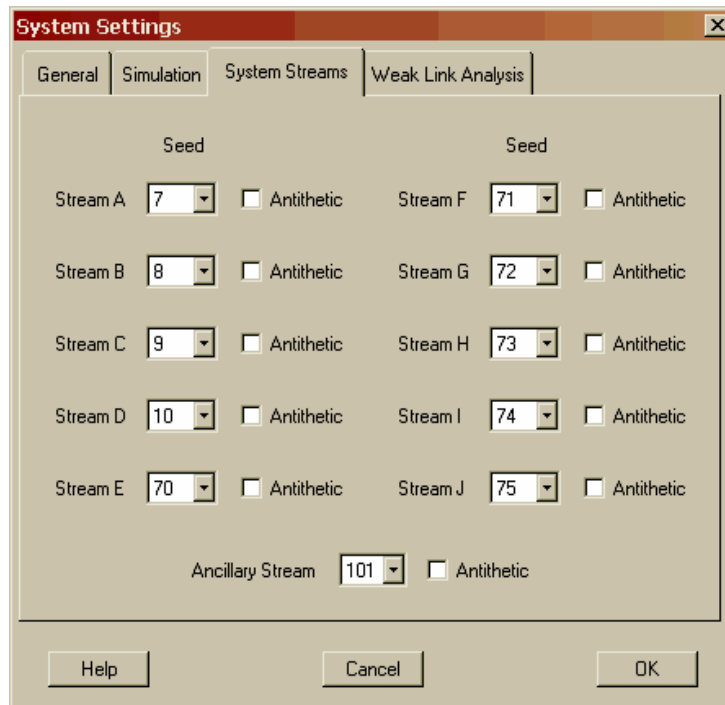


Figure 20.5 System Streams Tab of the System Settings Dialog Box

The final tab of the System Settings dialog box is the Weak Link Analysis tab (Figure 20.6). This tab is identical to the Weak Link Analysis Settings dialog box that can be accessed while in the Simulation or Weak Link Analysis view. The various options are explained in Chapter 11.

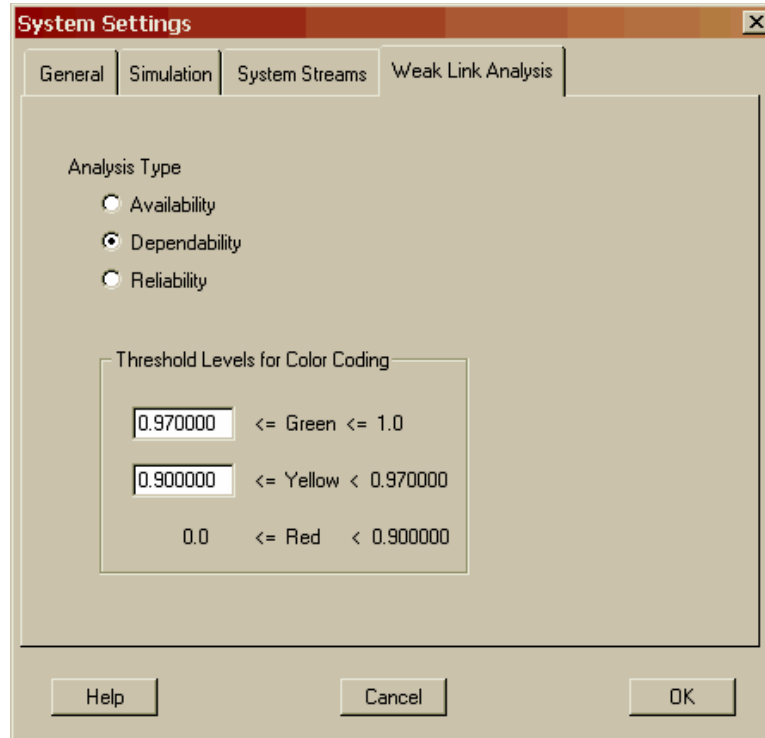


Figure 20.6 Weak Link Analysis Tab of the System Settings Dialog Box

Mass Edit Dialog Box

The Mass Edit dialog box is opened by selecting the *Edit - Mass Edit* menu item from the Workspace View menu bar or from the right click menu. This option is deactivated unless a group of blocks, nodes, events, or at least one hierarchy is selected. A group of elements can be highlighted using the rubber-band box or by selecting elements while holding down the control key. Selection of a hierarchy element automatically selects any objects contained within that hierarchy for edit as well. The dialog box displays between one and four tabs depending on the elements selected.

Figure 20.7 displays the Blocks tab. The phasing preference, dependency, block name, and system stream source can be set from this tab for all blocks selected. The block name can be altered in one of three ways. The existing block name can have characters appended as a prefix, characters appended as a suffix, or the block name can be replaced in its entirety. The option to have no repair, infinite spares, or no resource requirements can also be set. These options are valuable for running simulations to determine upper or lower bounds of logistics constraints. For example, the results obtained when all blocks have infinite spares represent the best possible result when various sparing schemes are being tested.

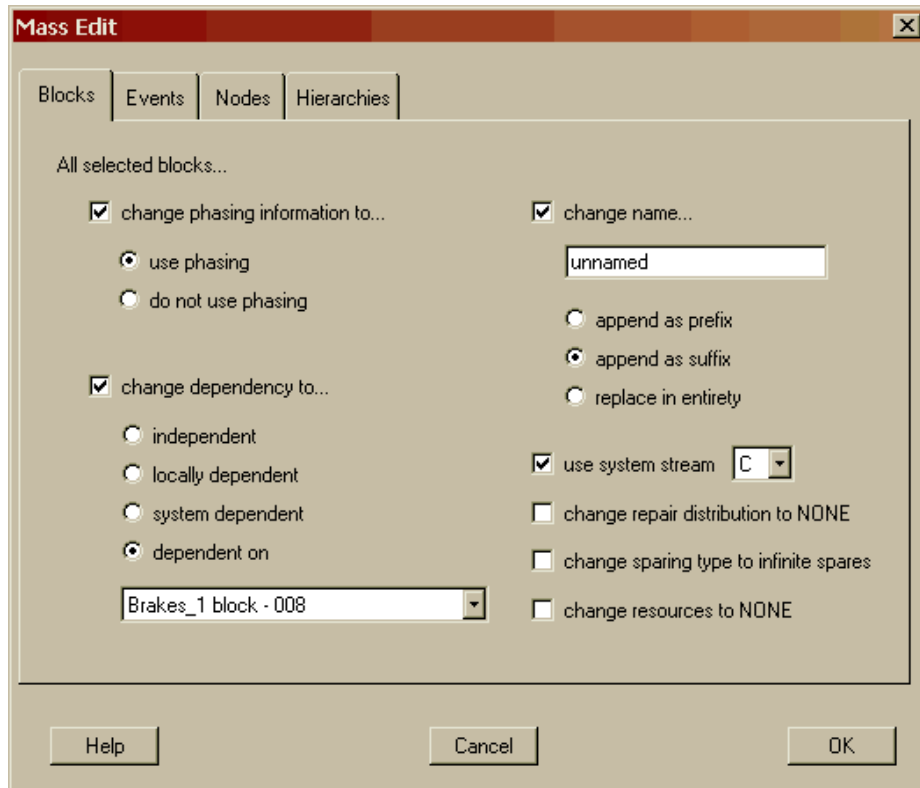


Figure 20.7 Mass Edit Blocks

If events are included in the group of selected elements, the event tab is visible (Figure 20.8). The phasing preference, success probability, system stream source, and event name can be changed for all selected events.

If nodes are included in the group of selected elements, the node tab is visible (Figure 20.9). Like the block and hierarchy elements, the phasing preference, dependency, and node name can be set from this tab. Additionally, the *k-value* for the *k-out-of-n* parallel structures can be bounded. A *k-value* of one represents an upper bound since the redundancy structures have the minimum requirement for number of needed paths. A *k-value* of *n* represents a lower bound since the redundancy structures have been converted into series structures.

If any hierarchies are included in the group of selected elements, the hierarchy tab is visible (Figure 20.10). Like the block and node elements, the phasing preference, dependency and hierarchy name can be set from this tab.

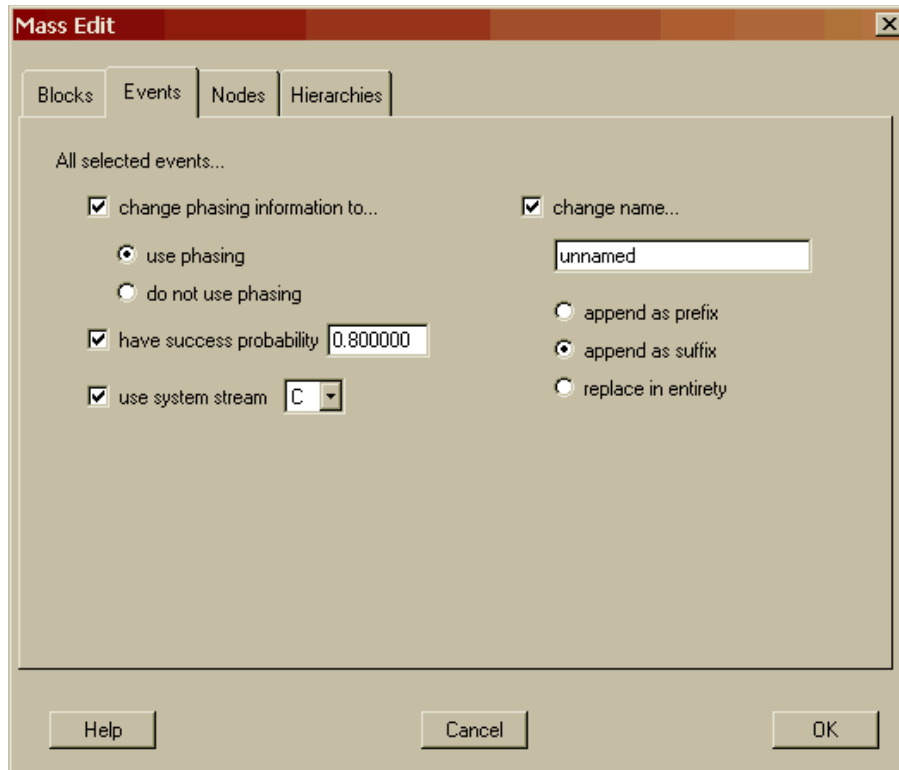


Figure 20.8 Mass Edit Events

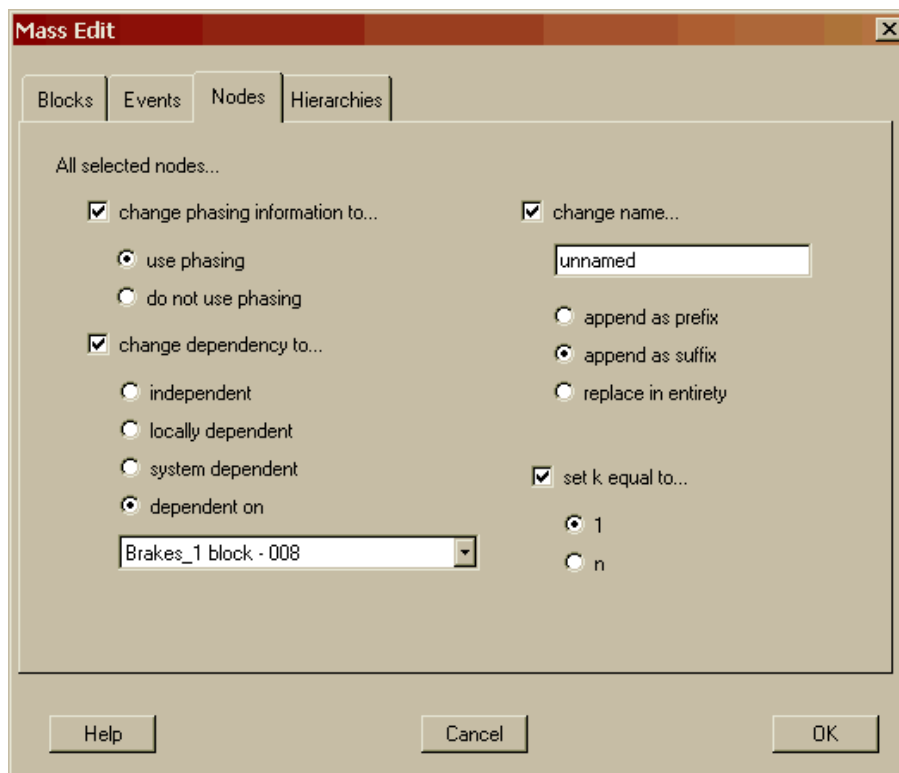


Figure 20.9 Mass Edit Nodes

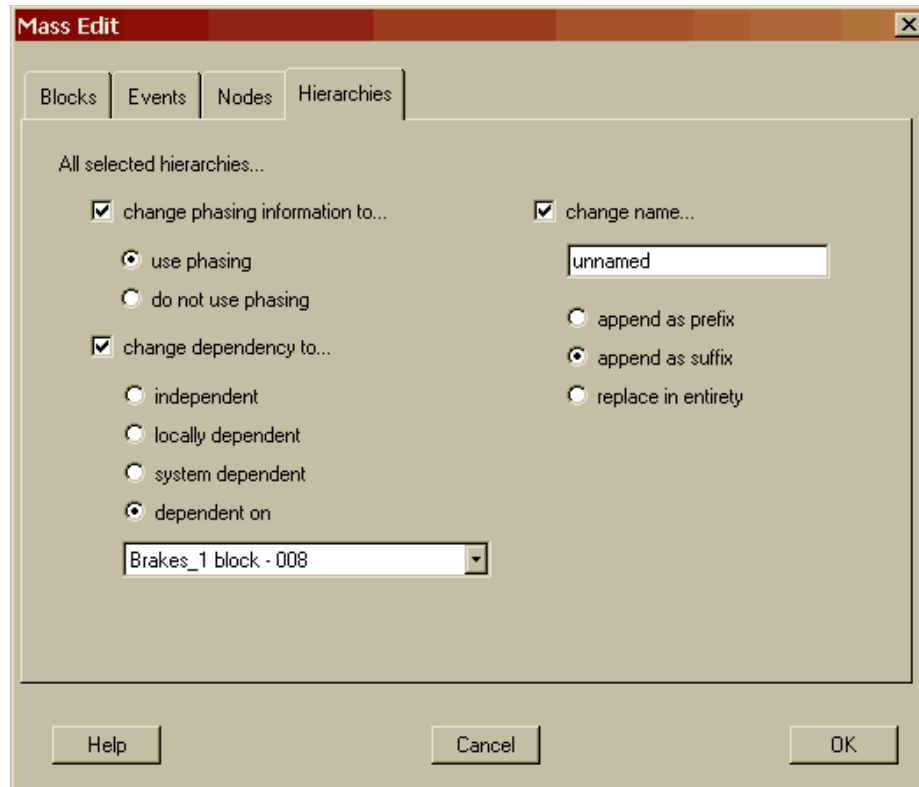


Figure 20.10 Mass Edit Hierarchies

A word of warning regarding the mass edit options is appropriate. The changes are permanent. Changing all repair distributions to None, for example, will delete all the previously entered selections for repair distributions and their relevant parameters. If the RBD is then saved, the original values are gone forever.

Library Functions

Raptor has the capability to read library files and thus allow users to select block templates from a library. The library must be in a specific format that is defined in the *Building a Block Library* section of the Raptor 7.0 help file. This document is available upon request from the Raptor Team. Users can integrate their own in-house tools by adding the ability to output data in the Raptor format. Raptor not only reads library files but it can create them by exporting an existing RBD to a library file. An update function is also available that can automatically update block templates within an RBD when a library is loaded. This feature is often employed by those users who receive monthly updates to their block failure and repair data. Library options are implemented by selecting the *File - Library - Import* or *File - Library - Export* menu options, as shown in Figure 20.11 on the following page.

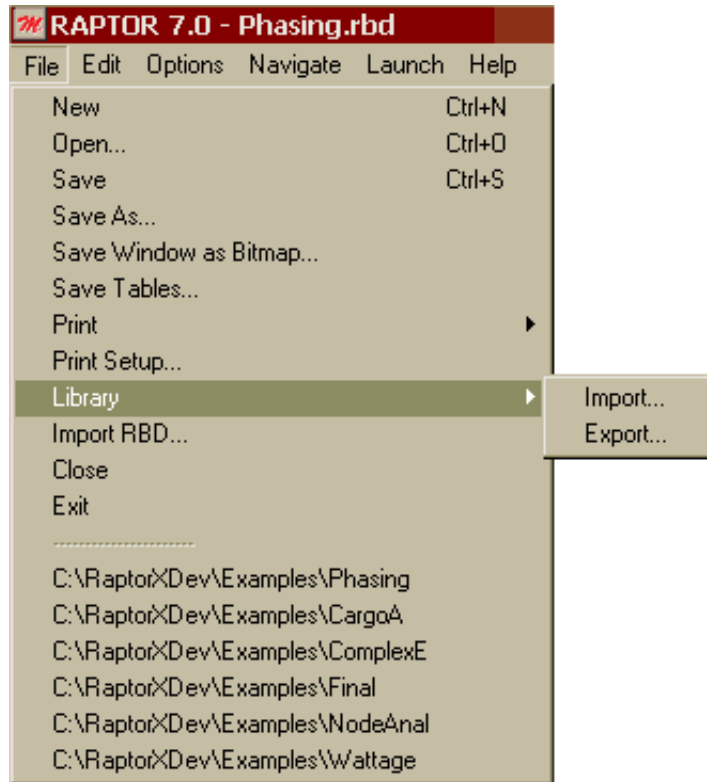


Figure 20.11 Library Options

Print Tables Dialog Box

The Print Tables dialog box (Figure 20.12) allows a user to print any of Raptor’s tables from one location and can be accessed from the *File – Print – Tables* menu option. The input table check boxes are for printing block, event, node, and hierarchy input information as well as input information related to system settings, spare pools, resources, and triggers.

Only tables relevant to the current RBD will be available for selection. For example, if an RBD does not contain any events, then the event table check box will be deactivated. The System Settings checkbox is always available. The phasing tables are only available for printing if at least one phase has been defined.

The output tables can only be printed if a simulation has been conducted. Any number of check boxes can be marked before selecting the OK button to implement printing selections.



Figure 20.12 Print Tables Dialog Box

Print Window Dialog Box

The Print Window dialog box (Figure 20.13) allows a user to print the current RBD window and can be accessed from the *File – Print – Window* menu option of the Workspace View menu bar, or the *File – Print Window* menu option from the menu bars associated with the other three views. Additionally, each view contains a Print Window button on its toolbar for quicker access to this feature. Users can select the portrait or landscape format for an RBD window printout and specify the number of pages to print the window across.

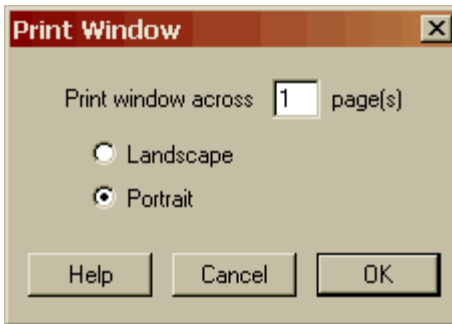


Figure 20.13 Print Window Dialog Box

Find Item Dialog Box

When large RBDs are constructed, it can be difficult to find a particular element. The find item feature provides a simple way to locate any Raptor element within the current RBD. Select the *Navigate - Find Item* menu item or simply click on the Find Item button from the toolbar and the Find Item dialog box will be displayed, as shown in Figure 20.14.

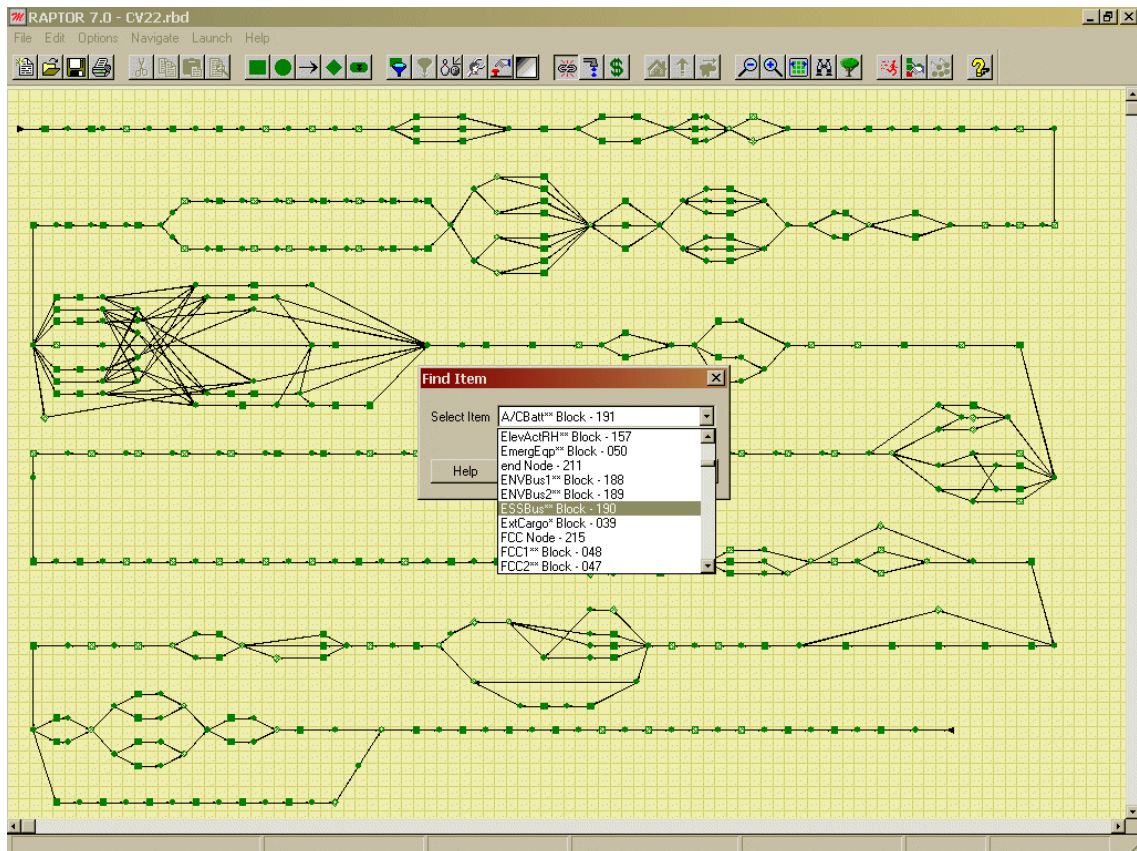


Figure 20.14 Find an item in a large RBD

The dialog box contains a list box with the names and ID numbers of all elements contained within the RBD. Users select an element by scrolling down and clicking on the name of the desired element. Raptor will then adjust the view of the RBD so that the chosen element is visible and highlighted. If the chosen element is not contained within the active window, Raptor will change into the window of that element's parent hierarchy so that it is visible on the screen.

Mean Life Growth and Decay

On the Advanced tab of the Block Properties dialog box (Figure 20.15), there is a check box labeled "This block will change its mean life following a failure". Checking this box allows the user to specify growth and decay options for a component. This instructs Raptor to modify a component's mean life during a simulation as it fails and repairs. The three types of growth or decay are known as linear, geometric and asymptotic.

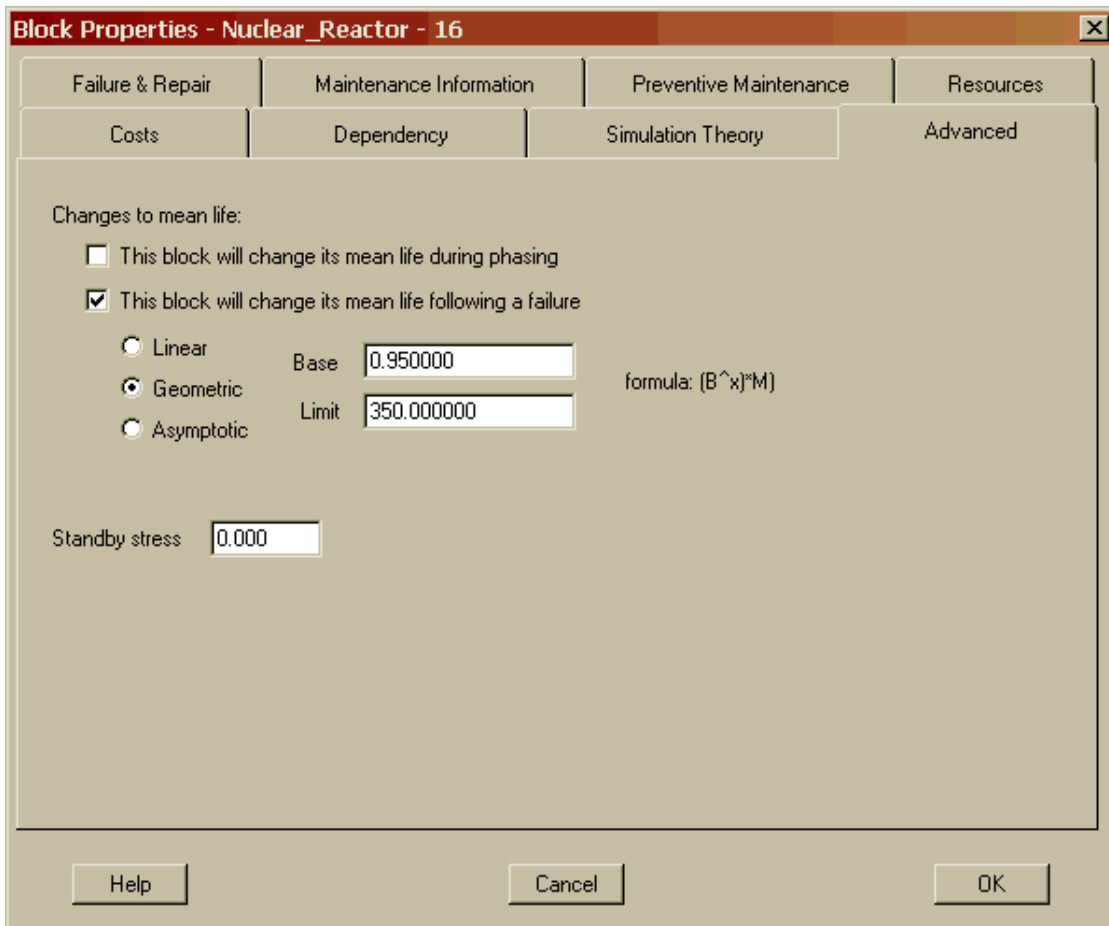


Figure 20.15 Advanced Tab of the Block Properties Dialog Box

Linear growth or decay: When this option is chosen, the relative mean life of a component will change after each failure in a linear manner according to the slope and

mean life limit entered. A limit can also be imposed on the growth or decay. The equation for calculating linear mean life is as follows:

$$\theta_{new} = [(Slope \times \# \text{ of failures}) + 1] \theta_{base}$$

If the slope is specified as zero, the mean life remains constant. A positive slope causes the mean life to grow over time and a negative slope causes the mean life to decay. For example, if the mean life of a component is 100 hours with a linear growth slope of 0.10 and a growth limit of 180 hours, the mean life curve that will be in effect during a simulation is shown in Figure 20.16.

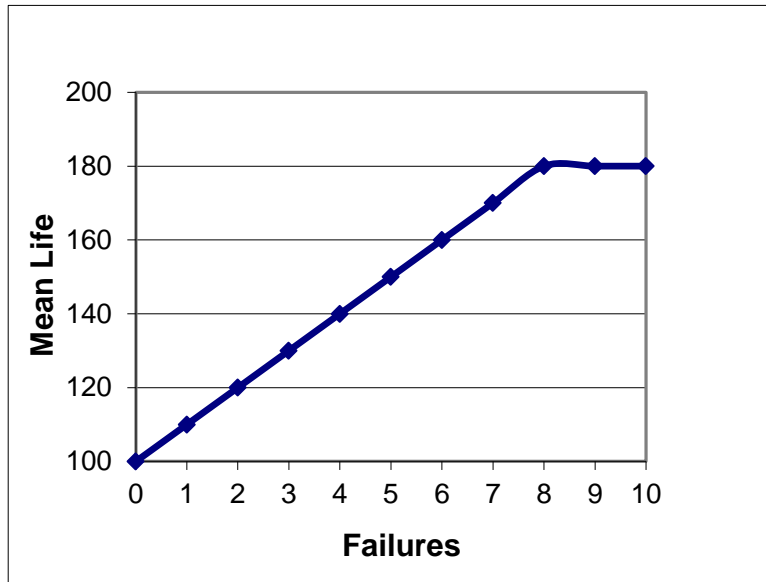


Figure 20.16 Linear Growth

Initially, when a component has not experienced any failures, it will have a mean life of 100 hours. Once a component has failed and repaired, its average time to the second failure will be 110 hours. Once the mean life has grown to 180 hours it will remain at that value.

Geometric growth or decay: When this option is chosen, the relative mean life of a component will change after each failure in a geometric manner according to the base value entered. A limit can also be imposed on the growth or decay. The equation for calculating geometric mean life is as follows:

$$\theta_{new} = (Base^{\# \text{ of failures}}) \theta_{base}$$

If the base value of one is specified, the mean life remains constant. A base value greater than one causes the mean life to grow and a value less than one cause the mean life to decay. For example, if the mean life of a component is 100 hours with a geometric decay base value of 0.90 and a limit of 40 hours, the mean life curve that will be in effect during a simulation is shown in Figure 20.17.

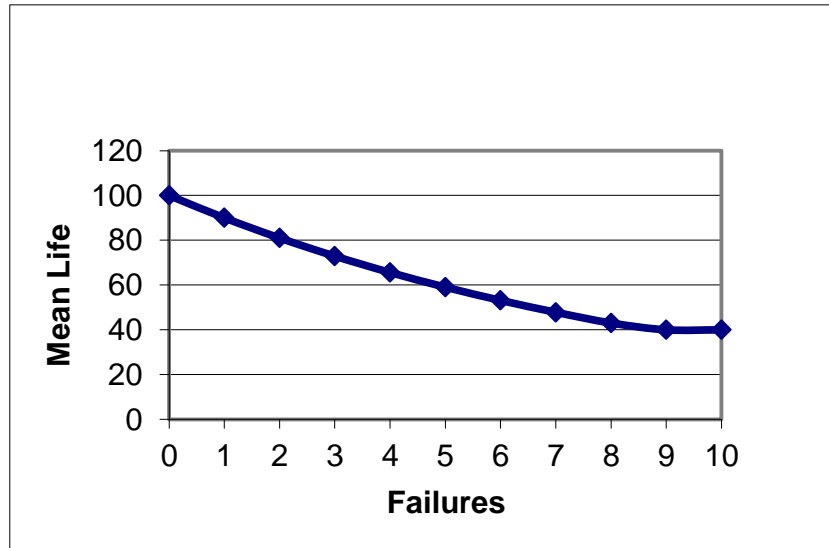


Figure 20.17 Geometric Decay

Initially, when a component has not experienced any failures, it will have a mean life of 100 hours. Once a component has failed and repaired, its average time to the second failure will be 90 hours. Once the mean life has decayed to 40 hours it will remain at that value.

Asymptotic growth or decay: When this option is chosen, the relative mean life of a component will change after each failure in an asymptotic manner according to the entered limit. The rate entered effects how quickly a mean life approaches the limit. The equation for calculating mean life is as follows:

$$\theta_{new} = Limit + (\theta_{Base} - Limit)e^{(-rate \times \# \text{ of failures})}$$

If the limit equals the original mean life, the mean life remains constant. If the limit is greater than the original mean life then growth occurs and if it is less, decay occurs. The rate entered must be greater than zero. High rate values cause the mean life to approach a limit faster. Assume a component has a mean life of 100 hours and it has asymptotic growth with a limit of 150 hours. The graph in Figure 20.18 shows the mean life that will be in effect during a simulation for various growth rates.

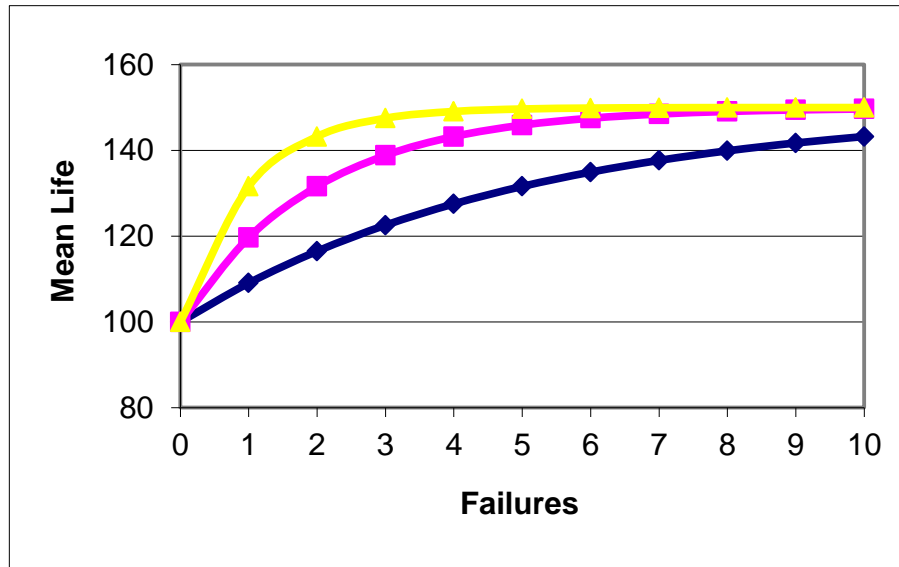


Figure 20.18 Asymptotic Growth

Initially, when a component has not experienced any failures, it will have a mean life of 100 hours. Once the mean life has grown to 150 hours it will remain at that value. Although this method has the greatest flexibility of the three growth or decay methods, it is somewhat difficult to determine the appropriate rate value. It is probably best to use an analysis tool that generates graphs and displays the expected growth curve along with an asymptotic growth curve for a given value of rate and limit. Vary the rate and limit values until an asymptotic curve matches the expected growth of the component.

Random Numbers

Raptor is a discrete event simulator that simulates the operation of a system through the use of random numbers. Regardless of the statistical distribution selected to model a component, the process of determining the failure or repair time of a component starts with a random number that is uniformly distributed between zero and one (denoted UR[0,1]). Various algorithms are used to convert this UR[0,1] random number into a value from a particular distribution.

Raptor defines two types of streams for use; independent streams and system streams. To avoid confusion, the independent streams are referred to using the numbers one through six and 11 through 69 and the system streams are referred to as streams A through J. By default, system streams A through D are set to seed numbers 7, 8, 9 and 10 while streams E through J are set to seeds 70 through 75. The assignment of seeds for the system streams can be changed by using the System Settings dialog box.

An additional feature of Raptor is the ability to use a stream's antithetic values. An antithetic stream is a set of random numbers equal to one minus the value in the original stream. For example, if the first random number in stream one is 0.888888,

then the first value in the antithetic of stream one is 0.111112. The option to choose the antithetic of each stream is available as a checkbox just to the right of each list box that contains the stream choices. When checked, the antithetic of the selected stream will be used.

The source of random numbers a component will use during a simulation is defined on the Simulation Theory tab of the Block Properties dialog box (Figure 20.19).

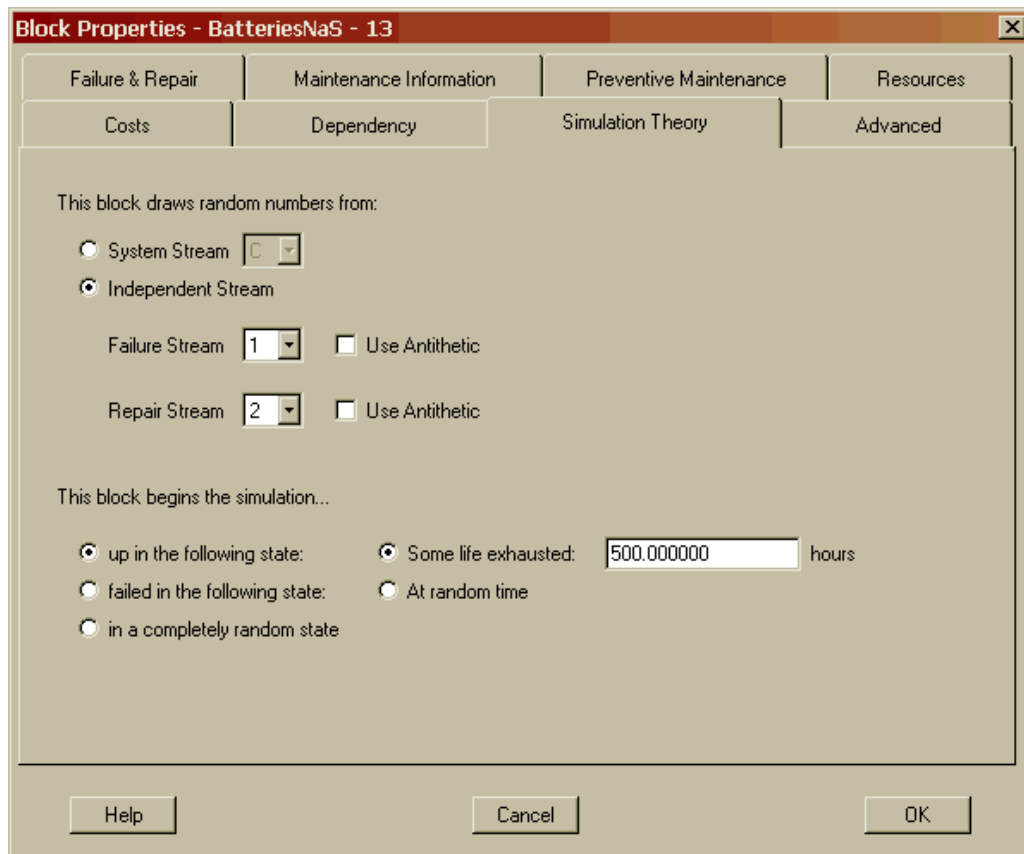


Figure 20.19 Independent Stream Selected

Block Defaults

When a block is placed within the Workspace view, all of its fields are loaded with default values. The initial Raptor settings for the component's default values are set so that it operates in a simplistic manner. It will have infinite spares, no logistics delays, no resource requirements and no dependencies. All cost values are set to one dollar and no advanced features are activated. The only parameters that must be defined are the failure and repair distributions. The default failure distribution is the exponential distribution, as this distribution is very commonly used in the reliability community. For the same reason, the lognormal distribution was chosen as the default repair distribution.

Users have the ability to modify the default values that the component possesses when it is placed within the Workspace. To set a component's default values, select the *Edit - Block Defaults* menu item. A modified Block Properties dialog box is displayed (Figure 20.20). Any changes made are saved with the RBD file and become the default setting for that RBD only.

The image shows a software dialog box titled "Block Properties - unnamed - 0". It has a tabbed interface with the following tabs: Costs, Dependency, Simulation Theory, Advanced, Failure & Repair, Maintenance Information, Preventive Maintenance, and Resources. The "Failure & Repair" tab is currently selected. The dialog contains the following fields and controls:

- Block Name:** A text input field containing "unnamed".
- Block Template:** A dropdown menu set to "Default".
- Comment:** A large text area for entering a comment.
- Failure Distribution:** A dropdown menu set to "Exponential".
- Repair Distribution:** A dropdown menu set to "Lognormal".
- Mean (Failure):** A text input field containing "100.000000" with the unit "hours" to its right.
- Mean (Repair):** A text input field containing "10.000000" with the unit "hours" to its right.
- Location:** A text input field containing "0.000000" with the unit "hours" to its right.
- Standard Dev:** A text input field containing "2.000000" with the unit "hours" to its right.
- Summary Statistics:** Two columns of text showing "Mean: 100.000000" and "Stdev: 100.000000" on the left, and "Mean: 10.000000" and "Stdev: 2.000000" on the right.
- Buttons:** "Update Statistics", "Help", "Cancel", "Reset Defaults", and "OK".

Figure 20.20 Setting Block Defaults using the Block Properties Dialog Box

The Reset Defaults button will reconfigure the block defaults to the Raptor factory settings. These are the defaults that existed when Raptor was first installed.

Simulation Options Dialog Box, Advanced Tab Options

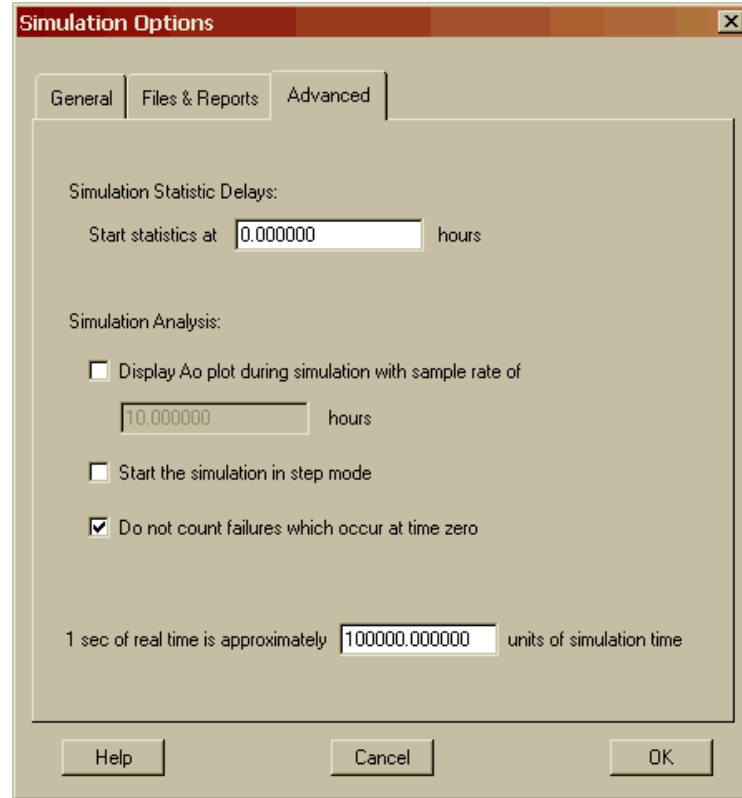


Figure 20.21 Advanced Tab of the Simulation Options Dialog Box

There is a check box on the Advanced tab of the Simulation Options dialog box (Figure 20.21) that has not yet been discussed in this workbook. This check box is labeled “Do not count failures which occur at time zero”. In most simulations, all components start in a good status, so a system starts up. When recovery simulation options are implemented, it is possible to start a simulation with components in a variety of conditions, resulting in the possibility of a system starting down. Since failure-truncated simulations are designed to run until a specified number of system failures occur, this option allows the user to specify whether the initial failure will count towards that total.

Problems

1. If you enter the mean of an exponential distribution as 200 hours for component A and then change the preferences so that the exponential parameter is entered as a failure rate, what value for the distribution parameter is displayed for component A in the Failure & Repair Distributions tab of the Block Input Tables dialog box?

2. If you select “Print RBD across 3 pages”, how many pages down are printed?
3. If a component has a base mean life of 100 hours and has linear growth with a slope of one, what is the mean life after two failures?
4. What is the mean life of the component in Problem 3 after two failures if the component is also phased and currently has a stress level of 0.5?
5. What happens when identical components are set to the same independent steam?
6. What is the difference in the way preferences are saved versus the way system settings and block defaults are saved?
7. Why is it generally not a good idea to count failures that occur at time zero?

Chapter 1: Elements and Views

1. There are seven rules to linking elements within Raptor. They are:
 - links cannot connect into the start marker
 - only one link can connect out of the start marker
 - links cannot connect out of the end marker
 - only one link can connect into the end marker
 - only one link can connect into or out of a block, event, or hierarchy
 - any number of links can connect into or out of a node
 - all elements must be properly linked together
2. First, blocks function (or operate) as simulation time progresses while events function instantaneously. Second, blocks are repaired while events are not.
3. While in the Workspace view, the status bar information is concerned with attributes of the elements. While in the Simulation view, the status bar information focuses on a simulation and the cumulative results of the trials.
4. Only the Workspace view allows changes to the attributes and structure of an RBD. Failure Effects view allows the user to determine the impact of failures on the system, while the other two views, Simulation view and Weak Link Analysis view , allow the results to be viewed .

Chapter 2: A Basic Reliability Block Diagram

1. A failure and repair distribution must be specified for each block.
2. The user controls the consistency of the units.
3. A right mouse click while in any add cursor mode will stop the add function.
4. No spaces are allowed in naming any element or file.
5. An element's name cannot exceed 20 characters.
6. Just before the start of any simulation, all unspecified k -out-of- n nodes would cause a dialog box to appear prompting the user for each node's k -information.

7. For a time-truncated simulation the number of system failures is a random variable, while for a failure-truncated simulation, the ending simulation time is a random variable.

8. A cycle-termination is most appropriate when an RBD has random-length phases defined, and the period of interest is in conducting all of the phases some number of times.

9. Each trial draws different random numbers. This feature ensures that the trials are independent and identically distributed (iid). Most basic statistical techniques have an inherent assumption of iid sample data.

10. The Minimum is the smallest MDT value produced from the ten trials. The Mean is the average MDT value across the ten trials. The Maximum is the largest MDT value produced from the ten trials. The Standard Dev is the *sample* standard deviation of the MDT for the ten trials. The SEM is the standard error of the mean.

11. Often, more precision is required for availability type parameters since systems are built that require resolution in the sixth or seventh decimal place. It is an extremely rare case that would require that kind of precision in the mean down time or mean time between downing events (especially given the typical precision of the input data).

12. Results from a particular simulation remain within Raptor until another simulation is conducted or an rbd file is opened or closed.

13. The exact reliability of the system is 0.838807966. A Raptor simulation of 1000 trials gives an approximate system reliability of 0.836. The true reliability of the system for any time t is:

$$R_{\text{system}}(t) = \left(e^{-2\lambda_a t} \right) \left(3e^{-2\lambda_b t} - 2e^{-3\lambda_b t} \right) \left(e^{-2\lambda_c t} \right)$$

14. An approximate solution by simulation gives $R_{\text{system}}(75) \approx 0.0775$. The exact solution is 0.072586167.

15. By conducting a simulation in Raptor of 1000 trials, we obtain an approximate reliability of 0.796024799 at time 75. An exact solution for steady-state availability could be calculated, but it is not possible to compute an exact solution for the reliability at time 75.

Chapter 3: Distributions

1. No. Only the parameters of a distribution can be modified within the Failure & Repair Distributions table; not the distribution type.

2. Systems that use the None repair distribution are usually those where the repair is economically unwarranted (i.e., too expensive to repair or a system itself is inexpensive so replacement without repair is justified). Examples include system models of satellites, or system models that include satellites as components; system models of smoke detectors or VCRs. Also, the None repair distribution may be used for simulations that address particular questions, such as a system model of an aircraft when the parameter in question is reliability over one mission, or a system model of a washing machine when the question to be addressed is how long should the warranty be.

3. If a component uses the None repair distribution, upon failure it is done: the block will not be available to the system, and will turn Indian Red if simulating with graphics. If the Fixed 0.0 repair is selected, upon a failure the component will immediately repair.

4. Six. The Chi-squared, Exponential, Laplace, Lognormal, Normal, and the Poisson distributions.

5. The empirical distribution does not contain information beyond the minimum and maximum data values entered. Thus, the empirical distribution usually possesses no information about the extreme regions of the data. These extreme regions are known as the tails of a distribution. Without the tail information, the extreme conditions are never simulated. The effects of long or short failures or repairs never occur and a simulation may not provide sufficient information concerning a system during extreme situations. These extreme situations often provide significant insight into a system's design.

6. Since three of the ten trials did not experience a downing event (i.e., reliability is 0.3), the MDT parameter can only be based on seven trials. If additional statistical analysis is performed on the data provided by Raptor, it is often critical to know the proper sample size of the data (i.e., seven vs. ten in this case).

7. Three trials of the ten did not contain any system downing events and thus the mathematically best nonparametric estimate for the MTBDE parameter is infinity. That is, averaging the seven good estimates (trials in which downing events occurred) with three estimates of infinity leads to an estimate for MTBDE that is also infinite. Raptor does not average, however, infinity into its estimate; instead, Raptor will assume the simulation termination time is the best estimate of the MTBDE for a trial without downing events. Since this is not the most mathematically correct estimate for the MTBDE parameter, Raptor warns users of this situation by placing a greater than sign in front of the MTBDE value.

8. Since three trials of the ten did not contain any system downing events, a good estimate of the MTBDE's standard deviation cannot be adequately determined. (See the answer to the previous question for a more detailed explanation.)

9. The data is discrete and bell shaped so the binominal distribution is a good guess but a better guess is the Poisson distribution. An interesting attribute of the Poisson distribution is that its mean is equal to its standard deviation; which is the case described in this problem.

10. Two clues indicate the normal distribution. First, the data is a collection of measurements, which therefore has a great probability of being normally distributed. Second, the shape of the data is generally bell-shaped. The dip near the mean is more likely caused by the small sample size and less likely an indication of bi-modal behavior.

11. Given the relatively large sample size, the dip near the mean is unlikely to be caused by a lack of data. Thus, the best distribution is likely the empirical distribution since this would mimic the bi-modal behavior.

12. $A_0 = 0.976734135$ and the MDT = 13.019976.

Chapter 4: Logistics Constraints

1. Use the infinite sparing replacement strategy when it is desired to determine a system's best possible RAM characteristics without regard to the effects of sparing or resources.

2. The block is held waiting for a spare to arrive into the pool. The block does not continue its repair cycle nor does it request resources. Once a spare arrives, the block continues with its repair cycle by requesting resources to initiate its hands-on repair if its pre-repair logistic delay time has expired.

3. Although the spare comes from a common pool, the repair characteristics emulate the repair distribution of the broken block. Thus, it is customary to place "identical" blocks into the same pool.

4. No. Ten spares arriving every 1000 hours stipulates that a batch of ten spares arrive after a thousand hours of simulation time has expired. Contrast this to a single spare arriving after 100 hours of simulation. In the first case, more delays due to the lack of spares are likely.

5. The priority used to assign resources to competing blocks is "First to ask, first to acquire".

6. Yes. A block assigned to a resource pool can be assigned to one of the three types of sparing strategies (i.e., infinite, spare pool, or custom).

7. The repair of Block B cannot begin because the resource is being held by Block A. The priority scheme of "First to ask, first to acquire" means that resources are held by

blocks even if these blocks do not have all the resources they need to start their repair. Thus, the action of holding resources can cause delays for other blocks.

8. The failure of a mechanical component in any facility that requires paperwork to be completed at the end of a maintenance action would best be modeled with a fixed length logistical time delay. Buoys out at sea are an example of a system that could require random length logistic time delays.

9. A 95% confidence interval for the number of system failures is given by (0.312746, 1.887253). This is computed by multiplying Student's t distribution, with $\alpha = 0.025$ ((1-0.95)/2, dividing by 2 makes it two-sided) and 9 (the number of trials minus one) degrees of freedom, by the standard error of the mean.

10. A 99% confidence interval on mean green time: (35.64, 65.86).

11. This question is the same as that of the confidence intervals, only in reverse. We are asked, essentially, what is the probability of being a certain number of SEMs away from the mean. To require more than one spare, we would be $(1-0.2)/(0.421637/\sqrt{10})$ SEMs away from the mean. Finding the inverse of the t distribution for this value gives us a probability of 0.00010125.

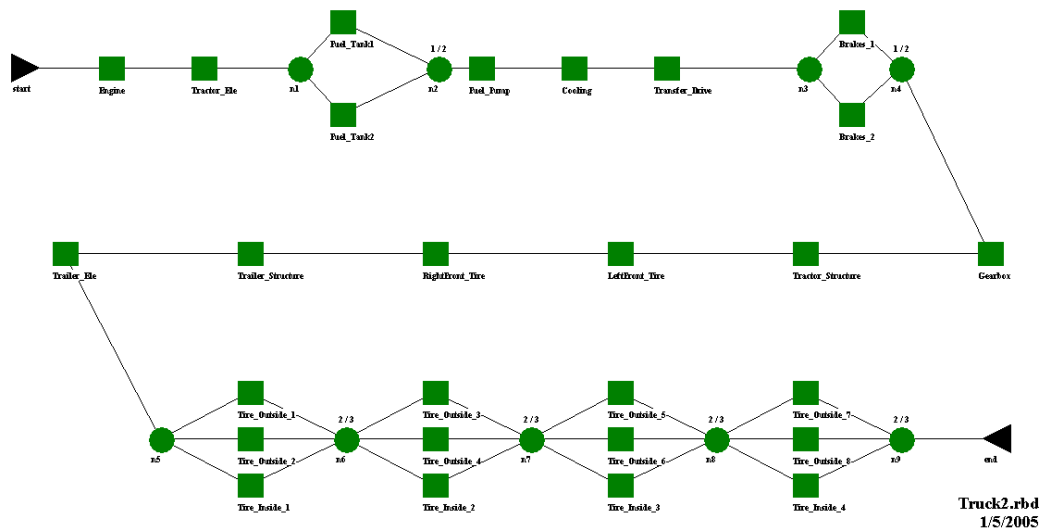
12. The length of any non-trivial repair cycle is random. The sum of any number of fixed repair or random repair lengths is always itself a random length.

Chapter 5: Hands-On Example

A. A system level downing event can be caused by any of the following:

1. Failure of any series component.
2. Failure of both fuel tanks.
3. Failure of both brake systems.
4. Failure of two tires on the same corner after spares for that type of tire have been depleted.

B. The key features of this RBD are the 1-out-of-2 structures for the fuel tanks and brakes and the 2-out-of-3 structures for the four rear corners. These six structures must be in series with the remaining blocks. The order in which they are placed in series does not matter.



C. A spare pool must be created that has an initial stock of one outside-tire. All eight outside tires must be set so they draw spares from this pool. A second spare pool must be created that has an initial stock of one inside-tire. All four inside tires must be set so they draw spares from this pool. All other blocks can be left with the default of infinite sparing since the remaining blocks are designated to have no repair capability (i.e., a sparing strategy is not relevant).

D. A mission reliability for 1200 miles is 0.84 and the MTBDE is 6375 miles. See Figures 5.3 and 5.5, respectively. Discussion is included in Chapter 5.

1. Only 5 of the 25 trials experienced a downing event, so the MDT parameter was calculated by averaging the MDT obtained on those five trials.
2. In this case, they represent the amount of miles left to the end of simulation after the downing events occurred.
3. Availability is a function of a system's MTBDE and MDT. The MDT is only based on five data points and the MTBDE is displaying a greater than sign, which indicates that it is not good estimator. Since both the MTBDE and MDT are bad or weak estimators, any parameter that incorporates them into a function is also a poor estimator.

$$A_o = \frac{MTBDE}{MTBDE + MDT}$$

4. A mission reliability simulation is time-truncated and is used to determine a system's probability of surviving some specified time. An endurance simulation, however, is failure-truncated and is used to determine the average length of time before a user specified number of system level failures.

5. In order to obtain a meaningful system downtime in hours for the truck, failure data for the truck's components would have to be entered in hours. This may be a simple transformation relating miles traveled to hours traveled. The repair time for each component would also have to be entered for each component. Then a simulation could be run for an extended time to determine a steady state estimate of the truck's MDT.

6. First, the fuel tanks are not clearly defined to be a 1-out-of-2 parallel subsystem. They could be two tanks in series. Second, trailer tires are defined in Figure 4.1 to be any tires under the trailer even though two of the four axles are actually part of the tractor. These ambiguities are there to demonstrate that most modeling efforts take several iterations to converge on the best modeling approach.

7. The only components in the truck system that are repairable are the tires. Since the MMT parameter is reported as being based on eight runs, this means that on eight of the 25 trials, there was a tire failure that incurred maintenance. Thus, on the other 17 trials, a critical failure occurred before any tire failures.

Chapter 6: Preventive Maintenance

1. To have one or any components perform PM at a random frequency, the user must tie the components' PM frequency to a trigger.

2. With the length of the simulation being 730.5 hours, and the Attitude's preventive maintenance set to occur after 1500 hours, the Attitude component will never perform PM during the simulation, so one may argue that this does not make sense as it is irrelevant. If the simulation were due to run longer than 1500 hours, then this would be a situation where this set up would make more sense.

3. Assuming that the component does not experience a hard failure, the first PM activity will occur at time 50. This maintenance will last ten hours during which time it does not accumulate hours. Hence, the next PM activity will occur at time 160.

4. This is a near identical set up to that of problem 3, however now the component's PM is tied to a trigger's schedule. Hence the first PM activity will be at time 50 and the second will occur at time 150.

5. Despite the fact that the elapsed time field can only be a fixed value, it is possible for a component's first preventive maintenance activity to occur at a random time. One way for this to occur is if the component's preventive maintenance schedule is tied to a trigger, and the trigger fires at random intervals.

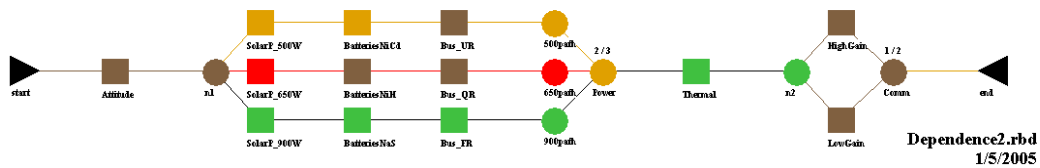
6. Consider the motorcycle as a system. If one conducts replacement preventive maintenance on the tires, this would require a spare. If one conducts a cleaning PM on the chain, this would not require a spare.

7. Changing the oil in a vehicle is a preventive maintenance action that is necessary but would not refresh the engine as new. Changing the tire on that same vehicle would refresh it as a new component.
8. If the defer preventive maintenance until spare and resources are available option is not selected there is a risk of component unavailability and subsequent system impacts while it enters a hold time waiting for spares and/or resources.
9. Changing a tire would reset the time to the next preventive maintenance. A hard failure of a structural component that is cleaned for PM along with other components will not reset the PM time.
10. This particular condition may be attributable to a component that is taken out of service to be inspected or cleaned. The PM action itself does not extend the mean life of the component but it does affect the system in some way and utilizing a PM action allows for this condition to be modeled.

Chapter 7: Dependence

1. Automobiles. Tires, wheels, the engine and transmission of an automobile are examples of components that require the others to be operational in order to function properly.
2. Automobiles. Opposite the previous example, if the radio or air conditioning of an automobile fails most likely it can still function and many drivers continue driving.
3. No. Only the block that is directly tied to a failed block will go to an idle status. The other blocks will continue to operate.
4. Yes, this is common. For example, an adverse weather event might lead to the failure of a component.
5. No. An event cannot be dependent on any other element.
6. Local dependence can be set at the time a block is created even if its downstream node is not yet defined (the node does not exist or the block is not yet linked).
7. In order to insure a third string goes to an idle status, the effects of the failed Power node must be passed back to this string. The blocks are already dependent on the node at the end of the string. Select the three nodes at the end of each string, that is, the 500path, 650path and 900path node and set the dependency for these nodes to local dependence.
8. Change the Power node so that it is system dependent.

9. Starting from left to right: The Attitude is brown because it has gone idle because the system is failed. Node n1 is brown because it reflects the state of the subsystem preceding it and the Attitude is idle. In the top string, the first two components are orange because their triggered preventive maintenance has come up. The Bus_UR is brown because its compatriots are down for PM. The 500path node is orange because the reason this string is not operational is PM. The SolarP_650W is red because it has failed. This failure has caused the other two components in the string to go idle, hence their color is brown. The 650path node is red since it is a failure which is causing it to not be operational. The Power node is orange since it is the preventive maintenance in the top string which is causing it to not be available. The HighGain and LowGain components are system dependent, so the failure of the system moves them to an idle state, hence the brown color, which is then also reflected in the Comm node.

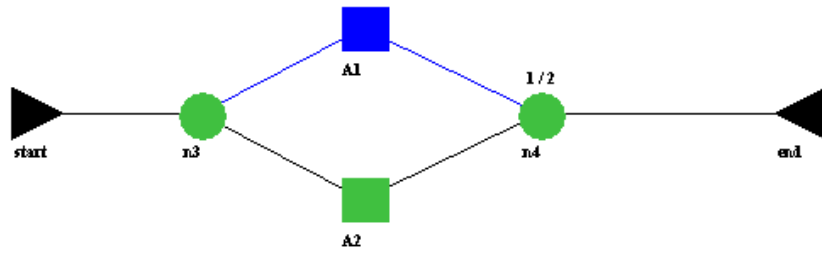


10. In a typical system, where the dependent components go to an idle state upon a failure, the true availability and dependability of the system are necessarily better than the results obtained. If induced failures are a possibility, the true values could be better or worse; we cannot determine which offhand.

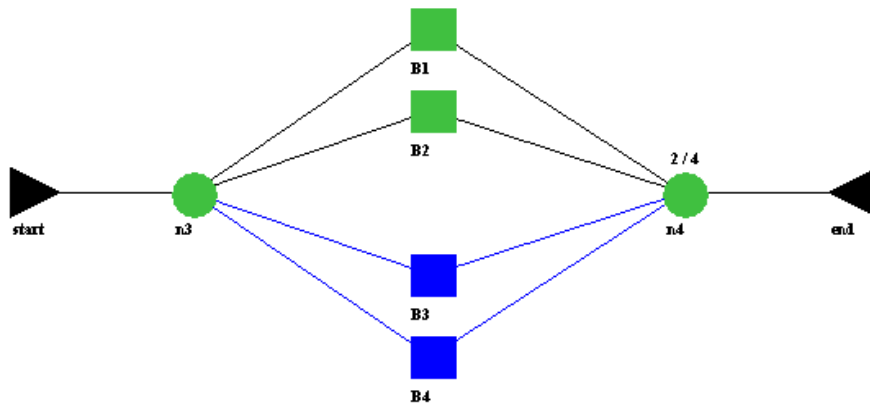
11. If all components in the electric motor are independent, we can mathematically compute the steady-state availability to determine that it is 0.578087791. When dependence is considered, we need a Raptor model. Conducting 5 runs to sim time 20,000 gives an availability of 0.613448667.

Chapter 8: Standby

1. Petroleum pumps and power generators.
2. Priority return is usually associated with non-identical standby components. An example might be alternate power sources such as commercial power, batteries and generators. Typically, when commercial power is returned after an outage, generators or batteries are turned off and power is switched back to the primary power source.
3. This has no affect on the system. When a standby node attempts to change an inbound path to standby, the node will check whether the first upstream element is a block or a node. If it is a block, that element is switched to its standby status. If it is a node, however, the upstream node is instructed to propagate the standby status to any elements that are dependent on the upstream node. If the upstream node does not have an element dependent upon it, the path will not change to standby.
4. System A has an availability of approximately 80% and System B has an availability of approximately 75%.



System A



System B

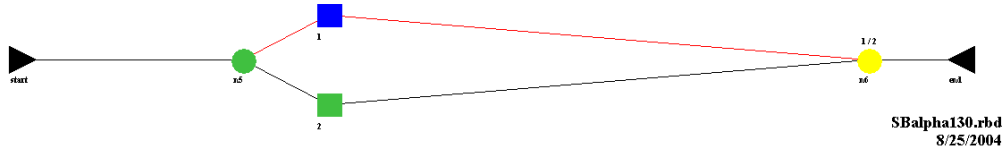
5. When system C conducts a switch-over, system downtime does not occur and thus, there are no downing events due to switch-over. Each time system D fails, however, the system is experiencing a small amount of system downtime. MTBDE is defined to be the system uptime (green time + yellow time) divided by the number of downing events. The uptime is essentially the same for these two systems but the number of downing events will differ greatly. The MTBDE for system D will be significantly lower than the MTBDE for system C. Availability is calculated as the uptime divided by the total simulation time. Since the uptime is nearly identical for both cases, the availability of the two systems will be very similar.

6. No. In order for a node to be designated a standby node it must have at least two inbound links.

7. Oil pumps. In the petrochemical industry, when oil pumps are not being used, they are more susceptible to rust and more likely to fail.

8. Events cannot go to standby. However, it is perfectly reasonable to place an event in a standby configuration with blocks, as long as the priority of the event's path is higher (lower number) than the blocks' paths.

9. The node is set to standby. There is automatic switching with probability < 1, and no manual switching. The top block tried to switch at some point and failed, hence it is now (and forever) unavailable.



Chapter 9: Transients

1. 250 data points

$$\# \text{data points} = \frac{\text{SimTime}}{\text{SampleRate}} = \frac{500}{2} = 250$$

2. Graphs are limited to 1,000 data points and the combination of the simulation time and sample rate exceeds this limit.

$$\# \text{data points} = \frac{\text{SimTime}}{\text{SampleRate}} = \frac{5000}{2} = 2500 \not\leq 1000$$

3. Since graphs consume a significant amount of memory, Raptor was designed to require the user to request graphs each time they are desired.

4. The multi-run availability graph. The other two graphs (per trial and interval graphs) display availability for a particular trial.

5. The uppermost line is the maximum availability observed at a particular time. The lowermost line is the minimum availability observed at a particular time. Neither of these lines represents average values but extreme values with respect to time. The middle line is the average availability observed at a particular time.

6. If at some point the three lines of the Multi-run A_0 Graph converge, nothing can be concluded. However, if they converge while being parallel to each other, we can conclude that the system has achieved steady-state with respect to the availability parameter.

7. The block continues to repair but only the segment of the repair cycle that crosses into the statistics-gathering portion of a simulation is accounted for within the results.

8. A satellite, leased equipment, refrigerator, etc. Equipment that will never reach its true steady-state RAM value because it will be discarded long before this point in time occurs.

9. Yes. The Simulation Options dialog box changes to allow the user to enter the number of failures after which data is collected. It is equally possible to use the delayed statistics option with cycle-truncated simulations.

10. The block's total repair cycle is ten hours (i.e., the sum of 2, 5, and 3). Thirty-five percent of the repair cycle is 3.5 hours. Therefore, the block is in the hands-on repair portion of its repair cycle. The block has completed 1.5 hours of hands-on repair and has 6.5 hours (i.e., the sum of 3.5 and 3) remaining in its repair cycle.

11. The block's total repair cycle is ten hours (i.e., the sum of 2, 5, and 3). Eighty-five percent of the repair cycle is 8.5 hours. Therefore, the block is in its post logistic delay portion of its repair cycle. The block has 1.5 hours of logistic delay remaining in its repair cycle.

12. When the simulation begins, Raptor steps through each block in the order it was placed within the Workspace and initiates that block for the simulation. Once spares or resources are exhausted, other blocks requiring them will be put in a hold status waiting for a new spare or resources to arrive.

13. No. The exponential distribution is unique in that a component that has some life accumulated is virtually the same as one that is new. This is called the memoryless property of the exponential distribution. It can be shown mathematically that the conditional probability density function of an exponential distribution is equivalent to its probability density function.

14. It is possible for the two blocks to both start down when random startup techniques are utilized. Once they have repaired, however, this condition cannot occur again within a trial. Since both blocks are system dependent, if one block fails the other is sent to its idle status and cannot fail before the first block is repaired. Thus, it is not a natural state of this RBD for both series blocks to be down simultaneously.

15. Most modern components have a much larger failure mean than a repair mean. In other words, the availability of each component is relatively high so blocks spend more time up than down. A random start will reflect this since it is common for an RBD to start with many of its blocks in an up state, even if all were set to start in a random condition.

Chapter 10: Failure Effects View

1. If a block is hatched in Failure Effects view, then its current state is due to a dependence action.

2. The user may cycle through whichever states possible based on the dependence actions that were defined for that block when clicking on it when hatched in FEV.

3. When the top block is failed, in FEV mode this will cause the following block to fail, which will then cause the third block to fail, and so on. The result is that all ten blocks will fail, hence the system will fail, and we can say that the top block is a single point failure block. If the fifth block is failed, this will cause a chain reaction of failures as we saw before, with the result that the fifth block on will all be failed. Since the system only requires one block to be up, and the top four blocks have not failed, the fifth block is not an SPF block.

Chapter 11: Weak Link Analysis

1. Possibly but not likely because many blocks are not independent of each other logistically. Blocks that share resources or spares are correlated and multiplication of their node availabilities is meaningless in a probabilistic sense.

2. Node A represents the probability that block alpha is operating at any given time (i.e., the availability of block alpha).

Node B represents the probability that either block alpha or block beta or both are operating at any given time. $P[\alpha \text{ or } \beta]$

Node C represents the probability that block beta is operating at any given time.

Node D represents the probability that both block alpha and block gamma are operating at any given time. $P[\alpha \text{ and } \gamma]$

Node E represents the probability that either block beta or block alpha and block gamma or all three are operating at any given time. $P[\beta \text{ or } (\alpha \text{ and } \gamma)]$

Node F represents the probability that the system is operating at any given time. $P[(\alpha \text{ and } \gamma \text{ and } \delta) \text{ or } (\beta \text{ and } \epsilon) \text{ or } (\alpha \text{ and } \gamma \text{ and } \epsilon)]$

3. Logistics, Mean life, and Redundancy.

4. Blocks that are dependent are not available when the element they are dependent upon is down. It is not clear by looking at availability numbers only which blocks actually failed and which ones were down for dependency reasons. Consider using dependability weak link analysis for systems that possess a fair amount of internal dependencies.

6. The weak link analysis results change for both the blocks and the nodes. The only reason they change is due to the dependency settings.

Chapter 12: Cost Analysis

1. \$2500 (5 x \$500)

2. There is no such field. Raptor has costs associated with a component being in a hold status due to logistics constraints or preventive maintenance, but there is no cost associated with having a spare in stock.

3. \$2281

7 resources	\$840	(7 x 8hr x \$15)
2 hrs LDT	\$16	(2hr x \$8)
8 hrs repair	\$175	(8hr x \$20 + \$15)
<u>spare</u>	<u>\$1,250</u>	
Total	\$2,281	

4. At the top of both the End of Run and the Final Results report is a section that lists how many spares each block used. Values in this table can be multiplied by the cost of each spare to determine how much of a pool's total cost was due to each block.

Chapter 13: Hands-on Example II

1. Ambiguous in the system description of the pie-making facility include whether or not there were any pie-line preferences that could be used to determine stand-by priorities.

2. To determine how many pies of each type were produced, we would have to conduct the simulation with the final results report selected. Then we could look at the weak link analysis section of this report, node analysis state data subsection. This will give the percent of simulation time that the three pie nodes were in each state. At the times that they were green, they would be producing pies.

3. One remaining assumption was that the 1 of 3 pies node was always operating with 2 of the lines running, this was likely not correct leading to slightly incorrect capacity calculations.

4. The apple_pie node had an availability of 0.911361420, for the cherry_pie node it was 0.917006846, and for the rhubarb_pie node it was 0.912852608. These results are quite similar, yet this should not surprise any Raptor users. The components leading in to these nodes have identical MTBDEs and MDTs across the three nodes. Hence, one would expect the availabilities of the nodes to be equal once the system has reached steady-state; if we conducted the simulation for more time or more trials, we should expect the availabilities of the three nodes to be even closer than they are.

Chapter 14: Capacity Analysis

1. Throughput is a measure of what is flowing through a path or a system. Capacity is a measure of what a path or system is capable of flowing.

2. In order to obtain the cost of lost flow, both the capacity and cost analysis features must be activated.

3. Units produced are the product of the flow rate and time.

$$(10 \text{ units per hour}) \times (2 \text{ hours}) = 20 \text{ units produced}$$

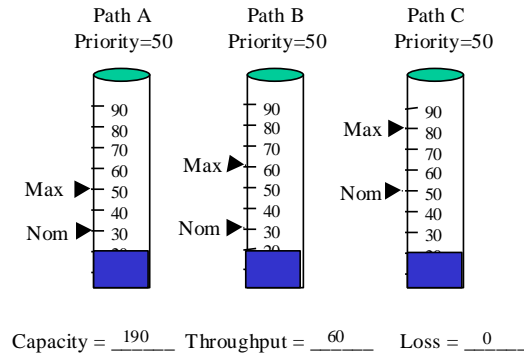
$$+ (40 \text{ units per hour}) \times (1 \text{ hours}) = 40 \text{ units produced}$$

= 60 units produced

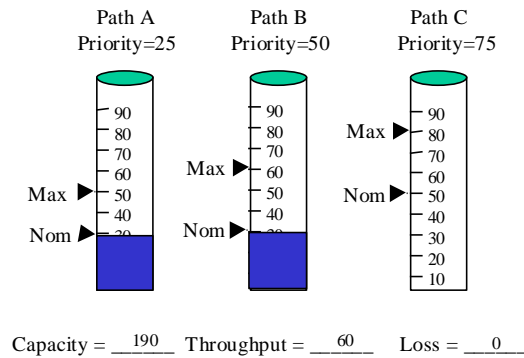
The average flow rate can be calculated using the units produced result.

$$\text{Average Flow} = \frac{\text{Total Units Produced}}{\text{Simulation Time}} = \frac{60 \text{ units}}{3 \text{ hours}} = 20 \text{ units/hour}$$

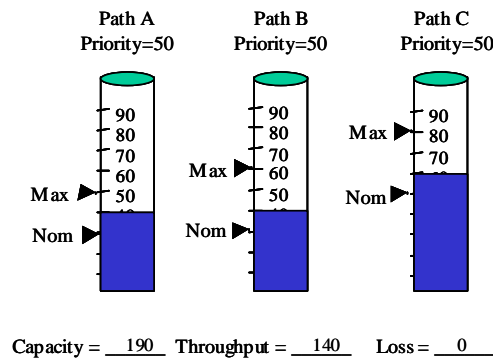
4. The 60 units of flow are evenly distributed.



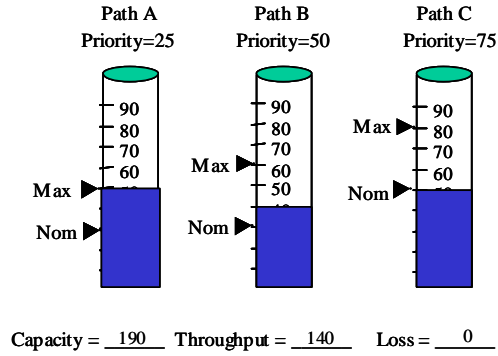
5. Paths A and B are each filled to their nominal values.



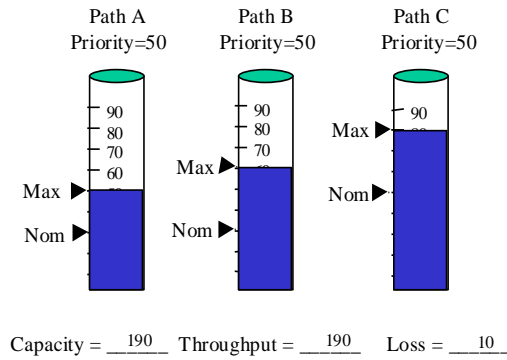
6. Paths A, B and C are each filled to their nominal values first; then the additional 30 is evenly distributed.



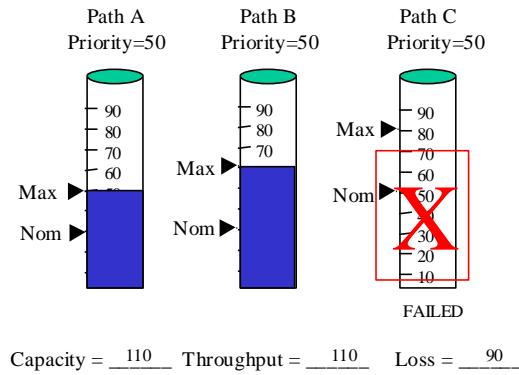
7. Paths A, B and C are each filled to their nominal values; then Path A is filled to its maximum flow value and then the remainder is sent to path B.



8. Paths A, B and C are all filled to their maximum capacity. The remaining 10 units of flow are lost.



9. Flow cannot be sent to Path C. Paths A and B are filled to their maximum capacity and the remaining 90 units of flow are lost.



10. The average flow rate is 204.904. This is an exact answer and with sufficiently long simulation lengths, a simulation solution will approach the exact solution.

11. There are multiple solutions. One is to add another 900 watt power string, which results in an availability of 0.971821224, a mean flow of 1,451.79 watts, at a cost of \$33,872,015.09.

12. A node can be designated as a choke point by de-selecting the unrestricted flow check box from the Capacity tab of the Node Properties dialog box.

Chapter 15: Events

1. The zoom feature focuses the Workspace view on the node that is causing the dialog box to appear. The Go Back button allows a user to change the k-value of a node that has previously been set via the Node Paths dialog box. For users that have large RBDs with many nodes, this feature can be most useful.

2. The left configuration is more reliable regardless of the reliability value since it contains two more possible paths that keep the system in a functional state, yet it contains the same amount of equipment as the configuration on the right.

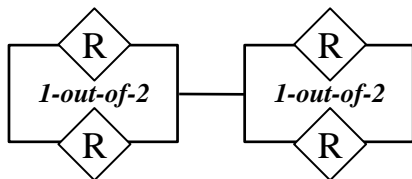
3.

Left Configuration

$$R_{\text{subsystem}} = 1 - (1 - R)^2$$

$$R_{\text{subsystem}} = 2R - R^2$$

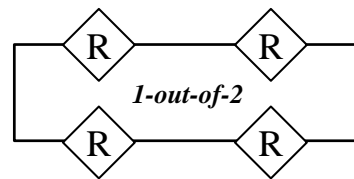
$$R_{\text{system}} = (2R - R^2)(2R - R^2) = 4R^2 - 4R^3 + R^4$$



Right Configuration

$$R_{\text{subsystem}} = 1 - (1 - R^2)^2$$

$$R_{\text{system}} = 2R^2 - R^4$$



Thus, it must be shown that

$$4R^2 - 4R^3 + R^4 > 2R^2 - R^4$$

$$4 - 4R + R^2 > 2 - R^2$$

$$(1 - R)(2 - 2R) > 0$$

$$1 - R > 0 \quad \text{or} \quad 2 - 2R > 0$$

which indicates that

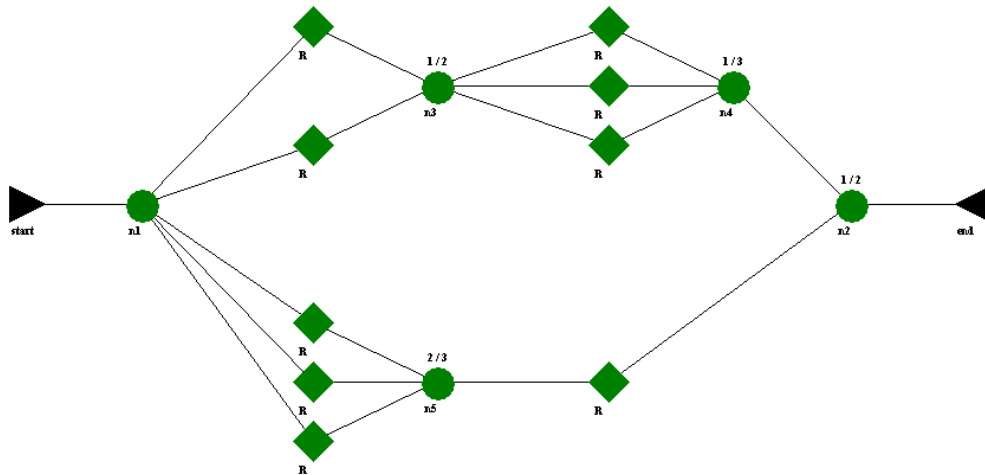
$$1 > R \text{ true} \quad 1 > R \text{ true}$$

Thus, the configuration on the left is always more reliable than the configuration on the right except for when R equals one, which is a trivial solution.

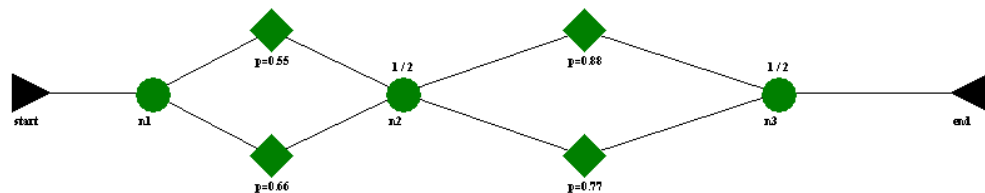
4. 100,000 trials. A general rule of thumb is described in the following equation:

$$\# \text{ trials} = 10^{(\text{digit of concern})+1}$$

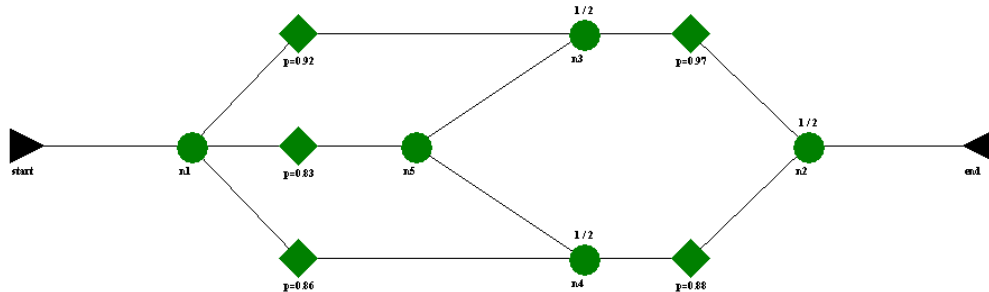
5. $R_{\text{system}} \approx 0.9749$



6. $R_{\text{system}} \approx 0.8229$



7. $R_{system} \approx 0.9926$



8. The equation for population standard deviation is $\sigma = \sqrt{\frac{\sum_{i=1}^n (X_i - \bar{X})^2}{n-1}}$.

The mean reliability is 0.85, as shown in Figure 15.6, so we have $\bar{X} = 0.85$. We know that this means that the trial was successful 170 times out of the 200, and unsuccessful 30 times, hence $X_i = 1.0$ for 170 of the i 's and $X_i = 0.0$ for 30 of the i 's. Thus

$$\sigma = \sqrt{\frac{\sum_{i=1}^{200} (X_i - 0.85)^2}{199}} = \sqrt{\frac{170(1.00 - 0.85)^2 + 30(0.00 - 0.85)^2}{199}} = 0.357967462$$

To compute the standard error of the mean, the equation is σ / \sqrt{n} . So we have

$$SEM = 0.357967462 / \sqrt{200} = 0.025312122$$

9. Using the t distribution, the lower bound is 0.800085576 and the upper bound is 0.899914424.

10. An event cannot go to standby. The three paths must have been set to be of equal priority (or the path with the event is a higher priority), and the event-path is the one receiving the “go to standby” message. One could correct this by setting the event-path to be the highest priority.

Chapter 16: Hierarchy

1. Upon modifying the Capacity3.rbd file and simulating, one finds availability of 0.949169788 and MDT of 9.412975, which does indeed match the results found in Figure 14.22. This is not surprising. Since the new RBD was created by collapsing existing blocks into hierarchies, the underlying ID numbers of all blocks has remained the same, hence one would expect the simulation results to be the same as in the RBD without the hierarchies.

3. The Electronics subsystem is yellow with a red Descendent Status Indicator. Thus, at a level below the main subsystem, there is a failure that produces the red DSI. This failure does not fail the main subsystem however, since this hierarchy is yellow opposed to red. Since the only redundancy of the Electronics subsystem is in the n25 node, who needs one of the two of the Batteries and AC-Power subsystems, there must be a failure of either the Batteries or the AC-Power. This could be caused by the failures of Batt-2A and Batt-2B. The Motors subsystem is yellow, hence it has some loss of redundancy that does not cause a failure. This could be due to a failure of the WindingOffset. The Mechanisms subsystem is orange, which indicates that the subsystem is down due to preventive maintenance. This could be a scheduled maintenance of the Dashpot component.

This figure can nearly be achieved in Failure Effects View, by failing the components mentioned above. However, Failure Effects View cannot send a component to preventive maintenance, so the orange color of the Mechanisms subsystem will not be seen in this view.

4. The Electronics subsystem is red, which means it has had a failure that prevents it from operating. As an example, this could be from a failure of the CPU in its Computer subsystem. The Motors subsystem is brown which means it is idle. In the current configuration of the Hierarchy4 RBD there is no dependency. To make this situation occur in the Failure Effects View, one would have to define some dependency in the Motors subsystem. For example, defining all the components in the Motors subsystem to be locally dependent, then making the Motors hierarchy element itself system dependent would create this situation.

Chapter 17: Phasing

1. Yes. Since a simulation length for a failure-truncated simulation is a random variable, the number of phases completed per trial is also a random variable.

2. Components that are not phased have a stress of 1.0.

3. 20 hours on average.

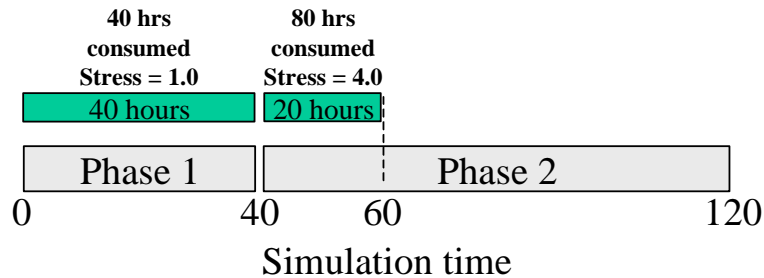
$$\text{phase mean life} = \frac{\text{average mean life}}{\text{stress}} = \frac{100\text{hrs}}{5.00} = 20.0\text{hrs}$$

4. 500 hours on average.

$$\text{phase mean life} = \frac{\text{average mean life}}{\text{stress}} = \frac{100\text{hrs}}{0.20} = 500.0\text{hrs}$$

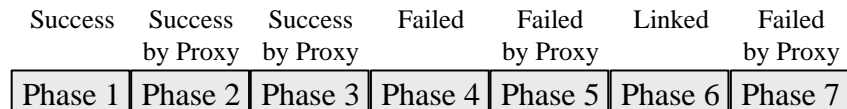
5. No. Over the long run (i.e., this phase occurring repeatedly), the mean life would average 50 hours, but a particular instance may or may not be exactly 50 hours.

6. Phase two, simulation time equal to 60 hours.



7. A linked node represents a subsystem that cannot cause a system failure but does drain logistic resources and thus incurs costs.

8.



9. Yes. Repairs are not stopped once they start.

10. The component costs \$45 per hour, regardless of the state in which it is in.

11. The first component still registers as a failed path. It will complete its repair, then will become operational and will not accumulate any life throughout the new phase. Since the second component is linked, the path of which it is a part is considered good. It will complete its repair, then turn green, but it too will not accumulate any life during this new phase.

12. Recall that the three components in the 500path power string are all locally dependent, with a 90% chance that they go idle, 10% chance that they fail. Each component is cut when the system enters phase 1. In the top figure, these three components are idle. Since any time the system enters phase 1 the components are cut, so the 500path node is necessarily down, these components receive the message to conduct their dependence action. All three in this case went idle. In the second figure, we see that the first two components are orange while the Bus_UR is brown. The SolarP_500W and the BatteriesNiCd must have been in preventive maintenance when this phase was entered. All maintenance activities, once begun, will be completed, hence they are completing their preventive maintenance. The Bus_UR has gone idle upon receiving its dependence message. In the third figure, we see the first two components are brown and the third is red. There are two ways this could occur. First, upon receiving the call to dependence, the first two went idle, and the third must have

had its 10% draw which told it to suffer a failure. Or, the Bus_UR could have been repairing when it entered this phase.

13. The color of the block will be blue, and the color of the node will be green, if the standby node uses priority return. If it does not use priority return, the block will remain green and the node will be green.

14. The block and node will both be green.

15. The cost for one day of operation is 20% that of one month because the initial acquisition costs are charged as soon as a simulation, of any length, is begun.

16. Yes. This is a method to change the status of elements during a phased simulation. Some might think this is being done when cuts or links are assigned to an element but recall cuts and links are not status changes but inactive situations.

17. If the phases are of fixed length, one could compute the amount of time required to complete a whole cycle, and run a time-terminated simulation with the termination time divisible by the cycle length. One could also conduct a cycle-terminated simulation, which will guarantee whole cycles being simulated whether the phases are of fixed length or not.

18. The average cycle length is 3010 hours, and the average mission length is 10 hours.

19. The average cycle length is 28 hours and the average mission length is 9.05 hours. Note that the average mission length would be 9.5 hours if the simulation was of near infinite duration.

20. The average cycle length is 35 hours and the average mission length over a 300 hour simulation is 28.33 hours.

21. The node is standby but the Tether block is independent. Hence, the TethPath path is being sent the message to go standby, but the Tether block is expressing attitude and will not listen. This could be corrected by making the Tether block locally dependent.

22. If a block is independent, and is up at the time the phase is entered, then there is no difference. If the block is repairing at the time the phase is entered, in the A:0.00 case it will remain failed making the path through the block bad until it repairs. If the phase were L:0.00, the block would continue to repair but the path through that block would be treated by the system as up. Also, if a block is dependent and set to A:0.00, it could go to the Idle state and cause the path through it to go down.

Chapter 19: Simulation Mechanics

1.

$$\text{Availability} = \frac{\text{uptime}}{\text{uptime} + \text{downtime}} = \frac{(55 + 1 + 7.2)}{(55 + 1 + 7.2) + (1.8 + 10)} = 0.842666$$

2.

$$\text{MTBDE} = \frac{\text{uptime}}{\text{number of system failures}} = \frac{(55 + 1 + 7.2)}{2} = 31.60 \text{ hours}$$

$$\text{MDT} = \frac{\text{downtime}}{\text{number of system failures}} = \frac{(1.8 + 10)}{2} = 5.90 \text{ hours}$$

3. Yellow time is defined to be the time when redundant or non-critical components are failed and the system is still operational. There were three periods when the system was in the yellow state.

$$\% \text{ Yellow} = \frac{\text{Yellow time}}{\text{Total time}} * 100 = \frac{(4.6 + 1.0 + 2.2)}{75} * 100 = 10.40 \%$$

4.

$$\text{MTBDE}_{\text{Block B}} = \frac{\text{uptime}_{\text{Block B}}}{\text{number of failures}_{\text{Block B}}} = \frac{(75 - 4.6)}{1} = 70.40 \text{ hours}$$

5.

$$\text{Availability} = \frac{\text{uptime}}{\text{uptime} + \text{downtime}} = \frac{(19 + 32.4 + 1 + 13.2)}{(19 + 32.4 + 1 + 13.2) + (4.6 + 0.8 + 4)} = 0.874666$$

6.

$$\text{MTBDE} = \frac{\text{uptime}}{\text{number of system failures}} = \frac{(19 + 32.4 + 1 + 13.2)}{3} = 21.87 \text{ hours}$$

$$\text{MDT} = \frac{\text{downtime}}{\text{number of system failures}} = \frac{(4.6 + 0.8 + 4)}{3} = 3.13 \text{ hours}$$

7. There are four periods when the system is yellow.

$$\% \text{ Yellow} = \frac{\text{Yellow time}}{\text{Total time}} * 100 = \frac{(1+1+6+2.2)}{75} * 100 = 13.60 \%$$

8.

$$MTBDE_{\text{Block B}} = \frac{\text{uptime}_{\text{Block B}}}{\text{number of failures}_{\text{Block B}}} = \frac{(55+1+7.2)}{2} = 31.60 \text{ hours}$$

Chapter 20: Miscellaneous Topics

1. All values will be displayed based on the current preference setting. The distribution parameter for Block A will be displayed as 0.005.

2. The number of pages down will be calculated based on the aspect ratio of the RBD. A short, wide RBD may be three pages across but still be one page down.

3. The formula for linear growth is:

$$\theta_{\text{new}} = [(Slope \times \# \text{ of failures}) + 1] \theta_{\text{base}}$$

$$\theta_{\text{new}} = [(1 \times 2) + 1] \times 100 = 300 \text{ hours}$$

4. The various growth or decay options and stress changes from phasing can each affect the mean life of a block simultaneously. In this case, the mean life would be 600 hours since the block is operating at half its normal stress.

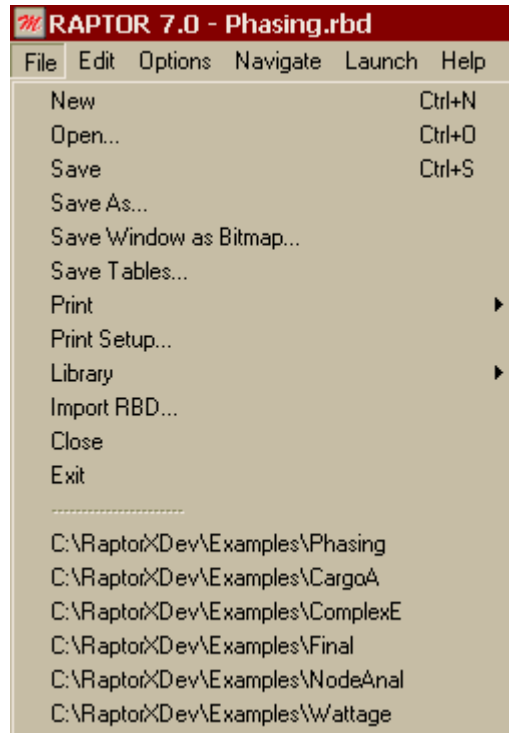
5. If two identical blocks are set to the same independent steam, they will fail and repair at exactly the same time throughout the simulation. If these two blocks are in series this is the same, system availability wise, as if there was only one block.

6. Preferences are saved in a separate file and are in effect for any RBD that is opened. System settings and block defaults are saved into an RBD file and are unique to that particular reliability block diagram.

7. When a failure-truncated simulation is set to run until X failures have occurred, it will actually run until X repairs have been completed. The results, in most simulations, are the same number. Counting the first failure could result in the final results being based on X repair times but only $X-1$ failure times if the system starts down.

Workspace View Menu Bar

File Menu



New menu item: Opens a new and therefore empty RBD file. If a modified RBD file is currently open, a save file dialog box will appear before closing the current file. This allows a user to save the changes incorporated in the currently opened RBD file.

Open menu item: Opens an existing RBD file. If a modified RBD file is currently open, a save file dialog box will appear before opening another RBD file. Then the Windows[®] open file dialog box appears. This allows a user to select an RBD file from a storage location (i.e., C-drive, A-drive, etc.). If the open file process is canceled, a new RBD file is loaded into the Workspace.

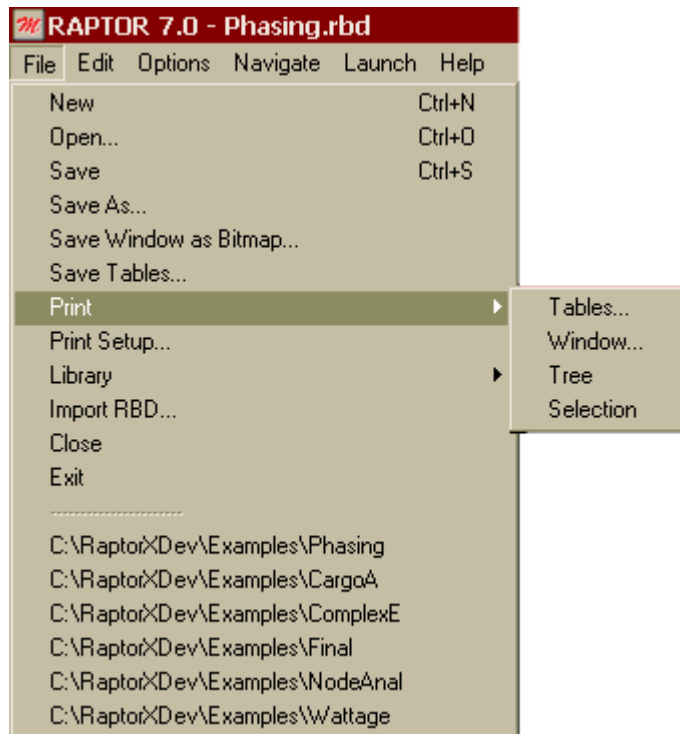
Save menu item: Saves the RBD file currently displayed within the Workspace with the same filename it previously possessed. If an RBD file has not been previously saved, Raptor prompts the user for a filename. All RBD files are saved with an RBD filename extension.

Save As menu item: Saves the currently displayed RBD file with a new filename.

Save Window as Bitmap menu item: Saves the image of the window currently displayed within the Simulation View to a bitmap file. The default size of the bitmap is determined by the number of grids an RBD image spans. Each grid represents 25 pixels by 25 pixels in the bitmap file. The size can be adjusted through the Bitmap Size dialog box. However, the original aspect ratio of the image must always be maintained.

Save Tables menu item: Opens the Save.TXT Options dialog box, which allows the tables of the currently displayed RBD file to be saved in text form. For each file selected, the Windows® save file dialog box will appear allowing a user to name each file.

Print submenu: The Print submenu (shown below) allows the user to select from four different print options (Tables, Window, Tree, or Selection).



Print Tables menu item: Opens the Print Selections dialog box, which allows the printing of tables of the currently displayed RBD file.

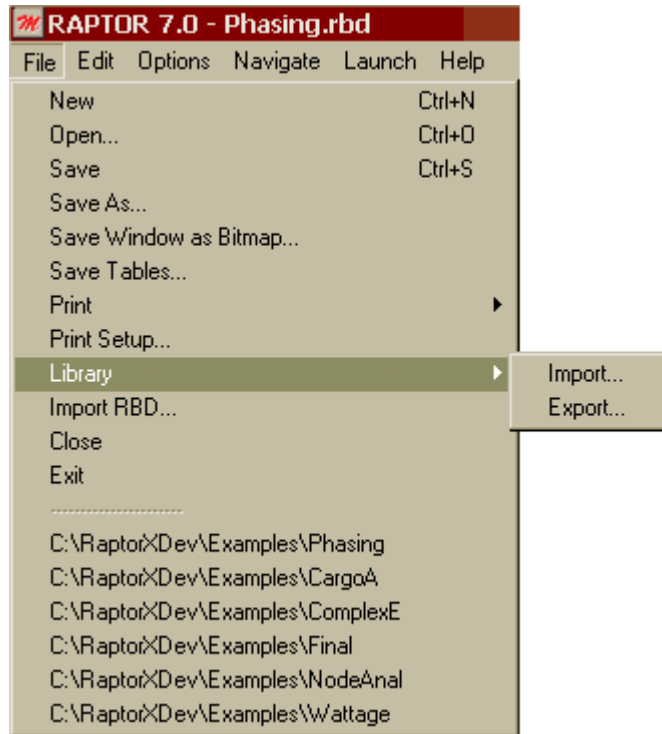
Print Window menu item: Allows the user to print the currently displayed window to their default printer.

Print Tree menu item: Prints the hierarchy tree of the currently displayed RBD.

Print Selection menu item: Prints the highlighted portion of the currently displayed window. A portion of a window can be selected via the rubber-band box by clicking and holding the left mouse button down and then dragging the mouse over a portion of the window. Another method of selection is to depress the Ctrl key while selecting different elements with the left mouse button.

Print Setup menu item: Opens the Windows® printer setup dialog box so that a default printer and page orientation can be chosen.

Library submenu: The Library submenu (shown below) allows the user to select from one of two different library options (Import or Export).



Import menu item: Displays the Windows® open file dialog box which allows a user to load a library file. A loaded library can provide a set of block templates to create new blocks with pre-specified data fields.

Export menu item: Exports the characteristics of all the blocks within the Workspace to a library file. The Windows® save file dialog box appears and allows a user to name the library file.

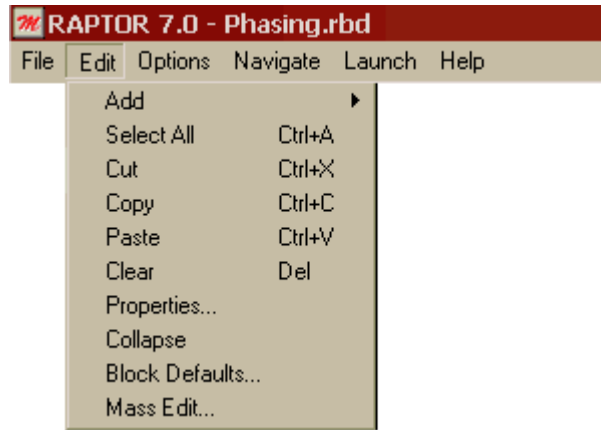
Import RBD menu item: Merges a pre-existing, Raptor 7.0 RBD file into the currently displayed RBD file. The RBD is brought in as a hierarchy, whose default name is the same as that of the RBD.

Close menu item: Clears the Workspace. If a modified RBD file is currently open, a save file dialog box will appear before closing the current file.

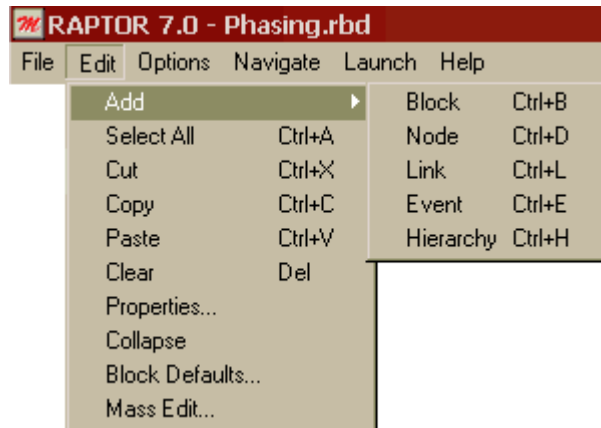
Exit menu item: Exits a user from the Raptor software. If a modified RBD file is currently open, a save file dialog box will appear before closing the current file.

Recently used RBD file list: Contains a list of the last six RBD files that have been opened in Raptor.

Edit Menu



Add submenu: The Add submenu (shown below) allows the user to add one of five types of elements to the Workspace.



Add Block menu item: Changes the Workspace to the block cursor mode, which allows a user to add new blocks to an RBD file. The normal pointer cursor changes to the block cursor and will add a block at the location of a left click of the mouse. Once a block is placed within the Workspace, the Block Properties dialog box will be displayed awaiting input. The Workspace remains in add block mode until a right mouse click occurs.

Add Node menu item: Changes the Workspace to the node cursor mode, which allows a user to add new nodes to an RBD file. The normal pointer cursor changes to the node cursor and will add a node at the location of a left click of the mouse. Once a node is placed within the Workspace, the Node Properties dialog box will be displayed awaiting input. The Workspace remains in add node mode until a right mouse click occurs.

Add Link menu item: Changes the Workspace to the link cursor mode, which allows a user to draw new links within an RBD file. Links are drawn by selecting a block, node, event, or hierarchy (left mouse click and hold), dragging a link to a second block, node, event, or hierarchy, and then releasing the held mouse button while over the second object. The Workspace remains in add link mode until a right mouse click occurs.

Add Event menu item: Changes the Workspace to the event cursor mode, which allows a user to add new events to an RBD file. The normal pointer cursor changes to the event cursor and will add an event at the location of a left click of the mouse. Once an event is placed within the Workspace, the Event Properties dialog box will be displayed awaiting input. The Workspace remains in add event mode until a right mouse click occurs.

Add Hierarchy menu item: Changes the Workspace to the hierarchy cursor mode, which allows a user to add new hierarchies to an RBD file. The normal pointer cursor changes to the hierarchy cursor and will add a hierarchy at the location of a left click of the mouse. Once a hierarchy is placed within the Workspace, the Hierarchy Properties dialog box will be displayed awaiting input. The Workspace remains in add hierarchy mode until a right mouse click occurs.

Select all menu item: Highlights all blocks, nodes, events, and hierarchies contained within the Workspace.

Cut menu item: Removes any highlighted elements from the Workspace while placing the removed items into the Raptor paste buffer.

Copy menu item: Copies the selected elements into the Raptor paste buffer.

Paste menu item: Pastes elements from the Raptor paste buffer into the Workspace.

Clear menu item: Removes any selected elements from the Workspace without placing the removed items into the Raptor paste buffer.

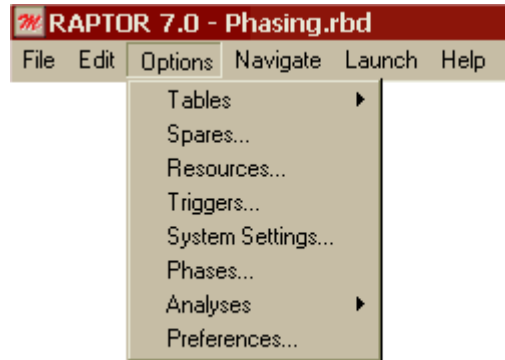
Properties menu item: Opens the selected item's Block, Node, Event, or Hierarchy Properties dialog box as appropriate. This menu item is deactivated if more than one element is highlighted.

Collapse menu item: Places the selected objects into a new hierarchy. Once the hierarchy is placed within the Workspace, the Hierarchy Properties dialog box will be displayed awaiting input.

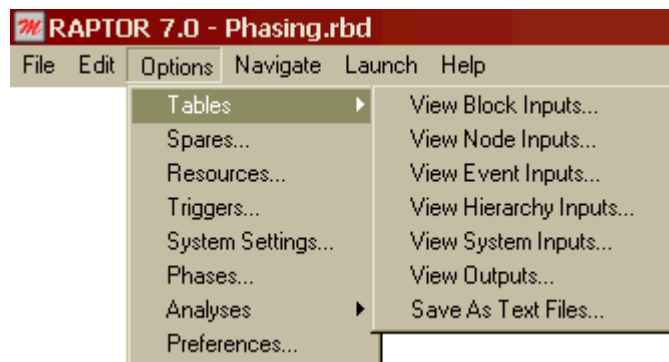
Block Defaults menu item: Opens the Block Properties dialog box and allows the default block attributes to be modified. Any blocks added to the Workspace after a change to the block defaults will possess the new block default characteristics.

Mass Edit menu item: Opens the Mass Edit dialog box. This menu item is deactivated if fewer than two elements are selected.

Options Menu



Tables submenu: The tables submenu (shown below) allows the user to view one of six table types as well as save these tables as text files.



View Block Inputs menu item: Opens the Block Input Tables dialog box. This menu item is deactivated if the RBD does not contain any blocks.

View Node Inputs menu item: Opens the Node Input Tables dialog box. This menu item is deactivated if the RBD does not contain any nodes.

View Event Inputs menu item: Opens the Event Input Tables dialog box. This menu item is deactivated if the RBD does not contain any events.

View Hierarchy Inputs menu item: Opens the Hierarchy Input Tables dialog box. This menu item is deactivated if the RBD does not contain any hierarchies.

View System Inputs menu item: Opens the System Input Tables dialog box.

View Outputs menu item: Opens the Output Tables dialog box. This menu item is deactivated if a simulation has not been conducted since the currently displayed RBD was last opened.

Save As Text Files menu item: Opens the Save.TXT Options dialog box, which allows the tables of the currently displayed RBD file to be saved in text form. For each file selected, the Windows[®] save file dialog box will appear allowing a user to name each file.

Spares menu item: Opens the Spare Pools dialog box.

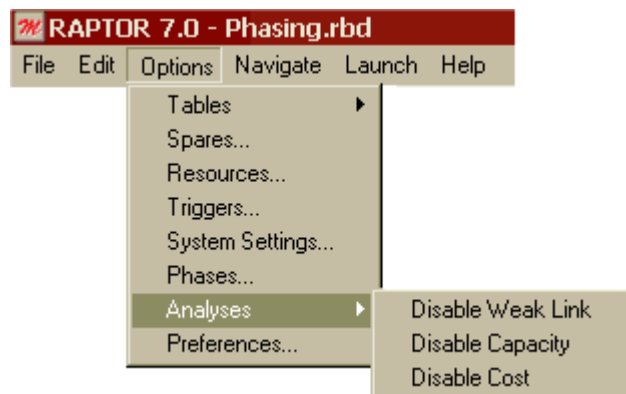
Resources menu item: Opens the Resource Pools dialog box.

Triggers menu item: Opens the Define Triggers dialog box.

System Settings menu item: Opens the System Settings dialog box.

Phases menu item: Opens the Phasing dialog box.

Analyses submenu: The Analyses submenu (shown below) allows the user to turn the Weak Link Analysis, Capacity Analysis, and Cost Analysis features on or off.



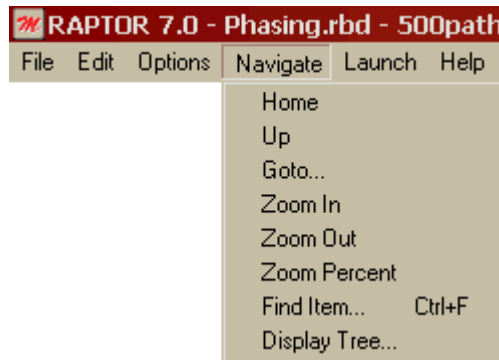
Enable/Disable Weak Link menu item: Turns the Weak Link Analysis feature on/off.

Enable/Disable Capacity menu item: Turns the Capacity Analysis feature on/off.

Enable/Disable Cost menu item: Turns the Cost Analysis feature on/off.

Preferences menu item: Opens the Preferences dialog box.

Navigate Menu



Home menu item: When inside a hierarchy, instructs Raptor to return to the top level. This button is deactivated if the current window is not inside a hierarchy.

Up menu item: When inside a hierarchy, instructs Raptor to go to the level immediate above the current window. This button will be deactivated if the current window is not inside a hierarchy.

Goto menu item: Instructs Raptor to change the view to that of the hierarchy specified in the Goto Hierarchy drop down list box.

Zoom In menu item: Decreases the amount of grids displayed by 5% or a decrease of 6 grids in width.

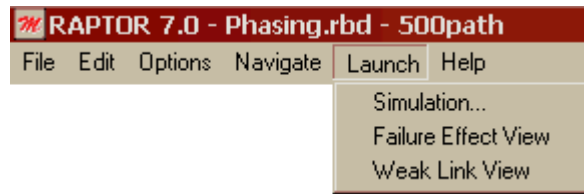
Zoom Out menu item: Increases the amount of grids displayed by 5% or an increase of 6 grids in width.

Zoom Percent menu item: Instructs Raptor to change the view to that specified in the Zoom Percent dialog box.

Find Item menu item: Opens the Find Item dialog box that contains a drop-down list box of all blocks, nodes, events, and hierarchies contained within the RBD.

Display Tree menu item: Opens the Tree dialog box that displays the hierarchical structure of the current RBD.

Launch Menu



Simulation menu item: Opens the Simulation Options dialog box allowing a simulation to be conducted.

Failure Effect View menu item: This menu item switches from the Workspace View to the Failure Effects View. The Failure Effects View is a view in which the user can choose which components to fail to see what affect that has on the system.

Weak Link View menu item: This item switches from the Workspace view to the Weak Link Analysis view. The Weak Link Analysis View is a post Simulation view that displays red, yellow and green color threshold information from the previous simulation. This menu item is deactivated if a simulation with weak link analysis has not been conducted during the current Raptor session.

Help Menu



Help Index menu item: Opens the Raptor help file.

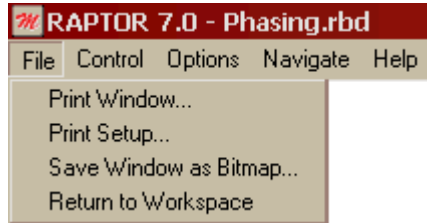
About menu item: Opens a scrolling dialog box that displays the current developers, past developers, and testers. The list also includes groups or persons that have provided inspiration or humor during those difficult times associated with the development of Raptor.

Visit Raptor Webpage menu item: Transports a user to the Raptor home page by using their default web browser. The latest training and consulting information are provided on this web page.

Raptor Reference Manual menu item: Opens the Raptor Reference Manual in pdf format. Adobe Acrobat reader must be installed for this option to work.

Simulation View Menu Bar

File Menu



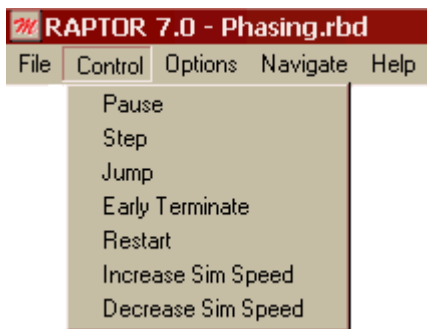
Print Window menu item: Opens the Print Window dialog box, which allows printing of the window that is currently displayed.

Print Setup menu item: Opens the Windows[®] printer setup dialog box so that a default printer and page orientation can be chosen.

Save Window as Bitmap menu item: Saves the image of the window currently displayed within the Simulation View to a bitmap file. The default size of the bitmap is determined by the number of grids an RBD image spans. Each grid represents 25 pixels by 25 pixels in the bitmap file. The size can be adjusted through the Bitmap Size dialog box. However, the original aspect ratio of the image must always be maintained.

Return to Workspace menu item: Returns focus from the Simulation View to the Workspace View. This menu item is deactivated until a simulation is completed or early terminated.

Control Menu



Resume or Pause menu item: Resumes or pauses a simulation.

Step menu item: This menu item is only activated while a simulation is paused. If selected, the simulation clock is advanced to the next simulation event that results in a color change of a block, node, or event.

Jump menu item: This menu item is only activated while a simulation is paused. If selected, the simulation is advanced to the beginning of the next trial.

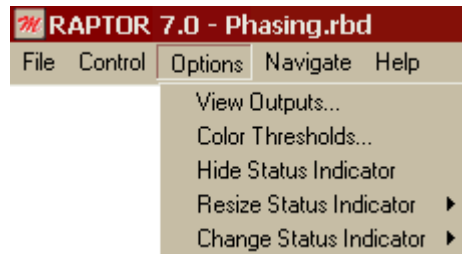
Early Terminate menu item: Terminates a simulation before a pre-specified ending simulation condition. Results accumulated until the termination point are calculated and are available.

Restart menu item: Restarts a simulation without returning to the Workspace. Displays the Resimulate Options dialog box which contains a subset of alternatives available in the Simulation Options dialog box.

Increase Sim Speed menu item: Increases the simulation speed by a factor of ten times the previous speed setting. The maximum simulation speed is 10,000,000. This menu item is deactivated if a simulation without graphics is conducted or when a simulation has been terminated.

Decrease Sim Speed menu item: Decreases the simulation speed by a factor of ten times the previous speed setting. The minimum simulation speed is 0.000001. This menu item is deactivated if a simulation without graphics is conducted or when a simulation has been terminated.

Options Menu

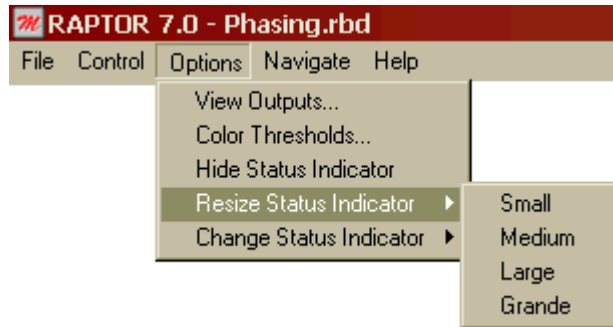


View Outputs menu item: This menu option is only available after a simulation is completed or early terminated. When selected, the Output Tables dialog box is displayed.

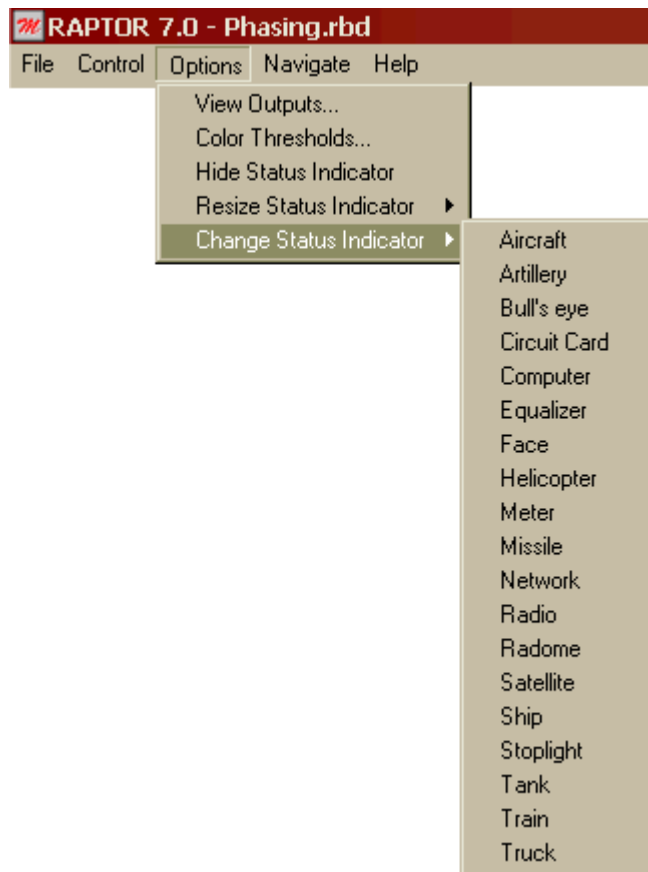
Color Thresholds menu item: This menu item is only available following the completion of a simulation that conducted Weak Link analysis. Selecting this item displays a System Settings dialog box with only the fourth tab available. This option provides a capability for changing the red, yellow and green color thresholds.

Show/Hide Status Indicator menu item: This menu item allows the user to show or hide the system status indicator. The default is shown.

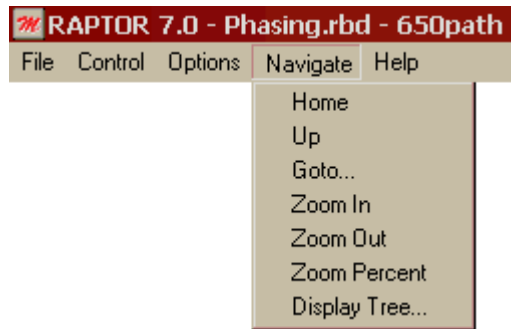
Resize Status Indicator submenu: This submenu (shown below) allows the user to select the size of the status indicator. There are four different sizes that a status indicator can be. The default size is Medium.



Change Status Indicator submenu: This submenu (shown below) allows the user to change which status indicator is displayed during a simulation. There are 19 different images that the status indicator can be. The default image is the Equalizer.



Navigate Menu



Home menu item: When inside a hierarchy, instructs Raptor to return to the top level. This button is deactivated if the current window is not inside a hierarchy.

Up menu item: When inside a hierarchy, instructs Raptor to go to the level immediate above the current window. This button will be deactivated if the current window is not inside a hierarchy.

Goto menu item: Instructs Raptor to change the view to that of the hierarchy specified in the Goto Hierarchy drop down list box.

Zoom In menu item: Decreases the displayed view space by 5%, equivalent to a decrease of 6 grids in width.

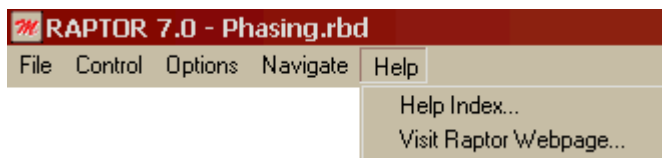
Zoom Out menu item: Increases the displayed view space by 5% equivalent to an increase of 6 grids in width.

Zoom Percent menu item: Instructs Raptor to change the view to that specified in the Zoom Percent dialog box.

Find Item menu item: Opens the Find Item dialog box that contains a drop-down list box of all blocks, nodes, events, and hierarchies contained within the RBD.

Display Tree menu item: Opens the Tree dialog box that displays the hierarchical structure of the current RBD.

Help Menu

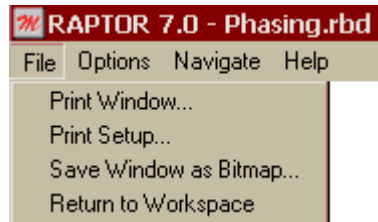


Help menu item: Opens the Raptor help file to a specific page concerning simulation capabilities.

Visit Raptor Webpage menu item: Transports a user to the Raptor home page by using their default web browser. The latest training and consulting information are provided on this web page.

Weak Link View Menu Bar

File Menu



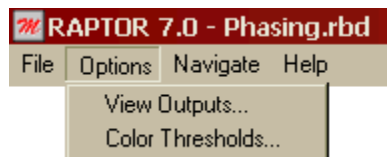
Print Window menu item: Opens the Print Window dialog box, which allows printing of the window that is currently displayed.

Print Setup menu item: Opens the Windows[®] printer setup dialog box so that a default printer and page orientation can be chosen.

Save Window as Bitmap menu item: Saves the image of the window currently displayed within the Simulation View to a bitmap file. The default size of the bitmap is determined by the number of grids an RBD image spans. Each grid represents 25 pixels by 25 pixels in the bitmap file. The size can be adjusted through the Bitmap Size dialog box. However, the original aspect ratio of the image must always be maintained.

Return to Workspace menu item: Returns focus from the Weak Link View to the Workspace View. This menu item is deactivated until a simulation is completed or early terminated.

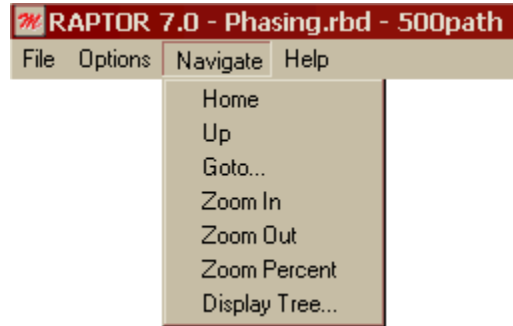
Options Menu



View Outputs menu item: When selected, the Output Tables dialog box is displayed.

Color Thresholds menu item: Selecting this item displays a System Settings dialog box with only the fourth tab available. This option provides a capability for changing the red, yellow and green color thresholds.

Navigate Menu



Home menu item: When inside a hierarchy, instructs Raptor to return to the top level. This button is deactivated if the current window is not inside a hierarchy.

Up menu item: When inside a hierarchy, instructs Raptor to go to the level immediate above the current window. This button will be deactivated if the current window is not inside a hierarchy.

Goto menu item: Instructs Raptor to change the view to that of the hierarchy specified in the Goto Hierarchy drop down list box.

Zoom In menu item: Decreases the amount of displayed view space by 5%, equivalent to a decrease of 6 grids in width.

Zoom Out menu item: Increases the amount of displayed view space by 5%, equivalent to an increase of 6 grids in width.

Zoom Percent menu item: Instructs Raptor to change the view to that specified in the Zoom Percent dialog box.

Find Item menu item: Opens the Find Item dialog box that contains a drop-down list box of all blocks, nodes, events, and hierarchies contained within the RBD.

Display Tree menu item: Opens the Tree dialog box that displays the hierarchical structure of the current RBD.

Help Menu

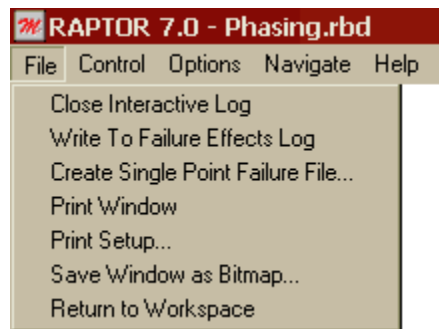


Help Index menu item: Opens the Raptor help file to a specific page concerning weak link analysis capabilities.

Visit Raptor Webpage menu item: Transports a user to the Raptor home page by using their default web browser. The latest training and consulting information are provided on this web page.

Failure Effects View Menu Bar

File Menu



Open or Close Interactive Log menu item: This menu item toggles between the open and close functions for an interactive log. If one is not already opened, it reads “Open Interactive Log” and upon selection, will open an interactive log. If an interactive log is already opened, this menu item reads “Close Interactive Log” and will close the interactive log upon selection.

Write To Failure Effects Log menu item: Upon selection, the affects on the system of the currently failed components will be written to the interactive log. This menu item is deactivated if an interactive log is not already opened.

Create Single Point Failure File menu item: Upon selection, requests a file name and creates a text file listing all of the single points of failure in the system, and their criticality toward system failures.

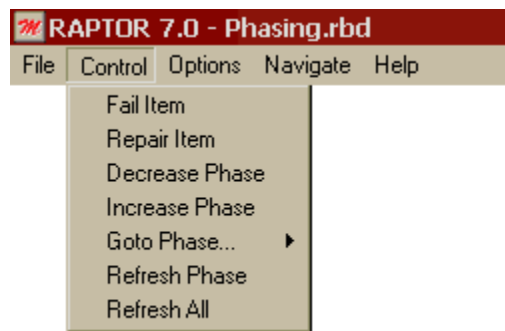
Print Window menu item: Opens the Print Window dialog box, which allows printing of the window that is currently displayed.

Print Setup menu item: Opens the Windows® printer setup dialog box so that a default printer and page orientation can be chosen.

Save Window as BMP menu item: Saves the image of the window currently displayed within the Simulation View to a bitmap file. The default size of the bitmap is determined by the number of grids an RBD image spans. Each grid represents 25 pixels by 25 pixels in the bitmap file. The size can be adjusted through the Bitmap Size dialog box. However, the original aspect ratio of the image must always be maintained.

Return to Workspace menu item: Returns focus from the Failure Effects View to the Workspace View.

Control Menu



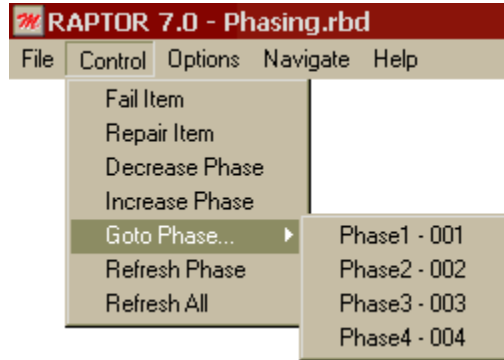
Fail Item menu item: Opens the Fail Item dialog box, which contains a drop-down list box of all blocks and events within the RBD. Upon selection of a component, if that component is operating, it will fail. If it is already failed, no action is taken.

Repair Item menu item: Opens the Repair Item dialog box, which contains a drop-down list box of all blocks and events within the RBD. Upon selection of a component, if that component is failed, it will be repaired. If it is already repaired, no action is taken.

Decrease Phase menu item: Decreases the current phase by one. This menu item is deactivated if no phases are defined within the RBD.

Increase Phase menu item: Increases the current phase by one. This menu item is deactivated if no phases are defined within the RBD.

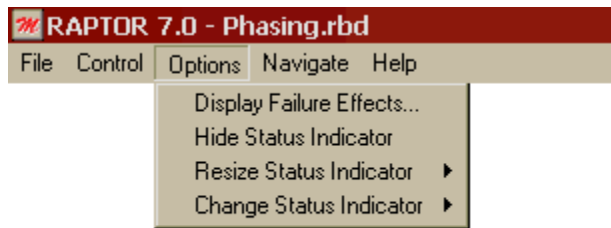
Goto Phase submenu: This submenu (shown below) provides a list of all defined phases so the user can select a specific phase to go to.



Refresh Phase menu item: Refreshes all components like new, but remains in the current phase. This menu item is deactivated if no phases are defined.

Refresh All menu item: Refreshes all components like new and returns to the first phase. This menu item is deactivated if no phases are defined.

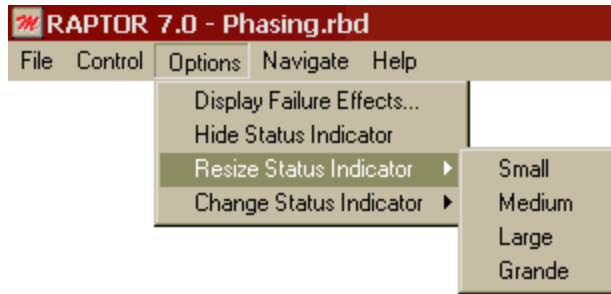
Options Menu



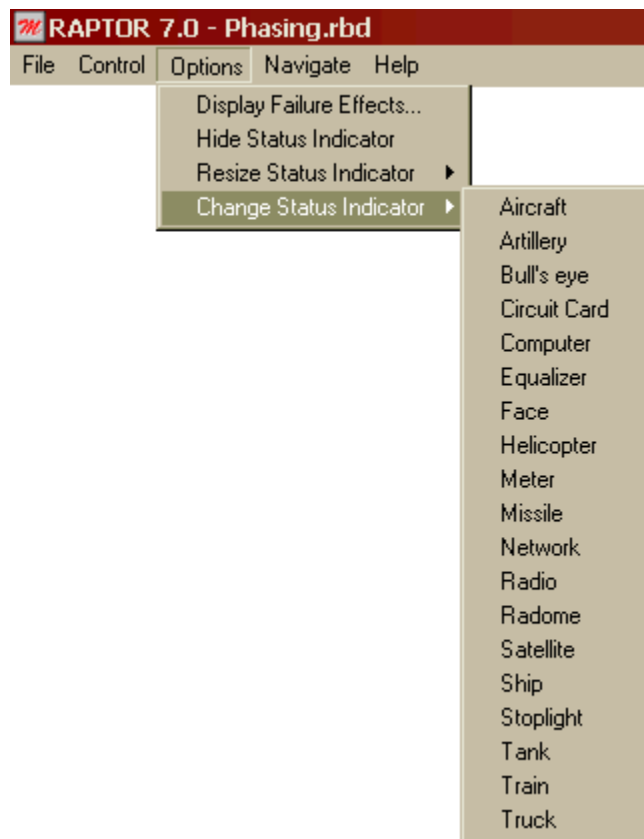
Display Failure Effects menu item: Displays how the currently failed components affect the system.

Show/Hide Status Indicator menu item: This menu item allows the user to show or hide the system status indicator. The default is shown.

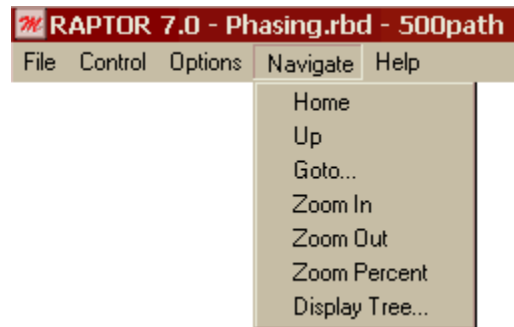
Resize Status Indicator submenu: This submenu (shown below) allows the user to select the size of the status indicator. There are four different sizes that a status indicator can be. The default size is Medium.



Change Status Indicator submenu: This submenu (shown below) allows the user to change which status indicator is displayed during a simulation. There are 19 different images that the status indicator can be. The default image is the Equalizer.



Navigate Menu



Home menu item: When inside a hierarchy, instructs Raptor to return to the top level. This button is deactivated if the current window is not inside a hierarchy.

Up menu item: When inside a hierarchy, instructs Raptor to go to the level immediate above the current window. This button will be deactivated if the current window is not inside a hierarchy.

Goto menu item: Instructs Raptor to change the view to that of the hierarchy specified in the Goto Hierarchy drop down list box.

Zoom In menu item: Decreases the amount of grids displayed by 5% or a decrease of 6 grids in width.

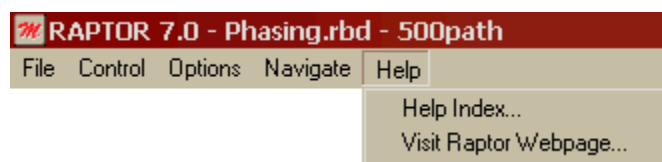
Zoom Out menu item: Increases the amount of grids displayed by 5% or an increase of 6 grids in width.

Zoom Percent menu item: Instructs Raptor to change the view to that specified in the Zoom Percent dialog box.

Find Item menu item: Opens the Find Item dialog box that contains a drop-down list box of all blocks, nodes, events, and hierarchies contained within the RBD.

Display Tree menu item: Opens the Tree dialog box that displays the hierarchical structure of the current RBD.

Help Menu



Help Index menu item: Opens the Raptor help file to a specific page concerning failure effects capabilities.

Visit Raptor Webpage menu item: Transports a user to the Raptor home page by using their default web browser. The latest training and consulting information are provided on this web page.

Ao Graph Menu Bar

File Menu



Print submenu: The Print submenu (shown below) allows the user to select from four different print options (Per trial Ao, Interval Ao, Multi-run Ao, All Ao Graphs).



Per Trial Ao menu item: Prints the Per Trial Ao plot as it appears within the graph window.

Interval Ao menu item: Prints the Interval Ao plot as it appears within the graph window.

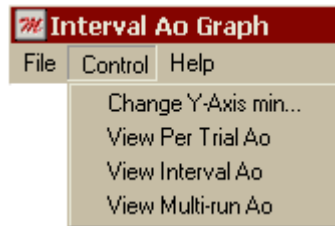
Multi-run Ao menu item: Prints the Multi-run Ao plot as it appears within the graph window.

All Ao menu item: Prints all Ao plots as they appear within the graph window.

Print Setup menu item: Opens the Windows[®] printer setup dialog box so that a default printer and page orientation can be chosen.

Close menu item: This option removes the graph window from the screen. The Simulation view is then expanded to fill the entire screen. Any re-simulations started will not display the graph window.

Control Menu



Change Y-Axis menu item: Displays the Set Y-Axis Values dialog box and allows the minimum and maximum values of the y-axis to be modified.

View Per Trial Ao menu item: Displays the Per Trial Ao plot in the graph window. This is a plot of the red, yellow, and green percentage values versus the simulation time for the current run. The line separating the red region from the yellow and green regions represent the system availability versus simulation time.

View Interval Ao menu item: Displays the Interval Ao plot in the graph window. This is a plot of the availability of a system during the previous interval. For example, if the user selects an Ao plot sample rate of 50.0 hours the value plotted at simulation time 150 is the availability observed between times 100 and 150.

View Multi-run Ao menu item: Displays the Multi-run Ao plot in the graph window. This plot reflects data obtained from each run of simulation. For each time displayed on the x-axis, the minimum, maximum, and average availability obtained from all runs are plotted. As these lines converge and become parallel, the plot is indicating that a system is approaching a steady-state condition.

Help Menu



Help Index menu item: Opens the Raptor help file to a specific page concerning graphing availability during a simulation.

Visit Raptor Webpage menu item: Transports a user to the Raptor home page by using their default web browser. The latest training and consulting information are provided on this web page.

Workspace View Toolbar



New button: Opens a new and therefore empty RBD file. If a modified RBD file is currently open, a save file dialog box will appear before closing the current file. This allows a user to save the changes incorporated in the currently opened RBD file.



Open button: Opens an existing RBD file. If a modified RBD file is currently open, a save file dialog box will appear before opening another RBD file. Then the Windows[®] open file dialog box appears. This allows a user to select an RBD file from a storage location (e.g., C-drive). If the open file process is canceled, a new RBD file is loaded into the Workspace.



Save button: Saves the RBD file currently displayed within the Workspace with the same filename it previously possessed. If an RBD file has not been previously saved, Raptor prompts the user for a filename. All RBD files are saved with an RBD filename extension.



Print Window button: Opens the Print Window dialog box, which allows printing of the window that is currently displayed.



Cut button: Removes any highlighted elements from the Workspace while placing the removed items into the Raptor paste buffer.



Copy button: Copies the selected elements into the Raptor paste buffer.



Paste button: Pastes elements from the Raptor paste buffer into the Workspace.




Properties button: Opens the selected item's Block, Node, Event, or Hierarchy Properties dialog box as appropriate. This button is deactivated if more than one element is highlighted.





Add Block button: Changes the Workspace to the block cursor mode, which allows a user to add new blocks to an RBD file. The normal pointer cursor changes to the block cursor and will add a block at the location of a left click of the mouse. Once a block is placed within the Workspace, the Block Properties dialog box will be displayed awaiting input. The Workspace remains in add block mode until a right mouse click occurs.





Add Node button: Changes the Workspace to the node cursor mode, which allows a user to add new nodes to an RBD file. The normal pointer cursor changes to the node cursor and will add a node at the location of a left click of the mouse. Once a node is placed within the Workspace, the Node Properties dialog box will be displayed awaiting input. The Workspace remains in add node mode until a right mouse click occurs.


- 


Add Link button: Changes the Workspace to the link cursor mode, which allows a user to draw new links within an RBD file. Links are drawn by selecting a block, node, event or hierarchy (left mouse click and hold), dragging a link to a second block, node, event or hierarchy, and then releasing the held mouse button while over the second object. The Workspace remains in add link mode until a right mouse click occurs.
- 


Add Event button: Changes the Workspace to the event cursor mode, which allows a user to add new events to an RBD file. The normal pointer cursor changes to the event cursor and will add an event at the location of a left click of the mouse. Once an event is placed within the Workspace, the Event Properties dialog box will be displayed awaiting input. The Workspace remains in add event mode until a right mouse click occurs.
- 


Add Hierarchy button: Changes the Workspace to the hierarchy cursor mode, which allows a user to add new hierarchies to an RBD file. The normal pointer cursor changes to the hierarchy cursor and will add a hierarchy at the location of a left click of the mouse. Once a hierarchy is placed within the Workspace, the Hierarchy Properties dialog box will be displayed awaiting input. The Workspace remains in add hierarchy mode until a right mouse click occurs.
- 


View Block Inputs button: Opens the Block Input Tables dialog box. This button is deactivated if the RBD does not contain at least one block.
- 


View Outputs button: Opens the Output Tables dialog box. This button is deactivated if a simulation has not been conducted since Raptor was last opened.
- 


Spares button: Opens the Spare Pools dialog box.
- 


Resources button: Opens the Resource Pools dialog box.
- 

Triggers button: Opens the Triggers dialog box.
- 












Phasing button: Opens the Phasing dialog box.
- 

Weak Link Analysis button: Instructs Raptor to perform a weak link analysis during the next simulation. The color thresholds assigned in the System Settings dialog box will be used.
- 



Capacity Analysis button: Instructs Raptor to perform a capacity analysis during the next simulation.
- 













Cost Analysis button: Instructs Raptor to perform a cost analysis during the next simulation.
- 


Home button: When inside a hierarchy, instructs Raptor to return to the top level. This button is deactivated if the current window is not inside a hierarchy.


-  Up button: When inside a hierarchy, instructs Raptor to go to the level immediate above the current window. This button will be deactivated if the current window is not inside a hierarchy.
-  Goto Hierarchy button: Instructs Raptor to change the view to that of the hierarchy specified in the Goto Hierarchy drop down list box.
-  Zoom Out button: Increases the amount of grids displayed by 5% or an increase of 6 grids in width.
-  Zoom In button: Decreases the amount of grids displayed by 5% or a decrease of 6 grids in width.
-  Zoom to Fit button: Changes the zoom level to one in which the entire RBD fits on the screen.
-  Find Item button: Opens the Find Item dialog box that contains a drop-down list box of all blocks, nodes, events and hierarchies within an RBD.
-  Display Tree button: Opens the Tree dialog box that displays the hierarchical tree structure of the current RBD.
-  Simulate button: This button opens the Simulate Options dialog box, allowing a simulation to be conducted.
-  Failure Effects View button: This button switches from the Workspace View to the Failure Effects View. The Failure Effects View is a view in which the user can choose which components to fail to see what affect that has on the system.
-  Weak Link Analysis View button: This button switches from the Workspace View to the Weak Link Analysis View. The Weak Link Analysis View is a post Simulation view that displays red, yellow and green color threshold information from the previous simulation. This button is deactivated if a simulation with weak link analysis has not been conducted during the current Raptor session.
-  Help button: Opens the Raptor help file.


Simulation View Toolbar


-  Print Window button: Opens the Print Window dialog box, which allows printing of the window that is currently displayed.
-  Return to Workspace button: Returns focus from the Simulation View to the Workspace View. This button is deactivated until a simulation is completed or early terminated.


-  Pause button: Pauses and resumes a simulation.
-  Step button: This button is only activated while a simulation is paused. If selected, the simulation clock is advanced to the next simulation event that results in a color change of a block, node, or event.
-  Jump button: This button is only activated while a simulation is paused. If selected, the simulation is advanced to the beginning of the next trial.
-  Early Terminate button: Terminates a simulation before a pre-specified ending simulation condition. Results accumulated until the termination point are calculated and are available.
-  Restart button: Restarts a simulation without returning to the Workspace. Displays the Resimulate Options dialog box which contains a subset of alternatives available in the Simulation Options dialog box.
-  Decrease Sim Speed button: Decreases the simulation speed by a factor of ten times the previous speed setting. The minimum simulation speed is 0.000001. This button is deactivated if a simulation without graphics is conducted or when a simulation has been terminated.
-  Increase Sim Speed button: Increases the simulation speed by a factor of ten times the previous speed setting. The maximum simulation speed is 10,000,000. This button is deactivated if a simulation without graphics is conducted or when a simulation has been terminated.
-  View Outputs button: This button is only available after a simulation is completed or early terminated. When selected, the Output Tables dialog box is displayed.
-  Color Thresholds button: This button is only available following the completion of a simulation that conducted weak link analysis. Selecting this button displays the System Settings dialog box with only the fourth tab available. This option provides a capability for changing the red, yellow and green color thresholds.
-  Home button: When inside a hierarchy, instructs Raptor to return to the top level. This button is deactivated if the current window is not inside a hierarchy.
-  Up button: When inside a hierarchy, instructs Raptor to go to the level immediate above the current window. This button will be deactivated if the current window is not inside a hierarchy.
-  Goto Hierarchy button: Instructs Raptor to change the view to that of the hierarchy specified in the Goto Hierarchy drop down list box.

 Zoom Out button: Increases the displayed view space by 5%, equivalent to an increase of 6 grids in width.


 Zoom In button: Decreases the displayed view space by 5%, equivalent to a decrease of 6 grids in width.


 Zoom to Fit button: Changes the zoom level to one in which the entire RBD fits on the screen.


 Display Tree button: Opens the Tree dialog box that displays the hierarchical tree structure of the current RBD.


 Help button: Opens the Raptor help file to a specific page concerning simulation capabilities.


Weak Link View Toolbar


 Print Window button: Opens the Print Window dialog box, which allows printing of the window that is currently displayed.


 Return to Workspace button: Returns focus from the Weak Link View to the Workspace View.


 View Outputs button: When selected, the Output Tables dialog box is displayed.


 View Color Thresholds button: Selecting this button displays the System Settings dialog box with only the fourth tab available. This option provides the capability to change the red, yellow and green color thresholds.

 Home button: When inside a hierarchy, instructs Raptor to return to the top level. This button is deactivated if the current window is not inside a hierarchy.

 Up button: When inside a hierarchy, instructs Raptor to go to the level immediate above the current window. This button will be deactivated if the current window is not inside a hierarchy.

 Goto Hierarchy button: Instructs Raptor to change the view to that of the hierarchy specified in the Goto Hierarchy drop down list box.

 Zoom Out button: Increases the amount of displayed view space by 5%, equivalent to an increase of 6 grids in width.

 Zoom In button: Decreases the amount of displayed view space by 5%, equivalent to a decrease of 6 grids in width.



Zoom to Fit button: Changes the zoom level to one in which the entire RBD fits on the screen.



Display Tree button: Opens the Tree dialog box that displays the hierarchical tree structure of the current RBD.



Help button: Opens the Raptor help file to a specific page concerning weak link analysis capabilities.

Failure Effects View Toolbar



Print Window button: Opens the Print Window dialog box, which allows printing of the window that is currently displayed.



Return to Workspace button: Returns focus from the Failure Effects View to the Workspace View.



Open Interactive Log button: Opens an interactive log. This button is deactivated if an interactive log is already opened.



Write to Log button: Writes the affects on the system of the currently failed components to the interactive log. This button is deactivated if an interactive log is not already opened.



Close Interactive Log button: Closes the interactive log. This button is deactivated if an interactive log is not already opened.



Decrease Phase button: Decreases the current phase by one. This button is deactivated if no phases are defined within the RBD.



Increase Phase button: Increases the current phase by one. This button is deactivated if no phases are defined within the RBD.



Refresh Phase button: Refreshes all components like new, but remains in the current phase. This button is deactivated if no phases are defined.



Refresh All button: Refreshes all components like new and returns to the first phase. This button is deactivated if no phases are defined.



Display Failure Effects button: Displays how the currently failed components affect the system.



Home button: When inside a hierarchy, instructs Raptor to return to the top level. This button is deactivated if the current window is not inside a hierarchy.



Up button: When inside a hierarchy, instructs Raptor to go to the level immediate above the current window. This button will be deactivated if the current window is not inside a hierarchy.



Goto Hierarchy button: Instructs Raptor to change the view to that of the hierarchy specified in the Goto Hierarchy drop down list box.



Zoom Out button: Increases the amount of grids displayed by 5% or an increase of 6 grids in width.



Zoom In button: Decreases the amount of grids being shown by 5% or an increase of 6 grids in width.



Zoom to Fit button: Changes the zoom level to one in which the entire RBD fits on the screen.



Display Tree button: Opens the Tree dialog box that displays the hierarchical tree structure of the current RBD.

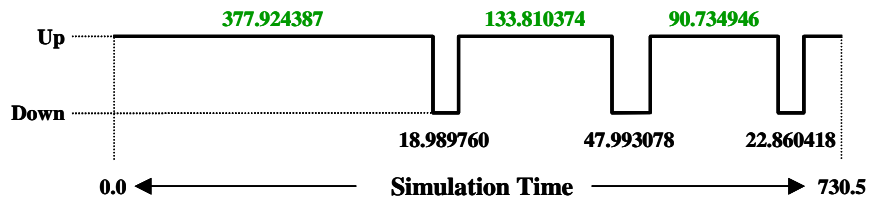


Help button: Opens the Raptor help file to a specific page concerning failure effects capabilities.

All system failure times: A file that contains the lengths of time between system downing events. The time that a system is down (red time) is not included within these values. This file is often used to aggregate sub-system RAM attributes into higher-level Raptor models.

```
377.924387
133.810374
90.734946
185.974263
11.006692
222.373303
408.772419
91.508274
392.889487
196.878277
254.110766
145.727042
231.634401
```

In this example, a simulation was conducted for ten trials of 730.5 hours. A review of the Detailed Event Log would reveal that the system failed three times during the first trial. The figure below represents the first three values of this file.

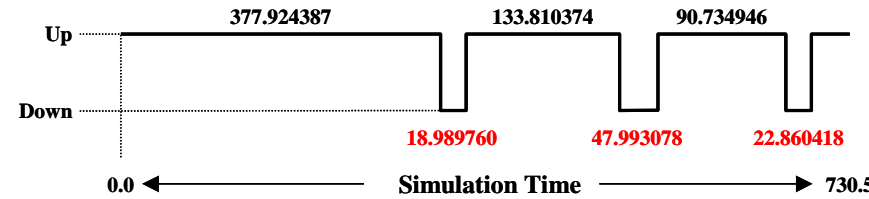


Results from multiple trials are included in a single list of values. Since there are 13 values in this file for the ten trials, the system experienced an average of 1.3 failures per trial. Censored operating times (i.e., ending simulation time is reached before a system downing event) are not included in this file.

All system down times: A file that contains the lengths of time a system was down. This file is often used to aggregate sub-system RAM attributes into higher-level Raptor models.

```
18.989760
47.993078
22.860418
2.470410
10.463316
37.179109
43.448759
39.429741
23.439194
23.748975
17.525759
22.381383
12.811535
```

In this example, the simulation was conducted for ten trials of 730.5 hours. A review of the Detailed Event Log would reveal that the system failed three times during the first trial. The figure below shows what the first three values of this file represent.



Results from multiple trials are included in a single list of values. Censored down times (i.e., simulation terminates while system is in a down status) are not included in this file.

Key parameters: A file that contains eight of the key system-level parameters for each trial. The file writes system availability, dependability, reliability, conditional reliability, mean time between downing events (MTBDE), mean down time (MDT), mean time between maintenance (MTBM), and mean maintenance time (MMT). See Appendix F for the precise definitions of these parameters.

Ao	Do	R	RCond	MTBDE	MDT	MTBM	MMT
0.939145220	0.939145220	0.000000000	0.000000000	686.045584	44.454416	85.755698	39.932971
0.990475825	0.990475825	0.000000000	0.000000000	723.542590	6.957410	80.393621	43.578551
0.926284876	0.926284876	0.000000000	0.000000000	169.162776	13.462224	48.332222	51.683069
1.000000000	1.000000000	1.000000000	1.000000000	730.500000	N/A	146.100000	53.200951
1.000000000	1.000000000	1.000000000	1.000000000	730.500000	N/A	66.409091	45.804751
0.932465037	0.932465037	0.000000000	0.000000000	227.055236	16.444764	75.685079	49.187865
0.925293153	0.925293153	0.000000000	0.000000000	337.963324	27.286676	45.061777	49.492804
0.882976377	0.882976377	0.000000000	0.000000000	161.253561	21.371439	71.668249	39.850200
0.933212629	0.933212629	0.000000000	0.000000000	340.855913	24.394087	85.213978	49.115646
0.918749961	0.918749961	0.000000000	0.000000000	167.786712	14.838288	61.013350	35.567465

If a simulation is terminated early after the first trial has been completed, the key parameter results for the clipped trial are discarded and not written to this file. If the simulation is terminated early before the first trial is completed, the cumulative values for the key parameters at the point of termination are written to this file.

The MTBDE parameter is the average operating time between failures that result in a system being in a down condition for each trial. If a trial is completed without a system-downing event, the ending simulation time is written to this file as MTBDE. Caution is warranted when the file contains ending simulation times as MTBDE's since these values can be shown to be poor estimators of a system's true MTBDE.

The MDT parameter is the average length of time a system was down or in a red condition. MDT includes logistical delays as well as actual repair times. If a trial is completed without a system-downing event, an estimate of a system's MDT cannot be determined and thus N/A is written for MDT.

The MTBM parameter is the average operating time between any maintenance occurring in a system including events that do not bring a system down. For example, failure of redundant components will cause maintenance activities to occur but it may not bring a system down. If a trial is completed without a maintenance event occurring, the ending simulation time is written to this file for MTBM. Caution is warranted when the file contains ending simulation times as MTBM's since these values can be poor estimators of a system's true MTBM.

The MMT parameter is the average maintenance time of a system, for either repair or preventive maintenance activities. Logistical delays are not included in the MMT values. If a maintenance activity is begun but not completed before the end of a trial, it is not included, as the clipped maintenance time is viewed as an incomplete and inaccurate data point. If a trial is completed without a maintenance event occurring, an estimate of the system's MMT cannot be determined and thus N/A is written for MMT.

Ending simulation times: A file that contains the time a simulation terminated for each trial. This file is predominately used when conducting failure-truncated simulations since the ending simulation time is a random variable.

```
396.914147
129.680388
712.809064
493.009966
123.861389
225.707158
132.429639
654.708392
187.794444
57.437747
```

This file is always comprised of a single column of values with n number of rows; where n is the number of trials.

If this file is used when conducting a time-truncated simulation, each value will equal the ending simulation time specified in the Simulation Options dialog box and thus the worth of this file would be questionable.

Availability plot data: A file that contains values that can be used to plot the availability of a system versus simulation time. These values are identical to those displayed when the display A_o plot simulation option is engaged (via the Advanced tab of the Simulation Options dialog box). This file can only be generated for time-truncated simulations.

SimulationTime	Ao	Green	Yellow	Red	Interval
50.000000	0.990000	0.214048	0.775952	0.010000	0.990000
100.000000	0.995000	0.107024	0.887976	0.005000	1.000000
150.000000	0.996667	0.264509	0.732158	0.003333	1.000000
200.000000	0.997500	0.448382	0.549118	0.002500	1.000000
250.000000	0.998000	0.558705	0.439295	0.002000	1.000000
300.000000	0.996667	0.531598	0.465069	0.003333	0.990000
350.000000	0.997143	0.455655	0.541487	0.002857	1.000000
400.000000	0.997500	0.427717	0.569783	0.002500	1.000000
450.000000	0.997778	0.491304	0.506474	0.002222	1.000000
500.000000	0.998000	0.542174	0.455826	0.002000	1.000000
550.000000	0.998182	0.583794	0.414388	0.001818	1.000000
600.000000	0.998333	0.618478	0.379855	0.001667	1.000000
650.000000	0.997692	0.644401	0.353292	0.002308	0.990000
700.000000	0.997857	0.598372	0.399485	0.002143	1.000000
730.500000	0.997947	0.573389	0.424558	0.002053	
*****End_of_Run_1*****					
50.000000	0.990000	0.859251	0.130749	0.010000	0.990000
100.000000	0.995000	0.429626	0.565374	0.005000	1.000000
150.000000	0.996667	0.286417	0.710250	0.003333	1.000000
200.000000	0.997500	0.302649	0.694851	0.002500	1.000000
250.000000	0.998000	0.319019	0.678981	0.002000	1.000000
300.000000	0.850493	0.274784	0.575710	0.149507	0.112959

The first column is the simulation time in increments of 50 units. The increment value is the A_0 plot sample rate and is set on the Advanced tab of the Simulation Options dialog box. The second column is the cumulative availability at the simulation time shown in the first column. The third, fourth and fifth columns are the percent of accumulated simulation time that the system was in fully functional, degraded and failed status, respectively. The last column is the availability during the specific interval.

If a simulation is not early terminated, the last line of the file is the ending simulation time and the availability at that instant. The interval availability is not provided on the last line of the file since there is no guarantee that the last interval is equal in length to the previous intervals. If the simulation time can be evenly divided by the A_0 plot sample rate, then the last line will be identical to the previous line (without the interval availability). If more than one trial is simulated, the results from each trial are included.

Capacity plot data: A file that contains values that can be used to plot the throughput of a system and the capacity of a system for all times during a simulation for which a change in capacity occurs. Thus, when the configuration of a system changes in a manner that affects the amount of flow handled by a system, a line of capacity information is written to this file.

SimulationTime	Flow	Capacity
0.000000	1500.000000	2050.000000
39.824560	0.000000	0.000000
42.824560	0.000000	0.000000
62.983326	0.000000	0.000000
65.983326	1500.000000	2050.000000
272.518642	1500.000000	1550.000000
272.768642	1500.000000	1550.000000
340.186423	1500.000000	1550.000000
340.436423	1500.000000	2050.000000
384.955400	1400.000000	1400.000000
385.205400	1400.000000	1400.000000
438.085577	1400.000000	1400.000000
438.335577	1500.000000	2050.000000
*****End_of_Run_1*****		
0.000000	1500.000000	2050.000000
397.978146	1500.000000	1550.000000
398.228146	1500.000000	1550.000000
426.557192	1500.000000	1550.000000
426.807192	1500.000000	2050.000000

Flow is the amount of throughput (e.g., manufactured items, messages, liquid flow, etc.) per unit time that is being processed by a system.

Capacity is the maximum amount of throughput that a system is capable of processing at any given time.

Final results. The final results report contains information summarized from across all trials conducted in a simulation. This file always contains a summary section, which is identical to the Summary tab of the Output Tables dialog box.

SUMMARY

	Minimum	Mean	Maximum	St.Dev	SEM
Total_Cost	20991314.84	21934833.58	23263835.52	861016.27	385058.18
Units_Produced	1071511.67	1075667.35	1079250.00	3570.77	1596.90
Mean_Flow	1488.210652	1493.982436	1498.958333	4.959399	2.217911
Mean_Capacity	2771.951837	3274.460623	3526.606291	292.125549	130.642517
Availability	0.997222222	0.998611111	1.000000000	0.001098013	0.000491046
MTBDE	179.500000	>443.600000	>720.000000	N/A	N/A
MDT(4-runs)	0.500000	0.500000	0.500000	0.000000	0.000000
Dependability	0.997222222	0.998611111	1.000000000	0.001098013	0.000491046
Reliability	0.000000000	0.200000000	1.000000000	0.447213595	0.200000000
Conditional_Reliability	0.000000000	0.200000000	1.000000000	0.447213595	0.200000000
System_Failures	0	2	4	1.581139	0.707107
Ending_Sim_Time	720.0	720.0	720.0	0.0	0.0

The logistics section is always included in the final results file. The information in this section is identical to that on the Logistics tab of the Output Tables dialog box.

LOGISTICS

	Minimum	Mean	Maximum	St.Dev	SEM
MTBM	51.285714	131.750476	239.833333	72.710236	32.517006
MTBMu	51.285714	131.750476	239.833333	72.710236	32.517006
MTBMs	N/A	N/A	N/A	N/A	N/A
MMT	36.260029	49.946663	62.804070	12.675939	5.668852
MMTu	36.260029	49.946663	62.804070	12.675939	5.668852
MMTs	N/A	N/A	N/A	N/A	N/A
Green_Percent	1.928992	31.811250	55.535492	24.052237	10.756488
Yellow_Percent	44.464508	68.049862	97.793230	23.945248	10.708640
Red_Percent	0.000000	0.138889	0.277778	0.109801	0.049105
Mission_Green_Percent	1.928992	31.811250	55.535492	24.052237	10.756488
Mission_Yellow_Percent	44.464508	68.049862	97.793230	23.945248	10.708640
Mission_Red_Percent	0.000000	0.138889	0.277778	0.109801	0.049105

The sparing section is also always included within the final results file. The spares utilization subsection provides data on how many spares were used by each block contained within an RBD. The AveUsed column displays the average number of spares used by a block across all trials. The MinUsed and MaxUsed columns display the minimum and maximum number of spares used by a block for a particular trial, respectively. The StDev column is the sample standard deviation of the spares used by a block across all trials.

SPARING

Spares_Utilization

blockName	SpareSource	MinUsed	AveUsed	MaxUsed	StDev
Attitude	Infinite	0	0	0	0
BatteriesNaS	Cells	0	0.6	1	0.55
BatteriesNiCd	Cells	0	0.4	1	0.55
BatteriesNiH	Cells	0	0.8	2	0.84
Bus_FR	Electronics	0	0.2	1	0.45
Bus_QR	Electronics	0	0.6	2	0.89
Bus_UR	Electronics	0	1	3	1.22
HighGain	Infinite	0	2	4	1.58
LowGain	Infinite	0	0	0	0
Nuclear_Reactor	Infinite	0	1.2	3	1.3
SolarP_500W	PhotoVPanels	0	0.2	1	0.45
SolarP_650W	PhotoVPanels	0	0	0	0
SolarP_900W	PhotoVPanels	0	0	0	0
Tether	Custom	0	0.2	1	0.45
Thermal	Custom	0	0	0	0

The sparing section is expanded to provide statistics on custom sparing and spare pools if these replacement strategies are used in a simulation. Values such as MinStock that are normally integers, are displayed as real numbers since the value is the average of the minimum stock levels observed from all trials simulated. The NumDelays column is the average number of delays observed. Delays are times when a block was put into a hold status because a spare was unavailable.

Custom_Sparing

blockName	InitNumSpares	NewSpares	MinStock	AveStock	MaxStock	EndStock	NumDelays
Tether	0	0.2	0	0	0	0	0.2
Thermal	1	0	1	1	1	1	0

SparePool_Sparing_Information

poolName	InitNumSpares	NewSpares	MinStock	AveStock	MaxStock	EndStock	NumDelays
PhotoVPanels	2	0	1.8	1.98	2	1.8	0
Cells	3	0.6	1.4	2.05	3	1.8	0.2
Electronics	2	0.6	0.8	1.17	2	0.8	0.6

The resources section of the final results file is included if a resource pool has been defined.

RESOURCES

Resource_Information

resourceName	InitNumRes	MinNumRes	AveNumRes	MaxNumRes	EndNumRes	NumDelays
Astronauts	5	0.4	3.84	5	4	1.6

The capacity section of the final results file is included if the capacity analysis option has been selected for a simulation.

CAPACITY

FlowGenerated	1500								
UnitsProduced	1075667.35								
FlowLoss	MinLostFlow	AveLostFlow	MaxLostFlow						
	0	6.02	1400						
SystemCapacity	MinFlow	AveFlow	MaxFlow	UsageRate	MinCapacity	AveCapacity	MaxCapacity	CapableRate	
	100	1493.98	1500	0.46	100	3274.46	4050	0.81	
NodeCapacity	MinFlow	AveFlow	MaxFlow	UsageRate	MinCapacity	AveCapacity	MaxCapacity	CapableRate	
500path	0	200.14	500	0.46	100	436.87	500	0.87	
650path	0	210.2	530	0.38	130	568.79	650	0.88	
900path	0	242.71	740	0.28	360	859.36	900	0.95	
end	100	1493.98	1500	N/A	N/A	N/A	N/A	N/A	
n1	100	1493.98	1500	N/A	N/A	N/A	N/A	N/A	
n2	100	1493.98	1500	N/A	N/A	N/A	N/A	N/A	
NukePath	0	659.28	920	0.54	0	1231.69	1500	0.82	
Power	100	1493.98	1500	0.25	4500	5913.69	6500	0.91	
TethPath	0	181.65	500	1	0	182	500	0.36	

The cost section of the final results file is included if the cost analysis option has been selected for a simulation. The first three subsections are costs associated with blocks. The first subsection delineates the various state costs for each block, the second subsection lists sparing costs, and the third subsection incorporates the state costs and sparing costs along with initial and disposal cost into a life cycle cost summary.

COST

Operating

Block	InHier	Good	Repair	Idle	RepHold	PM	PMHold	Standby	Done
Attitude	0	107850.00	0.00	52.00	0.00	0.00	0.00	0.00	0.00
BatteriesNaS	38	206245.58	16619.29	831.36	9.90	0.00	0.00	0.00	0.00
BatteriesNiCd	32	235907.79	24881.12	9821.81	59.03	0.00	0.00	0.00	0.00
BatteriesNiH	35	220514.81	28019.10	5882.88	2237.49	0.00	0.00	0.00	0.00
Bus_FR	38	79060.81	2449.21	605.03	86.40	0.00	0.00	0.00	0.00
Bus_QR	35	78755.29	14587.71	1513.58	112.00	0.00	0.00	0.00	0.00
Bus_UR	32	91217.68	32111.20	1366.20	500.89	0.00	0.00	0.00	0.00
HighGain	0	2024.99	13869.37	0.00	76.07	0.00	0.00	0.00	0.00
LowGain	0	1063.77	0.00	0.00	0.00	0.00	0.00	1014.49	0.00
Nuclear_Reactor	0	171823.39	119137.61	0.00	65281.57	0.00	0.00	0.00	0.00
SolarP_500W	32	141544.68	4068.04	2837.55	6.80	0.00	0.00	0.00	0.00
SolarP_650W	35	141759.52	0.00	2968.60	0.00	0.00	0.00	0.00	0.00
SolarP_900W	38	137497.05	0.00	1072.99	0.00	0.00	0.00	0.00	0.00
Tether	0	17710.01	1051.97	0.00	96.00	0.00	0.00	0.00	0.00
Thermal	0	10800.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Totals		1643775.4	256794.63	26951.99	68466.14	0.00	0.00	1014.49	0.00

Sparing

Block	InitSpares	NewSpares	EmerShip
Attitude	0.00	0.00	0.00
BatteriesNaS	0.00	0.00	0.00
BatteriesNiCd	0.00	0.00	0.00
BatteriesNiH	0.00	0.00	0.00
Bus_FR	0.00	0.00	0.00
Bus_QR	0.00	0.00	0.00
Bus_UR	0.00	0.00	0.00
HighGain	0.00	440.00	0.00
LowGain	0.00	0.00	0.00
Nuclear_Reactor	0.00	64800.00	0.00
SolarP_500W	0.00	0.00	0.00
SolarP_650W	0.00	0.00	0.00
SolarP_900W	0.00	0.00	0.00
Tether	0.00	256.00	400.00
Thermal	5185.00	0.00	0.00
Totals	5185.00	65496.00	400.00

Life_cycle_cost

Block	Purchase	Operating	Sparing	Disposal	Total
Attitude	6500.00	107902.00	0.00	0.00	114402.00
BatteriesNaS	9000.00	223706.13	0.00	0.00	232706.13
BatteriesNiCd	4000.00	270669.74	0.00	0.00	274669.74
BatteriesNiH	8000.00	256654.28	0.00	0.00	264654.28
Bus_FR	11000.00	82201.45	0.00	0.00	93201.45
Bus_QR	12000.00	94968.58	0.00	0.00	106968.58
Bus_UR	10000.00	125195.97	0.00	0.00	135195.97
HighGain	1500.00	15970.43	440.00	0.00	17910.43
LowGain	900.00	2078.26	0.00	0.00	2978.26
Nuclear_Reactor	83650.00	356242.57	64800.00	0.00	504692.57
SolarP_500W	3125.00	148457.07	0.00	0.00	151582.07
SolarP_650W	3125.00	144728.13	0.00	0.00	147853.13
SolarP_900W	3125.00	138570.04	0.00	0.00	141695.04
Tether	2250.00	18857.98	656.00	0.00	21763.98
Thermal	85000.00	10800.00	5185.00	0.00	100985.00
Block_Totals	243175.00	1997002.62	71081.00	0.00	2311258.62

The next subsection summarizes the costs associated with the events in the RBD.

Event	InHier	Initial	Success	Failure	Total
Colorado_Springs	0	1.00	22.40	7.60	31.00
Inspection	0	1.00	57.00	3.00	61.00
Event_Totals		2.00	79.40	10.60	92.00

The next subsection summarizes the costs by subsystem.

Hierarchy	HierNum	Purchase	Operating	Sparing	Disposal	Events	Total
500path	32	17125.00	544322.79	0.00	0.00	0.00	561447.79
650path	35	23125.00	496350.98	0.00	0.00	0.00	519475.98
900path	38	23125.00	444477.62	0.00	0.00	0.00	467602.62

The next two subsections summarize the pool costs associated with any spare or resource pools that were defined for an RBD.

SparePools	Initial	NewSpares	EmerShip	Total
PhotoVPanels	6250.00	0.00	0.00	6250.00
Cells	9225.00	2460.00	135000.00	146685.00
Electronics	17570.00	5271.00	351300.00	374141.00
SparePool_Totals	33045.00	7731.00	486300.00	527076.00

Resources	Initial	PerUseCost	PerTimeCost	Total
Astronauts	18750000.00	513.00	208561.31	18959074.31
Resource_Totals	18750000.00	513.00	208561.31	18959074.31

The last cost subsection is a summary of all costs incurred from each of the previous subsections. System down cost is included within this report only if the system red time cost is greater than zero. The system lost flow cost is included within this report only if a simulation has been conducted that implemented the cost and capacity analysis options.

Cost_Summary

Blocks	2311258.62
Events	92.00
SparePools	527076.00
Resources	18959074.31
SystemDownCost	133000.00
SystemLostFlowCost	4332.65
<hr/> Grand_Total	21934833.58

The weak link analysis section of the final results file is included if the weak link analysis option has been selected for a simulation. This section reports on the availability, dependability, and reliability of the nodes, blocks, events and hierarchies. For each parameter, it gives the min, the mean, the max, the standard dev and the SEM. This is the same data that can be found in the Output Tables dialog box. It also gives information on the percent of time that each element spent in the various states. A sample of the weak link analysis section is shown below.

WEAK_LINK_ANALYSIS

NODE_ANALYSIS

Reliability_Data

Node	RMin	RMean	RMax	RStdDev	RSEM
500path	0.00000000	0.20000000	1.00000000	0.447213595	0.20000000
650path	0.00000000	0.20000000	1.00000000	0.447213595	0.20000000
900path	0.00000000	0.40000000	1.00000000	0.547722558	0.244948974
Comm	0.00000000	0.20000000	1.00000000	0.447213595	0.20000000
n1	1.00000000	1.00000000	1.00000000	0.00000000	0.00000000
n2	1.00000000	1.00000000	1.00000000	0.00000000	0.00000000
NukePath	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000
Power	1.00000000	1.00000000	1.00000000	0.00000000	0.00000000
TethPath	0.00000000	0.00000000	0.00000000	0.00000000	0.00000000

State_Data

Node	%Green	%Yellow	%Red	%Blue	%Brown	%Orange
500path	87.373256	0.000000	12.626744	0.000000	0.000000	0.000000
650path	87.505878	0.000000	12.494122	0.000000	0.000000	0.000000
900path	95.484066	0.000000	4.515934	0.000000	0.000000	0.000000
Comm	70.311997	29.549114	0.138889	0.000000	0.000000	0.000000
n1	99.861111	0.000000	0.000000	0.000000	0.138889	0.000000
n2	100.000000	0.000000	0.000000	0.000000	0.000000	0.000000
NukePath	82.112821	0.000000	17.887179	0.000000	0.000000	0.000000
Power	29.522961	70.477039	0.000000	0.000000	0.000000	0.000000
TethPath	36.400057	0.000000	63.599943	0.000000	0.000000	0.000000

BLOCK_ANALYSIS

Availability_Data

Block	AoMin	AoMean	AoMax	AoStdDev	AoSEM
Attitude	0.997222222	0.998611111	1.000000000	0.001098013	0.000491046
BatteriesNaS	0.890170906	0.954840656	1.000000000	0.046349647	0.020728192
BatteriesNiCd	0.615723503	0.873732562	1.000000000	0.153226566	0.068525004
BatteriesNiH	0.694379650	0.875058778	1.000000000	0.113949994	0.050959987
Bus_FR	0.890170906	0.954840656	1.000000000	0.046349647	0.020728192
Bus_QR	0.694379650	0.875058778	1.000000000	0.113949994	0.050959987
Bus_UR	0.615723503	0.873732562	1.000000000	0.153226566	0.068525004
HighGain	0.406866515	0.703119970	1.000000000	0.280969998	0.125653603
LowGain	1.000000000	1.000000000	1.000000000	0.000000000	0.000000000
Nuclear_Reactor	0.690660440	0.867794877	1.000000000	0.147307977	0.065878130
SolarP_500W	0.615723503	0.873732562	1.000000000	0.153226566	0.068525004
SolarP_650W	0.694379650	0.875058778	1.000000000	0.113949994	0.050959987
SolarP_900W	0.890170906	0.954840656	1.000000000	0.046349647	0.020728192
Tether	0.919448438	0.983889688	1.000000000	0.036023753	0.016110312
Thermal	1.000000000	1.000000000	1.000000000	0.000000000	0.000000000

Caution is warranted when this section is used to interpret results in conjunction with failure-truncated simulations. Since a failure-truncated simulation's ending time is a random variable, and usually not consistent across multiple trials, the utility of this section is questionable. For example, a trial that lasts 500 hours with a node's percent green time of 50%, and a trial that lasts 50 hours with a node's percent green time of 100%, will yield an average percent green time of 75%. Although the first trial lasted ten times longer, the non-time-weighted averages calculated in this report disregards this information.

End of run: A report that contains summary, sparing, resource, capacity, cost and node analysis results for each trial of a simulation. The format of this report is the same as the final results report. This report can be large if numerous trials are conducted. This report provides useful insight into the variability between trials while

the final results report provides summaries across trials. The first two sections of this report are shown below.

Run_1**

SUMMARY

Availability	0.794316121
MTBDE	3.917911
MDT	1.014522
Reliability	0.000000000
ConditionalReliability	0.000000000
EndingSimTime	365.0
NumberOfSystemFailures	74
NumberOfComponentFailures	347

LOGISTICS

MTBM	0.835520
MTBMu	0.835520
MTBMs	N/A
MMT	0.380829
MMTu	0.380829
MMTs	N/A
GreenPercent	62.957009
YellowPercent	16.474603
RedPercent	20.568388

	Minimum	Mean	Maximum	St.Dev
TBDE	0.299331	3.862710	15.414227	2.973433
DT	0.013038	1.014522	3.571355	0.829151
TBM	0.000161	1.047954	5.514521	0.936055
RT	0.011279	0.380829	2.595407	0.333877

SPARING

Spare_Utilization

blockName	Failures	SpareSource	Consumed
1553card	12	Custom	12
AC_power	12	Infinite	12
apple_filler	42	Infinite	42
batteries	3	Custom	3
boxer	10	Infinite	10
cherry_filler	44	Infinite	44
computer	3	Infinite	3
crust_former	32	Infinite	32
crust_former	30	Infinite	30
crust_former	24	Infinite	24
emerg_switch	7	Infinite	7
folder_motor	13	motor_parts	13
main_motor	37	motor_parts	37
mixer	3	mixer_parts	3
mixer	3	mixer_parts	3
mixer	1	mixer_parts	1
oven	11	Infinite	11
rhubarb_filler	60	Infinite	60

Detailed event log: A report that contains every significant event that transpires during each simulation trial. All block and event failures, block repairs, logistics aspects, node conditions and phasing information are provided in this file. The file is often used to facilitate the verification and validation of a particular RBD or software module.

Column 1 of this log contains the simulation time that an event line was written. Column 2 contains the type of element that the line is referring to and column 3 contains the name of that element. Column 4 contains the new state that the referenced element is in and column 5 contains a pertinent comment concerning the element. The information in column 5 varies depending on the action.

0.000000	System	Begin_Run_1		
0.000000	Block	Engine	Good	InitializeStress=1.000000
0.000000	Block	Tractor_Ele	Good	InitializeStress=1.000000
0.000000	Block	Fuel_Tank1	Good	InitializeStress=1.000000
0.000000	Block	Fuel_Tank2	Good	InitializeStress=1.000000
0.000000	Block	Fuel_Pump	Good	InitializeStress=1.000000
0.000000	Block	Cooling	Good	InitializeStress=1.000000
0.000000	Block	Transfer_Drive	Good	InitializeStress=1.000000
0.000000	Block	Brakes_1	Good	InitializeStress=1.000000
0.000000	Block	Brakes_2	Good	InitializeStress=1.000000
0.000000	Block	Gearbox	Good	InitializeStress=1.000000
0.000000	Block	Tractor_Structure	Good	InitializeStress=1.000000
0.000000	Block	LeftFront_Tire	Good	InitializeStress=1.000000
0.000000	Block	RightFront_Tire	Good	InitializeStress=1.000000
0.000000	Block	Trailer_Structure	Good	InitializeStress=1.000000
0.000000	Block	Trailer_Ele	Good	InitializeStress=1.000000
0.000000	Block	Tire_Outside_1	Good	InitializeStress=1.000000
0.000000	Block	Tire_Outside_2	Good	InitializeStress=1.000000
0.000000	Block	Tire_Outside_3	Good	InitializeStress=1.000000
0.000000	Block	Tire_Outside_4	Good	InitializeStress=1.000000
0.000000	Block	Tire_Outside_5	Good	InitializeStress=1.000000
0.000000	Block	Tire_Outside_6	Good	InitializeStress=1.000000
0.000000	Block	Tire_Outside_7	Good	InitializeStress=1.000000
0.000000	Block	Tire_Outside_8	Good	InitializeStress=1.000000
0.000000	Block	Tire_Inside_1	Good	InitializeStress=1.000000
0.000000	Block	Tire_Inside_2	Good	InitializeStress=1.000000
0.000000	Block	Tire_Inside_3	Good	InitializeStress=1.000000
0.000000	Block	Tire_Inside_4	Good	InitializeStress=1.000000
908.938960	Block	Tire_Outside_6	Failed	Operating_time=908.938960
908.938960	Block	Tire_Outside_6	Repairing	No_resources_used
908.938960	Node	n8	Degraded	2-2/3_df32_ur33_ur43
908.938960	Block	Tire_Outside_6	Repaired	MaintTime=0.000000
908.938960	Block	Tire_Outside_6	Good	Stress=1.000000
908.938960	Node	n8	Good	3-2/3_ur32_ur33_ur43
1531.094330	Block	Tire_Outside_8	Failed	Operating_time=1531.094330
1531.094330	Block	Tire_Outside_8	Sparing	No_spare_in_stock-Outside_Tires
1531.094330	Block	Tire_Outside_8	Done	No_spares
1531.094330	Node	n9	Degraded	2-2/3_ur36_df37_ur45
1531.094330	System	State_change	Yellow	_d23
4046.782428	Block	Brakes_2	Failed	Operating_time=4046.782428
4046.782428	Block	Brakes_2	Done	No_repair

Interim cost log: A report that contains the same cost information included in the end of run and final results reports at each interval defined by a frequency value. This report is often used to facilitate the post processing of the cost information into a time-valued money analysis. Cost information is written to this report for each trial that the frequency value is less than a simulation's ending simulation time.

The report is only available to be written when a cost analysis simulation is conducted. If this report is used in conjunction with delayed statistics gathering, only costs that occur after the delay point are written to this report.

Appendix D

Raptor Colors and Statuses

Block Colors













Image	Status			Description
	Condition	Class	State	
	Up	Functioning	Running	Block is actively running, accumulating time towards its next failure.
	Up	Functioning	Running	Block is actively running in an under-stressed condition, accumulating time at a lesser than normal rate towards its next failure. (Also Workspace color)
	Up	Functioning	Running	Block is actively running in an over-stressed condition, accumulating time at a greater than normal rate towards its next failure.
	Up	Reserved	Standby	Block is in cold standby (i.e., time is not accumulated toward failure).
	Up	Reserved	Standby	Block is in warm standby (i.e., time is accumulated at a rate less than typical operating failure rate).
	Up	Reserved	Standby	Block is in hot standby (i.e., time is accumulated at a rate greater than or equal to typical operating failure rate).
	Down	Constrained	Idle	Block is not operating because it is dependent on another element or the system which is down (i.e., time is not accumulated toward failure).
	Down	Scheduled	PM	Block is actively undergoing scheduled preventive maintenance.
	Down	Scheduled	PM-hold	Block is waiting for a spare or resource to arrive for preventive maintenance to be performed.
	Down	Failed	Hold	Block has failed but is not actively repairing since it is waiting for a spare or resource to arrive or a logistic delay to expire.
	Down	Failed	Repairing	Block is actively repairing (i.e., hands-on maintenance).
	Down	Failed	Done	Block has permanently failed and will not be repaired again within the current simulation trial.



Image	Status		Description
	Phase	State	
	Linked	Running	Block is linked and actively running, accumulating time towards its next failure.
	Linked	Running	Block is linked and actively running in an under-stressed condition, accumulating time at a lesser than normal rate towards its next failure. (Also Workspace color)
	Linked	Running	Block is linked and actively running in an over-stressed condition, accumulating time at a greater than normal rate towards its next failure.
	Linked	Standby	Block is linked and in cold standby (i.e., time is not accumulated toward failure).
	Linked	Standby	Block is linked and in warm standby (i.e., time is accumulated at a rate less than typical operating failure rate).
	Linked	Standby	Block is linked and in hot standby (i.e., time is accumulated at a rate greater than or equal to typical operating failure rate).
	Linked	Idle	Block is linked and not operating because it is dependent on another element or the system which is down (i.e., time is not accumulated toward failure).
	Linked	PM	Block is linked and actively undergoing scheduled preventive maintenance.
	Linked	PM-hold	Block is linked and waiting for a spare or resource to arrive for preventive maintenance to be performed.
	Linked	Hold	Block is linked and has failed but is not actively repairing since it is waiting for a spare or resource to arrive or a logistic delay to expire.
	Linked	Repairing	Block is linked and actively repairing (i.e., hands-on maintenance).
	Linked	Done	Block is linked and has permanently failed and will not be repaired again within the current simulation trial.





















Image	Status		Description
	Phase	State	
	Cut	Running	Block is cut and actively running, accumulating time towards its next failure.
	Cut	Running	Block is cut and actively running in an under-stressed condition, accumulating time at a lesser than normal rate towards its next failure. (Also Workspace color)
	Cut	Running	Block is cut and actively running in an over-stressed condition, accumulating time at a greater than normal rate towards its next failure.
	Cut	Standby	Block is cut and in cold standby (i.e., time is not accumulated toward failure).
	Cut	Standby	Block is cut and in warm standby (i.e., time is accumulated at a rate less than typical operating failure rate).
	Cut	Standby	Block is cut and in hot standby (i.e., time is accumulated at a rate greater than or equal to typical operating failure rate).
	Cut	Idle	Block is cut and not operating because it is dependent on another element or the system which is down (i.e., time is not accumulated toward failure).
	Cut	PM	Block is cut and actively undergoing scheduled preventive maintenance.
	Cut	PM-hold	Block is cut and waiting for a spare or resource to arrive for preventive maintenance to be performed.
	Cut	Hold	Block is cut and has failed but is not actively repairing since it is waiting for a spare or resource to arrive or a logistic delay to expire.
	Cut	Repairing	Block is cut and actively repairing (i.e., hands-on maintenance).
	Cut	Done	Block is cut and has permanently failed and will not be repaired again within the current simulation trial.

Image	Description
	The block's weak link parameter is greater than or equal to the green color threshold.
	The block's weak link parameter is greater than or equal to the yellow color threshold, yet less than the green color threshold.
	The block's weak link parameter is less than the yellow color threshold.
	The block is running due to the lack of induced effects.
	The block is in opportunistic preventive maintenance due to induced effects.
	The block is idle due to induced effects.
	The block has failed due to induced effects.
	The standard (non-phased) block color in the Workspace View.
	A phased block will display as a hatched block in the Workspace View.

Node Colors


















Image	Status			Description
	Condition	Class	State	
	Up	Functioning	Good	The number of inbound functional links (j) is equal to the total number of inbound links (n).
	Up	Functioning	Degraded	The number of inbound functional links (j) is greater than or equal to the number required (k), but less than the total number of inbound links (n).
	Up	Reserved	Standby	Node is in standby (i.e., enough inbound links are in standby to allow switchover if called upon).
	Down	Constrained	Idle	Node is not operating because upstream blocks are dependent on another element or the system, which is down.
	Down	Scheduled	PM	Node is down since the number of inbound functional links (j) is less than the number required (k) due to upstream elements undergoing preventive maintenance.
	Down	Failed	Bad	Node is down since the number of inbound functional links (j) is less than the number required (k) due to failed or cut elements.
	Up	Functioning	Good	The number of inbound functional links (j) is equal to the total number of inbound links (n).
	Up	Functioning	Degraded	The number of inbound functional links (j) is greater than or equal to the number required (k), but less than the total number of inbound links (n).
	Up	Reserved	Standby	Node is in standby (i.e., enough inbound links are in standby to allow switchover if called upon).
	Down	Constrained	Idle	Node is not operating because upstream blocks are dependent on another element or the system, which is down.
	Down	Scheduled	PM	Node is down since the number of inbound functional links (j) is less than the number required (k) due to upstream elements undergoing preventive maintenance.
	Down	Failed	Bad	Node is down since the number of inbound functional links (j) is less than the number required (k) due to failed or cut elements.

Image	Description
	The node's weak link parameter is greater than or equal to the green color threshold.
	The node's weak link parameter is greater than or equal to the yellow color threshold, yet less than the green color threshold.
	The node's weak link parameter is less than the yellow color threshold.
	The standard (non-phased) node color in the Workspace View.
	A phased node will display as a hatched block in the Workspace View.

Event Colors




Image	Status		Description
	Condition	State	
	Up	Success	Event has fired successfully (random draw was less than or equal to the event's p-value).
	Up	Armed	A phased event that has not yet been fired.
	Down	Failed	Event has fired unsuccessfully (random draw was greater than the event's p-value).












Image	Status		Description
	Phase	State	
	Linked	Success	Event is linked and the localized path is by definition good. Event cannot fire but maintains its previous state.
	Linked	Armed	Event is linked and the localized path is by definition good. Event cannot fire but maintains its previous state.
	Linked	Failed	Event is linked and the localized path is by definition good. Event cannot fire but maintains its previous state.
	Cut	Success	Event is cut and the localized path is by definition bad. Event cannot fire but maintains its previous state.
	Cut	Armed	Event is cut and the localized path is by definition bad. Event cannot fire but maintains its previous state.
	Cut	Failed	Event is cut and the localized path is by definition bad. Event cannot fire but maintains its previous state.

Image	Description
	<p>The event's weak link parameter is greater than or equal to the green color threshold.</p>
	<p>The event's weak link parameter is greater than or equal to the yellow color threshold, yet less than the green color threshold.</p>
	<p>The event's weak link parameter is less than the yellow color threshold.</p>
	<p>The standard (non-phased) event color in the Workspace View.</p>
	<p>A phased event will display as a hatched block in the Workspace View.</p>

Hierarchy Colors







Image	Status			Description
	Condition	Class	State	
	Up	Functioning	Good	Hierarchy is operating with no loss of redundancy.
	Up	Functioning	Degraded	Hierarchy is operating with some loss of redundancy.
	Up	Reserved	Standby	Hierarchy is in standby.
	Down	Constrained	Idle	Hierarchy is not operating because certain internal elements are idle.
	Down	Scheduled	PM	Hierarchy is not operating because certain internal elements are undergoing preventive maintenance.
	Down	Failed	Bad	Hierarchy is not operating because certain internal elements are failed or cut.







Image	Status		Description
	Phase	State	
	Linked	Good	Hierarchy is linked and operating with no loss of redundancy.
	Linked	Degraded	Hierarchy is linked and operating with some loss of redundancy.
	Linked	Standby	Hierarchy is linked and in standby.
	Linked	Idle	Hierarchy is linked and not operating because certain internal elements are idle.
	Linked	PM	Hierarchy is linked and not operating because certain internal elements are undergoing preventive maintenance.
	Linked	Bad	Hierarchy is linked and not operating because certain internal elements are failed or cut.






























Image	Status		Description
	Phase	State	
	Cut	Good	Hierarchy is cut and operating with no loss of redundancy.
	Cut	Degraded	Hierarchy is cut and operating with some loss of redundancy.
	Cut	Standby	Hierarchy is cut and in standby.
	Cut	Idle	Hierarchy is cut and not operating because certain internal elements are idle.
	Cut	PM	Hierarchy is cut and not operating because certain internal elements are undergoing preventive maintenance.
	Cut	Bad	Hierarchy is cut and not operating because certain internal elements are failed or cut.

Image	Description
	<p>The hierarchy's weak link parameter is greater than or equal to the green color threshold.</p>
	<p>The hierarchy's weak link parameter is greater than or equal to the yellow color threshold, yet less than the green color threshold.</p>
	<p>The hierarchy's weak link parameter is less than the yellow color threshold.</p>
	<p>The standard (non-phased) hierarchy color in the Workspace View.</p>
	<p>A phased hierarchy will display as a hatched block in the Workspace View.</p>







Hierarchy DSI Colors

Image	Status			Description
	Condition	Class	State	
	Up	Functioning	Good	The hierarchy displaying this DSI and all internal hierarchies are up, in a non-degraded state.
	Up	Functioning	Degraded	The hierarchy displaying this DSI and all internal hierarchies are up. However, at least one is in a degraded state.
	Up	Reserved	Standby	The hierarchy displaying this DSI and all internal hierarchies are up. However, at least one is in a standby state.
	Down	Constrained	Idle	Either the hierarchy displaying this DSI or at least one of the internal hierarchies is idle.
	Down	Scheduled	PM	Either the hierarchy displaying this DSI or at least one of the internal hierarchies is down due to preventive maintenance.
	Down	Failed	Bad	Either the hierarchy displaying this DSI or at least one of the internal hierarchies is down.
	N/A	N/A	N/A	The hierarchy displaying this DSI is cut. See the Hierarchy Colors section for more information. This is also the color of the DSI in the Workspace View.
	N/A	N/A	N/A	The hierarchy displaying this DSI is linked. See the Hierarchy Colors section for more information.




Marker Colors

Image	Status			Description
	Condition	Class	State	
	N/A	N/A	N/A	Start marker. This marker is black in all views.
	N/A	N/A	N/A	End marker. This marker is black in all views.
	N/A	N/A	N/A	Hierarchy In marker (All views).
	N/A	N/A	N/A	Hierarchy Out marker (Workspace View).
	Up	Functioning	Good	Hierarchy is operating with no loss of redundancy.
	Up	Functioning	Degraded	Hierarchy is operating with some loss of redundancy.
	Up	Reserved	Standby	Hierarchy is in standby.
	Down	Constrained	Idle	Hierarchy is not operating because certain internal elements are idle.
	Down	Scheduled	PM	Hierarchy is not operating because certain internal elements are undergoing preventive maintenance.
	Down	Failed	Bad	Hierarchy is not operating because certain internal elements are failed or cut.

Link Colors

Image	Status			Description
	Condition	Class	State	
	Up	Functioning	Good	Path is functioning properly. When capacity analysis is implemented, the link is carrying no amount of flow. (Also link Workspace color)
	Up	Reserved	Standby	At least one element in the string is in a cold standby state. All others are functioning or in standby.
	Down	Constrained	Idle	Link is down because elements within its string are down due to dependency.
	Down	Constrained	Switching	Link is down because standby elements within its string are switching from a primary to an alternate.
	Down	Scheduled	PM	Link is down because elements within its string are down due to preventive maintenance.
	Down	Failed	Bad	Link is down because elements within its string have failed or have been cut.

System Colors

Image	Status		Description
	Condition	Class	
	Up	Fully Functional	The system is completely functional; no elements are down.
	Up	Degraded	The system is operating with some loss of redundancy.
	Down	Failed	The system has an element or combination of elements that resulted in a system downing event.

System Status Indicators







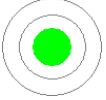
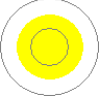


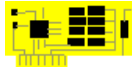
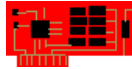















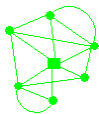
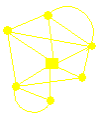
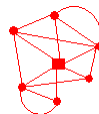












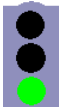

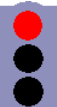









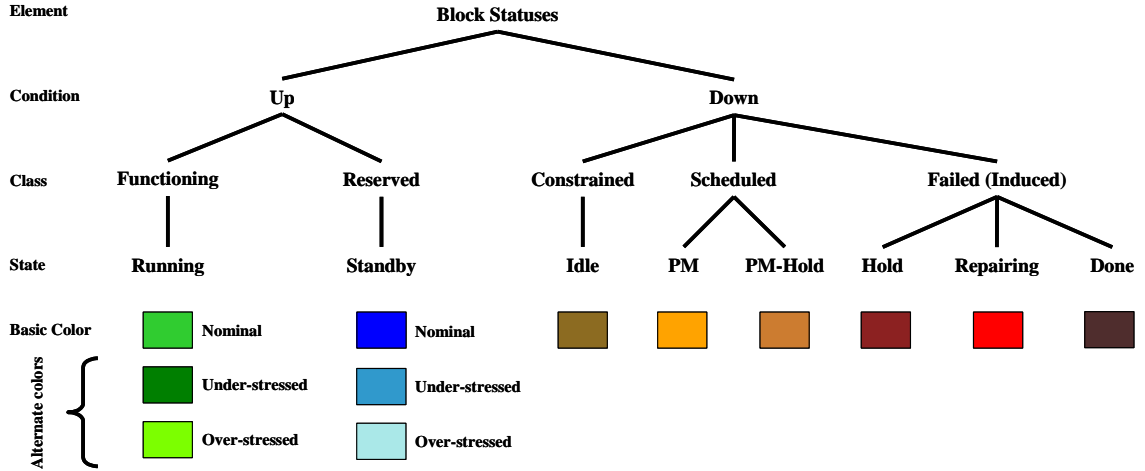
Image	Fully Functional	Degraded	Failed
Aircraft			
Artillery			
Bull's Eye			
Circuit Card			
Computer			
Equalizer			
Helicopter			
Meter			
Missile			

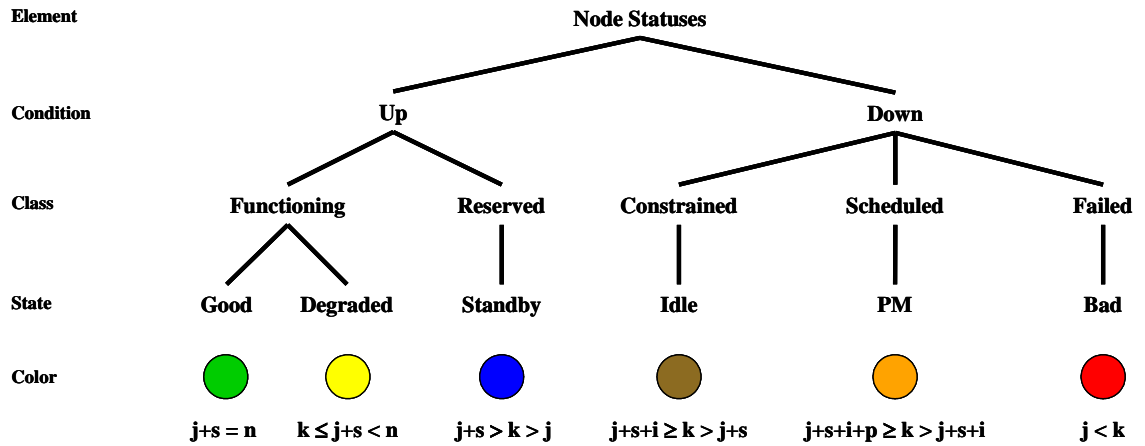
Image	Fully Functional	Degraded	Failed
Network			
Radio			
Radome			
Satellite			
Ship			
Stoplight			
Tank			
Train			
Truck			

Status Diagrams

Block



Node



k = the number of good paths required
 n = the number of total paths
 j = the number of good paths currently available
 s = the number of standby paths
 i = the number of idle paths
 p = the number of PM paths

Event 

Element

Event Statuses

Condition

Up

Down

State

Success

Armed

Failed

Color



$R \leq p_{\text{value}}$

$R = 0$

$R > p_{\text{value}}$

Where R is uniform real random number between zero and one

Hierarchy 

Element

Hierarchy Statuses

Condition

Up

Down

Class

Functioning

Reserved

Constrained

Scheduled

Failed

State

Good

Degraded

Standby

Idle

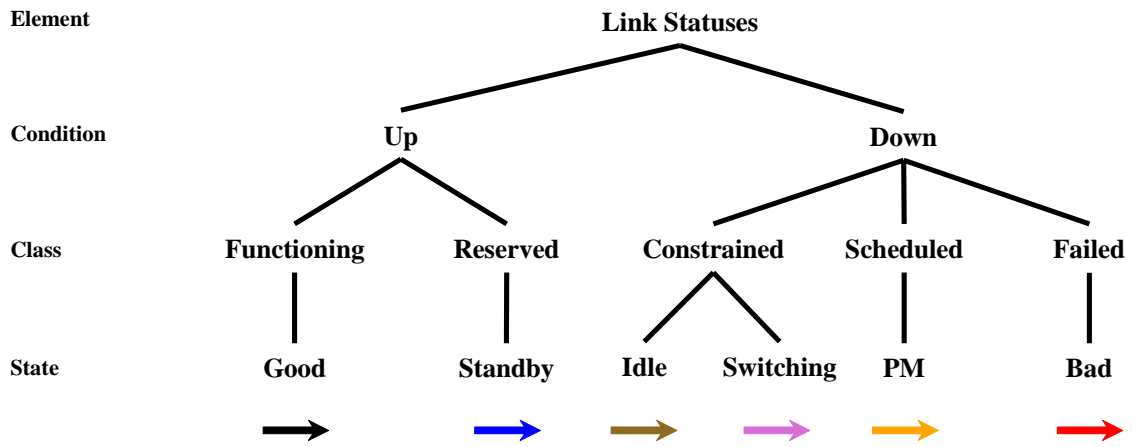
PM

Bad

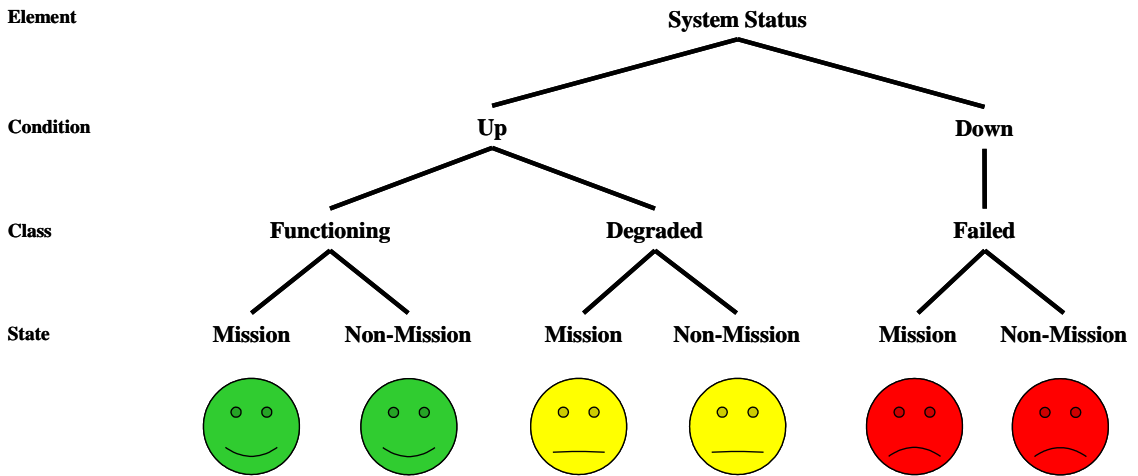
Color



Link



System



Elemental

1. RBDs must have at least one block or event to begin a simulation.
2. RBDs can be saved without links, but cannot be simulated.
3. Blocks can only possess one inbound link.
4. Blocks can only possess one outbound link.
5. Events can only possess one inbound link.
6. Events can only possess one outbound link.
7. Hierarchies can only possess one inbound link.
8. Hierarchies can only possess one outbound link.
9. A link cannot connect into the start marker or an in marker.
10. Only one link can connect out of the start marker or an in marker.
11. Only one link can connect into the end marker or an out marker.
12. A link cannot connect out of the end marker or an out marker.
13. Any number of links can connect into a connect node.
14. Any number of links can connect out of a connect node.

Distributional

1. When the Extreme Value, Laplace or Normal distributions are used, it is possible to obtain a negative number for a failure or repair time. When this occurs, a warning message is displayed stating that a negative random number has been modified to 0.000001.
2. If a failure occurs and repairs immediately, it is still consider a failure.

Logistics

1. If a block with a pre-logistic delay time fails and needs to order a spare, a spare will be ordered at the same time its logistics delay time begins. A block will request a spare once its pre-logistics delay time has been exhausted.
2. If a block requires two or more resources and only one is available, a block will acquire the available resource and continue to wait for additional resources. If another block fails or enters preventive maintenance during this waiting period and only requires one resource, it will go into a hold status and therefore cannot use the resource being held by the first block. Resources are assigned in a "first asked first gets" priority scheme.
3. The block's request for resources does not go into the queue until the block has obtained its spare.
4. If a block has used resources for its repair, and then it has a post-logistic delay time, the resources will be released at the completion of the hands-on repair.
5. Logistics delay time does not apply to preventive maintenance actions, only unscheduled maintenance actions.
6. If a spare pool (or custom sparing) is set to use stock level ordering, and the initial stock pile is less than the stock level ordering level, spares will not be ordered. Stock level ordering places its order when and only when the number of spares is reduced to exactly the stock level order level.

Cost

1. The initial cost of a block is accrued at time zero of a simulation.
2. A block using either the custom or pooled sparing replacement strategy will be charged at the beginning of a simulation for each spare of their respective stockpiles.
3. If the infinite sparing replacement strategy is selected for a block, a charge for the initial stockpile is not assessed.
4. A charge for a new spare occurs when ordered from a block using either the infinite, custom or pooled sparing replacement strategy.
5. Cost for spares that arrive at regular intervals is accrued when the spares arrive.
6. Cost for spares that arrive via stock level ordering or as an emergency order, are accrued at the time the order was placed.

7. A block being linked or cut has no affect on the costs accumulated by that block.
8. A disposal or rebate cost is assessed when a block has permanently failed. The user can specify that the disposal or rebate costs for all blocks are assessed at the end of a simulation whether a particular block has permanently failed or not.
9. Partial or pro-rated costs are assessed at the end of a simulation for all possible block statuses. For example, if a block is partially through a repair when a simulation trial terminated, only the partial repair cost will be assessed.
10. Any fixed costs associated with repair or preventive maintenance are assessed at the point at which maintenance begins.

Phasing

1. If a phase change is due to occur exactly at the end of a simulation, a simulation will terminate prior to that phase change.
2. The k -value for a node can be set to zero for any phased node. This situation creates a maintenance drain.
3. The random numbers associated with the phase lengths are drawn from the ancillary number stream.
4. For the purposes of dependence, cut and linked have the affect of failed and operating, respectively. If a block is dependent upon a block that is cut, it will enter its dependency state. Conversely, if a block is dependent upon a block that is linked, and the linked block fails and enters repair, the dependent block will not enter its dependency state.

Growth and Decay

1. If decay is implemented without a limit specified for the lower bound, a limit of 0.000001 is imposed.
2. If growth is implemented without a limit specified for the upper bound, a limit of 999,999,999.999999 is imposed.

Standby

1. If all blocks within a standby structure have the same priority, then the order the blocks were placed within the Workspace will determine which blocks are primaries and which are alternates at the start of a simulation trial.
2. Automatic switchover is always attempted before a block's manual switchover time is implemented.
3. Automatic switchover with a probability less than one will acquire a random number from the ancillary stream to determine if the switchover is successful.
4. If automatic switchover fails, and the path does not have manual switchover, the switch is assumed permanently failed and the path will be failed (red) for the remainder of the trial, regardless of the condition of the block. If automatic switchover fails and the path does have manual switching, then once a manual switch has occurred the automatic switch will be reset.
5. The stress on a block in warm or hot standby is the standby stress only. Phase stress and growth decay stress do not affect the time to failure.

Capacity

1. If a node has several outbound paths with the same priority and the "Use Any Available Path" checkbox is deselected, the flow amount is distributed evenly among the available paths.
2. If the "Use Any Available Path" option is checked for directing outbound flow through a node, then flow is not divided evenly among outbound paths but sent down paths that can handle the flow. If capacity information is not relevant through a portion of an RBD, this option is preferred since the algorithm that distributes flow is slower than the use any available path algorithm.

Random Start-ups

1. Blocks that start down will obtain spares or resources prior to beginning their repairs. If blocks share resources or spares, some blocks may start a simulation in hold status if sufficient logistics are not available.

2. If a block starts a simulation trial with a specified number of hours completed that exceed a block's failure distribution maximum, then that block will fail at simulation time 0.000001.
3. If a block starts a simulation trial with a specified number of hours completed that is near the extreme right edge of a block's failure distribution, then that block will fail at simulation time 0.000001 if an appropriate time to failure cannot be determined. One hundred random number attempts will be made to determine an appropriate failure time before the default failure time of 0.000001 is implemented.
4. If a block starts a simulation trial in one of three random conditions (i.e., random up, random down, or completely random), two random numbers are immediately used from a block's designated random number stream to determine its starting conditions.

Preventive Maintenance

1. Triggered preventive maintenance that occurs while a block is repairing is assumed to be accomplished while the repair is occurring with no additional time added.
2. The time to preventive maintenance is unaffected by any "pi" factors such as operating stress, standby stress, or block growth/decay.
3. If a block's preventive maintenance comes due during non-mission time, but the block cannot start the maintenance due to lack of a spare or resource, the block goes to preventive maintenance hold status. This initiates the preventive maintenance cycle, which then takes priority and will be completed. Even if the block is set to have its preventive maintenance deferred during a mission phase and a mission phase begins, the block will not interrupt the PM cycle and will continue in PM hold and PM until the maintenance is complete.
4. Triggers draw their random numbers from the ancillary stream.

System-level Parameters

Operational Availability (A_o)

$$A_o = \frac{Uptime}{Total\ Time} = \frac{Fully\ Functioning + Degraded}{Total\ Time} = \frac{Green + Yellow}{Green + Yellow + Red}$$

Mean Time Between Downing Events (MTBDE)

$$MTBDE = \frac{Uptime}{\#of\ Downing\ Events} = \frac{Green + Yellow}{\#of\ System\ Red\ Events}$$

Mean Down Time (MDT)

$$MDT = \frac{Downtime}{\#of\ Downing\ Events} = \frac{Red}{\#of\ System\ Red\ Events}$$

Dependability (D_o)

$$D_o = \frac{Mission\ Uptime}{Mission\ Total\ Time} = \frac{Mission\ Green + Mission\ Yellow}{Mission\ Green + Mission\ Yellow + Mission\ Red}$$

Reliability (R)

$$R = \frac{\#of\ missions\ completed\ without\ a\ system\ failure}{\#of\ missions\ attempted}$$

Conditional Reliability (R_C)

$$R_C = \frac{\text{\# of missions completed successfully when the system was up at the start of the mission}}{\text{\# of missions attempted when the system was up at the start of the mission}}$$

Mean Time Between Maintenance (MTBM)

$$MTBM = \frac{\text{Uptime}}{\text{\# of Maintenance Events}} = \frac{\text{Green + Yellow}}{\text{\# of Component Downing Events}}$$

Mean Time Between Maintenance Unscheduled (MTBMu)

$$MTBMu = \frac{\text{Uptime}}{\text{\# of Failure Maintenance Events}} = \frac{\text{Green + Yellow}}{\text{\# of Component Failures}}$$

Mean Time Between Maintenance Scheduled (MTBMs)

$$MTBMs = \frac{\text{Uptime}}{\text{\# of Preventive Maintenance Events}} = \frac{\text{Green + Yellow}}{\text{\# of Component Preventive Maintenance Events}}$$

Mean Maintenance Time (MMT)

$$MMT = \frac{\text{Total Maintenance Time}}{\text{\# of Maintenance Events}}$$

Mean Maintenance Time Unscheduled (MMTu)

$$MMTu = \frac{\text{Total Unscheduled Maintenance Time}}{\text{\# of Unscheduled Maintenance Events}}$$

Mean Maintenance Time Scheduled (MMTs)

$$MMTs = \frac{\text{Total Preventive Maintenance Time}}{\text{\#of Preventive Maintenance Events}}$$

Note: Each of these parameters is calculated at the end of each run. For simulations with multiple runs, the lowest observed value for each of these parameters is denoted as the minimum. The highest value observed across the multiple trials is denoted as the maximum. The standard deviation of the values of each parameter is calculated using the equation for sample standard deviation, where n denotes the number of runs.

$$\text{sample standard deviation} = s = \sqrt{\frac{\sum_{i=1}^n (X_i - \bar{X})^2}{n-1}}$$

From the standard deviation, the standard error of the mean is calculated.

$$SEM = \frac{s}{\sqrt{n}}$$

Weak Link Analysis Parameters

Node Availability (A_n)

$$A_n = \frac{\text{Uptime}}{\text{Uptime} + \text{Downtime}} = \frac{\text{Good} + \text{Degraded} + \text{Cold}}{(\text{Good} + \text{Degraded} + \text{Cold}) + (\text{Idle} + \text{Preventive Maintenance} + \text{Bad})}$$
$$= \frac{\text{Green} + \text{Yellow} + \text{Blue}}{(\text{Green} + \text{Yellow} + \text{Blue}) + (\text{Brown} + \text{Orange} + \text{Red})}$$

Node Dependability (D_n)

$$D_n = \frac{\textit{Principal Uptime}}{\textit{Principal Uptime} + \textit{Principal Downtime}} = \frac{\textit{Good} + \textit{Degraded}}{(\textit{Good} + \textit{Degraded}) + \textit{Down}}$$
$$= \frac{\textit{Green} + \textit{Yellow}}{(\textit{Green} + \textit{Yellow}) + \textit{Red}}$$

Node Reliability (R_n)

$$R_n = \frac{\textit{\#of missions completed without a node failure}}{\textit{\#of missions attempted}}$$

Block Availability (A_b)

$$A_b = \frac{\textit{Uptime}}{\textit{Uptime} + \textit{Downtime}} = \frac{\textit{Good} + \textit{Cold}}{(\textit{Good} + \textit{Cold}) + (\textit{Idle} + \textit{Preventive Maintenance} + \textit{Bad} + \textit{Done})}$$
$$= \frac{\textit{Green} + \textit{Blue}}{(\textit{Green} + \textit{Blue}) + (\textit{Brown} + \textit{Orange} + \textit{Red} + \textit{IndianRed})}$$

Block Dependability (D_b)

$$D_b = \frac{\textit{Principal Uptime}}{\textit{Principal Uptime} + \textit{Principal Downtime}} = \frac{\textit{Good}}{\textit{Good} + (\textit{Down} + \textit{Done})}$$
$$= \frac{\textit{Green}}{\textit{Green} + (\textit{Red} + \textit{IndianRed})}$$

Block Reliability (R_b)

$$R_b = \frac{\text{\# of missions completed without the block going down}}{\text{\# of missions attempted}}$$

Event Availability (A_e)

$$\begin{aligned} A_e &= \frac{\text{Uptime}}{\text{Uptime} + \text{Downtime}} = \frac{\text{Success} + \text{Armed}}{(\text{Success} + \text{Armed}) + \text{Failure}} \\ &= \frac{\text{Green} + \text{LightBlue}}{(\text{Green} + \text{LightBlue}) + \text{Red}} \end{aligned}$$

Event Dependability (D_e)

$$\begin{aligned} D_e &= \frac{\text{Principal Uptime}}{\text{Principal Uptime} + \text{Principal Downtime}} = \frac{\text{Success}}{\text{Success} + \text{Failure}} \\ &= \frac{\text{Green}}{\text{Green} + \text{Red}} \end{aligned}$$

Event Reliability (R_e)

$$R_e = \frac{\text{\# of missions completed without an event failure}}{\text{\# of missions attempted}}$$

Hierarchy Availability (A_h)

$$A_h = \frac{\textit{Uptime}}{\textit{Uptime} + \textit{Downtime}} = \frac{\textit{Good} + \textit{Degraded} + \textit{Cold}}{(\textit{Good} + \textit{Degraded} + \textit{Cold}) + (\textit{Idle} + \textit{Preventive Maintenance} + \textit{Bad})}$$
$$= \frac{\textit{Green} + \textit{Yellow} + \textit{Blue}}{(\textit{Green} + \textit{Yellow} + \textit{Blue}) + (\textit{Brown} + \textit{Orange} + \textit{Red})}$$

Hierarchy Dependability (D_h)

$$D_h = \frac{\textit{Principal Uptime}}{\textit{Principal Uptime} + \textit{Principal Downtime}} = \frac{\textit{Good} + \textit{Degraded}}{(\textit{Good} + \textit{Degraded}) + \textit{Down}}$$
$$= \frac{\textit{Green} + \textit{Yellow}}{(\textit{Green} + \textit{Yellow}) + \textit{Red}}$$

Hierarchy Reliability (R_h)

$$R_h = \frac{\textit{\# of missions completed without a hierarchy failure}}{\textit{\# of missions attempted}}$$

Growth and Decay Formulas

Linear

$$\theta_{new} = [(\textit{Slope} \times \textit{\# of failures}) + 1] \times \theta_{base}$$

Geometric

$$\theta_{new} = (\text{Base}^{\#of\ failures}) \times \theta_{base}$$

Asymptotic

$$\theta_{new} = \text{Limit} + (\theta_{base} - \text{Limit}) e^{(-rate \times \#of\ failures)}$$

Capacity Formulas

$$\text{Flow} = \frac{\text{substance}}{\text{time}} \triangleright \frac{\text{gallons}}{\text{minute}} \triangleright \frac{\text{cars}}{\text{hour}} \triangleright \frac{\text{chlorine atoms}}{\text{seconds}} \triangleright \frac{\text{messages}}{\text{nano - seconds}}$$

$$\text{Capacity} = \max(\text{Flow}) \triangleright \max\left(\frac{\text{gallons}}{\text{minute}}\right) \triangleright \max\left(\frac{\text{cars}}{\text{hour}}\right) \triangleright \max\left(\frac{\text{messages}}{\text{second}}\right)$$

$$\text{Usage rate} = \frac{\text{Average Flow}}{\text{Average Capacity}}$$

$$\text{Capability} = \frac{\text{Average Capacity}}{\text{Maximum Capacity}}$$

