extendsim®

RELIABILITY

TUTORIAL & REFERENCE

License, Software Copyright, Trademark, and Other Information

The software described in this manual is furnished under a separate license and warranty agreement. The software may be used or copied only in accordance with the terms of that agreement. Please note the following:

> ExtendSim blocks and components (including but not limited to icons, dialogs, and block code) are copyright © by Imagine That Inc. and/or its Licensors. ExtendSim blocks and components contain proprietary and/or trademark information. If you build blocks, and you use all or any portion of the blocks from the ExtendSim libraries in your blocks, or you include those ExtendSim blocks (or any of the code from those blocks) in your libraries, your right to sell, give away, or otherwise distribute your blocks and libraries is limited. In that case, you may only sell, give, or distribute such a block or library if the recipient has a valid license for the ExtendSim product from which you have derived your block(s) or block code. For more information, contact Imagine That Inc.

Imagine That!, the Imagine That logo, ExtendSim, Extend, and ModL are either registered trademarks or trademarks of Imagine That Incorporated in the United States and/or other countries. Mac OS is a registered trademark of Apple Computer, Inc. Microsoft is a registered trademark and Windows is a trademark of Microsoft Corporation. GarageGames, Inc. is the copyright owner of the Torque Game Engine (TGE), and the copyright for Stat::Fit® is owned by Geer Mountain Software. All other product names used in this manual are the trademarks of their respective owners. TGE and Stat::Fit are licensed to Imagine That, Inc. for distribution with ExtendSim. All other ExtendSim products and portions of products are copyright by Imagine That Inc. All right, title and interest, including, without limitation, all copyrights in the Software shall at all times remain the property of Imagine That Inc. or its Licensors.

Acknowledgments

Extend was created in 1987 by Bob Diamond; it was re-branded as ExtendSim in 2007.

Chief architects for ExtendSim 10: Steve Lamperti and Bob Diamond

Simulation Engineers: Anthony Nastasi, Cecile Pieper, Peter Tag, and Dave Krahl

Graphics, documentation, and editing: Kathi Hansen, Carla Sackett, and Pat Diamond

Imagine That Inc • 6830 Via Del Oro, Suite 230 • San Jose, CA 95119 USA
408.365.0305 • fax 408.629.1251 • info@extendsim.com
www.ExtendSim.com

Table of Contents

Reliability
Tutorial & Reference

Introduction

Welcome!

Thank you for using ExtendSim, the power tool for simulation modeling! We hope you enjoy using ExtendSim and that you find this document helpful.

About this document

This document explores the ExtendSim Reliability module and shows how to use it both as a standalone reliability block diagram (RBD) tool and as an RBD tool integrated with the ExtendSim process simulation software (PSS) capabilities found in ExtendSim DE and ExtendSim Pro.

Since RBD is a discrete event methodology, reliability models require an Executive block (Item library) for scheduling events. An RBD Executive is not yet available for the ExtendSim CP product.

Who should read this document

The availability of critical resources to perform work is often a key factor in limiting system performance. Yet identifying which resource availabilities are most important, and to what extent the timing and duration of their unavailability impacts the system, can be a complex problem to solve.

Since the ExtendSim Reliability module can be used as either a standalone RBD tool or in conjunction with ExtendSim simulation capabilities, this document will be helpful for:

- Anyone looking to explore the availability of an entire system and/or its individual resources.
- ExtendSim modelers using the Item and/or Rate libraries of ExtendSim DE and Pro to simulate systems. Since the Reliability module seamlessly integrates with those libraries, you can explore the impact of resource availability on key process metrics such as throughput, production costs, repair costs, utilization, inventory, service levels, and so forth.

See also “Advantages of integrating RBD with PSS” on page 4 and “When to use the Reliability module” on page 5.

This document assumes you already know how to launch ExtendSim and build a model. If not, see either the Discrete Event or Discrete Rate Quick Start Guide (QSG). In addition, since

building RBD's is so integrated with the ExtendSim database, we suggest you read the Extend-Sim Database Tutorial and Reference.

Chapters in this reference

1) Introduction to reliability (this chapter of the document):
2) Basic information: exploring a reliability model
3) Tutorial 1: building a stand-alone RBD of a bicycle
4) Tutorial 2: adding a discrete event process to the RBD
5) Tutorial 3: adding an RBD to a discrete rate model
6) Reference: a comprehensive catalog of Reliability features and capabilities

Introduction to reliability block diagramming

Reliability block diagramming (RBD) is a methodology that graphically and statistically describes a system's resource availability over time as well as the impact it has on the system as a whole.

Availability and down events

In the context of reliability, the term *availability* is defined as the percentage of time a resource is available to perform work. Factors that would cause a resource to become unavailable are categorized as either scheduled or unscheduled down events:

- Scheduled downs represent planned down times for things like maintenance and off-shifting.
- Unscheduled downs represent unexpected downtimes due to failures.

RBD

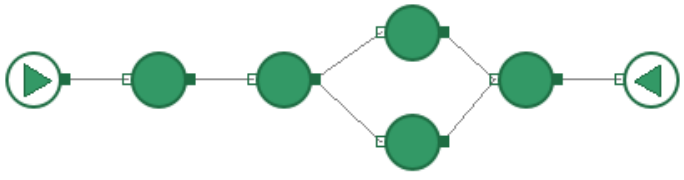
An RBD captures complex availability behavior for individual resources and for entire systems. It graphically and statistically describes when scheduled and unscheduled downs occur for individual resources and what impact that has on the availability of entire system.

Graphical description

An RBD is a *directed acyclic graph* of nodes and edges where interior *nodes* (also known as components) represent resources that fluctuate between their up and down states while *edges* describe how the nodes are related to each other.

A system represented by an RBD can be as simple as a one-component network or as complex as a web of many nodes that have been placed both in series and in parallel.

- Components placed in *series* indicate that all the resources need to be in an up state in order for the system to be available to perform work.
- Alternatively, components placed in *parallel* indicate redundancy. As shown in the RBD below, only one of the two parallel resources needs to be up in order for the system to be available for work.



Statistical description

Since each node in the graph contains information about its probability of being in an available state, an RBD is also a *statistical representation* of the resources in the system. In other words, each node contains one or more statistically defined *event cycles* (also known as *failure modes*) that describe the resource's availability behavior over time.

Name[2]	Purpose[3]	Distribution Type[4]	Parameter 1 [5]	Parameter 2 [6]
Failure Brake	TTD	Weibull	80.00000	50.00000
Failure Drivetrain Chain	TTD	Weibull	90.00000	10.00000
Failure Drivetrain Crank	TTD	Weibull	365.00000	100.00000
Failure Drivetrain Derailleur	TTD	Weibull	90.00000	20.00000

Advantages of RBD

Using RBD to describe both the individual resource and the overall system availability has the following advantages:

1) **Visual Logic**. RBD's are really good at visually describing the relationships between resources. Instead of filling out tables by hand or writing code or logical statements, relationship logic can be captured in a visual graph with minimal effort.
2) **Powerful**. There is a high degree of complex reliability logic that can be captured in an RBD without having to write code.
3) **Validation**. Complex reliability logic can be visually validated rather than having to sift through tables or decipher code.

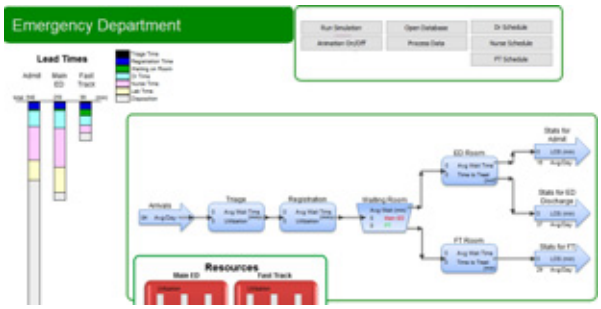
Two types of discrete event tools

Discrete event tools model systems where the simulation clock moves forward in discrete chunks of time. These tools can be divided into two groups:

1) Process Simulation Software (PSS) tools, which model the dynamic behavior of the system, simulating the process steps by which systems transform inputs into outputs.
2) Dedicated RBD tools, which graphically and statistically describe when individual resources and entire systems of resources become available and unavailable over time.

Even though the RBD and PSS tools both employ discrete event modeling techniques to manage the simulation clock, the two technologies were historically developed to solve different kinds of problems and evolved independently of each other. Consequently, very little cross-pollination of ideas and capabilities transpired between them. As a result PSS and RBD tools typically benefit from a different set of strengths and suffer from a different set of weaknesses.

Process simulation software



Discrete event process simulation software (e.g. the ExtendSim Item and Rate libraries) are ideally suited for capturing the detailed behavior of the complicated "process-based" types of systems that often occur in manufacturing, military operations, and logistics, such as the emergency department shown at right.

For example, to determine the ideal conveyor speed in a bottling plant you need to account for the impact the changing speed would have on other critical resources. One such resource would be the labor needed to restock consumables like stickers, caps, and bottles along the line. As conveyor speeds increase, how much more labor is needed to keep up? To answer that question, the PSS can literally map out the path walked by the workers and define the time needed to traverse that path using a statistical distribution that changes during the course of a shift based on when labor starts to get tired. Although you wouldn't want to "over-model" the system, these kinds of nitty-gritty details can be essential to understanding how the system currently works and planning how it could or should work.

RBD tools

RBD tools provide a sophisticated level of reliability analysis. As discussed on page 3, RBD tools can visually capture and validate complex reliability logic and the relationships between a system's resources.

Going back to the bottling conveyor example, each machine on a packing line could have more than 500 dependent and independent failure modes that are capable of bringing the packing line down. RBD tools can quickly and easily model those types of complex failure behaviors.

Advantages of integrating RBD with PSS

Dedicated RBD tools are very good at capturing availability over time. However they lack the ability to simulate the details of essential system behaviors (such as conveyor speed or the contention for labor resources) and to model how those behaviors impact the system as a whole.

On the other hand, process simulation tools excel at modeling overall system behavior but struggle to capture the sometimes extremely complex nature of availability. They would typically be ill equipped to model the complexity of hundreds of dependent and independent failure modes for the bottling plant.

The Reliability module bridges those gaps by providing a Reliability Block Diagramming (RBD) tool that can also be integrated with the powerful process simulation capabilities of ExtendSim.

Integrating the ExtendSim RBD capability with its process modeling modules provides a number of compelling advantages:

1) More accurate component wearing. The process model can be used to define when and what types of machine wearing is occurring on the resources in the RBD. This can lead to a more accurate assessment of when wear-based failures occur.

2) Detailed repair modeling. When a resource in the RBD fails, ExtendSim modeling capabilities can be used to break out the repair process in as much detail as needed:
   - Are the tools needed to make the repair currently available or are they currently allocated to other tasks?
   - What is the current level of spare parts inventory and supplier lead time?
   - Are the labor resources qualified to make the repair currently available or are they performing other work?
   - Should we preempt key resources and redirect them to this higher priority job?
   - If the resources required to make this repair are currently unavailable, should we outsource the job?
3) The RBD's ability to model when resources and/or entire systems are down can be used to impact the movement of material through the simulated process. This allows the modeler to explore the relationship between resource availability and system performance metrics like throughput, production costs, repair costs, utilization, inventory, service levels, etc.

When to use the Reliability module

The Reliability module can be used as either a stand-alone RBD tool or in conjunction with ExtendSim process simulation capabilities.

☞ In either instance, using reliability requires reliability-specific data to properly populate the RBD's. For example, you'll need to determine which distributions to use to categorize the resources failures, repairs, and so forth.

As a stand-alone tool

Use the Reliability module on its own any time you want to explore the availability of individual resources and the overall availability of the system.

In conjunction with PSS

An RBD is primarily an *availability* tool. In those cases where resource availability critically impacts model results, the Reliability library will play an important role. Specifically, consider integrating Reliability into your PSS models when the system being modeled possesses the following characteristics:

- Resource availability significantly impacts the model's key metrics.
- The availability status of a particular resource impacts the availability of other resources and/or the availability of the system as a whole.
- The system possesses key resources that go on/off shift, are prone to failure, and/or are taken off line for scheduled maintenance.

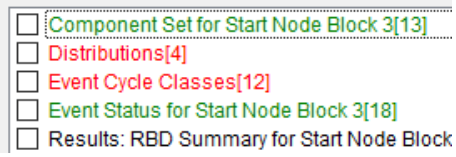
In systems such as these, the Reliability module will provide meaningful benefits to your PSS model building efforts and your post run analysis.

Framework

The following components comprise the Reliability module.

1) ExtendSim **RBD databases**:

- Contain all the information needed to characterize the model's RBD's as you build them.
- Can be used to control RBD behaviors during the run.
- Store information that describes the current state of the RBD's.
- Document the results from model runs.
- Allows RBD's to easily scale up.



2) Nodes for creating the graphical structure of the RBD:

- **Start Node**. The first block in an RBD, it is in charge of scheduling down or up events for all components in its RBD and is responsible for determining the state of each individual node and the overall state of the RBD.

- **Component.** The interior of an RBD contains one or more Component blocks that represent components (in RBD talk) or resources (in PSS language), interchangeably. Failure, repair, shift, and maintenance events are used to change component availability status over time.

- **End Node**. The terminating node in an RBD. During the run it reports the state of the RBD on its output connectors.

☞ Each RBD must begin with a single Start Node which is then connected to one or more Components placed in series and/or in parallel to each other. Each RBD must terminate with a single End Node.

3) Other blocks:

- **Distribution Builder**. Creates reliability-specific distributions that specify the time between downs (TBD), time to downs (TTD), and time to ups (TTU) and stores them in an RBD database. These distributions specify how much time an event cycle spends in its up or down state.

- **Event Builder**. Creates event cycles and stores them in an RBD database. Event cycles use the reliability distributions to control how components (resources) move between the up and down states over time.

Event cycles can also be thought of as failure modes. However, event cycles indicate failures, repairs, shifts, and maintenance events, so failure modes is a more limiting term.

Reliability module features

☞ *Failure modes* is a common term used in RBD. To be more inclusive, the term *event cycles* will be used throughout this document to indicate failures, repairs, shifts, and maintenance events.

The ExtendSim Reliability module has many features that make it a powerful tool for modeling resource and system availability and determining how to better manage the resources in terms of redundancy and maintenance scheduling. In conjunction with a PSS tool, it is essential for analyzing the role resource availability plays in system performance.

This module includes:

- Graphical diagram/database builder. The Reliability module provides a graphical interface for building RBD's. As the user builds the diagram, the databases needed to support that diagram are automatically built.
- RBD/PSS interface. The Reliability module is fully integrated and supports communication between the RBD and PSS sections of the model.
- Distribution scaling. Since reliability distributions are stored in a generic format in the RBD databases, importing distributions from an external source (such as Excel) is supported. This means the number of distributions needed to support an RBD scales up easily.
- Multiple event cycles per node. To accurately model resource and system availability, any number of different event cycles such as failures, repairs, shifts, and maintenance events can be associated with a particular RBD node.
- Event cycle scaling. Since event cycles are stored in database tables, importing event cycles that have been defined in an external data source is supported. This means the number of event cycles needed to support an RBD scale up easily.
- Control logic. Optionally, the ExtendSim IDE can be used to write code to control all aspects of a node's behavior. For example, you can write code to control the speed at which a component progresses towards its next down based on any number of factors including the status of other related components, the state of the process model, seasonal policy changes, projected demand, and so forth.
- RBD databases. As mentioned earlier, these auto-built databases contain all the information needed to characterize the structure and current state of the RBD's in a model and document all the results from model runs. Additionally, these databases can be used to control RBD behavior during the run.

⚠ By default the ExtendSim Pro product allows a maximum of 100 event cycles/failure modes per model. As needed, additional event cycles can be purchased and added to ExtendSim Pro.

Where to get more information

The ExtendSim documentation, example models, and the video files and documents on the ExtendSim.com website provide comprehensive help.

Quick Start Guides

The purpose of a Quick Start Guide (QSG) is to get new users quickly familiar with a specific ExtendSim simulation methodology and aware of the ExtendSim features and capabilities. There are three Quick Start Guides—Continuous Process Modeling, Discrete Event Simulation, and Discrete Rate Modeling. Depending on the product purchased, one or more of these will be installed as eBooks in the Documents/ExtendSim/Documentation folder.

☞ It is recommended that you read one of these Quick Start Guides before continuing with this document. Each of the QSG's has similar structure and information, so it is not necessary to read all of them.

Tutorial & Reference documents

In addition to the Quick Start Guides, there are three Tutorial & Reference documents that are included as eBooks in the Documents/ExtendSim/Documentation folder:

- *ExtendSim Database*. This internal relational database provides model developers with a systematic way to manage information for the model and makes models scalable.

- *Advanced Resource Management*. ARM is a sophisticated architecture for systematically dealing with multiple types of resources and a model's complex resource requirements.
- *Reliability* (this document). Graphically capture and validate complex availability behavior. Determine when scheduled and unscheduled downs occur for individual resources and what impact that has on the availability of the system as a whole.

User Reference

The ExtendSim User Reference has a lot of information you will find helpful when building, using, and presenting models.

How To chapters cover general modeling and simulation topics

- Using libraries and blocks
- Performing analysis
- Enhancing presentations
- Creating a user interface
- Using equation-based blocks
- And much more

Appendices list menu commands and the ExtendSim libraries and blocks

Every menu command is explained; the main libraries are described block by block.

Technical Reference

You probably won't build your own ExtendSim blocks, but it's very common to use functions and logical statements in an Equation block (Value library) in a model. The Technical Reference lists over 1,000 functions and has information about using include files and other programming tools.

☞ The eBooks ship with the appropriate ExtendSim product. To access these documents, see the Documents/ExtendSim/Documentation/folder or launch the books from the Getting Started model that opens when ExtendSim launches. The User Reference and Technical Reference are also available if you select the Help menu when using ExtendSim.

Example models and videos show you how

ExtendSim includes numerous tutorial models as well as videos and example models that explain concepts discussed in the documentation.

Reliability
Tutorial & Reference

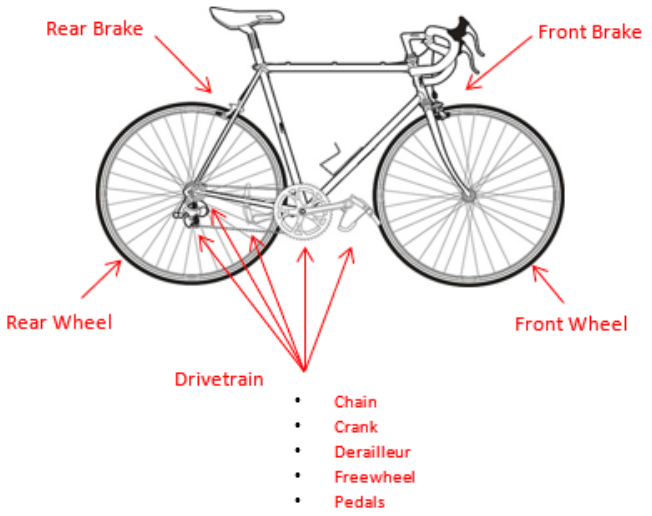
Basics: Exploring an RBD

Overview

This chapter discusses RBD terminology and describes RBD components while exploring an ExtendSim reliability model.

The bicycle

Assume a bike-messenger wants to model the reliability of her bicycle.



Bicycle components

As pictured above, the bicycle has the following components, each with their own failure rates:

1) Front wheel
2) Rear wheel

3) Front brake
4) Rear brake
5) A drivetrain composed of five subcomponents:
   - Chain
   - Crank
   - Dérailleur
   - Freewheel
   - Pedals

Assumptions

The bike-messenger:

- Has the tools and skills to make any repairs and has all the necessary spare parts on site.
- Services the drivetrain annually. During this event, every subcomponent of the drivetrain, except the crank, is serviced.
- Services the front and rear wheels bi-annually.
- Can use the bicycle if either the front or rear brakes are working, but not if neither of them are working.

☞ For this model, the wearing of the bicycle components is not modeled explicitly. Instead, wearing is assumed to be occurring while the bicycle is in an up state as the simulation is running.

Why simulate the bicycle using RBD?

The purpose of running an RBD is to determine if the system (in this case the bicycle) is up and available for work or if and how often one or more downed components are conspiring to take the entire system down for some period of time. The bicycle is in an "up" state if one or more paths through the network are up; otherwise it is down.

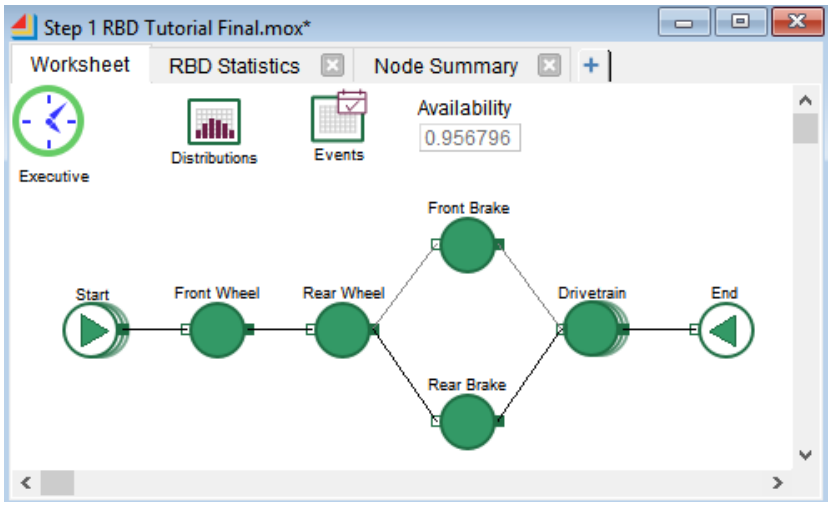
Exploring an RBD model of the bicycle

This section uses a model of the reliability of the bicycle to discuss an RBD and its components.

Open the example model

▸ Launch ExtendSim

▸ Open the model **Step 1 RBD Tutorial**; it is located at Documents/ExtendSim/Examples/Tutorials/Reliability.



The model worksheet

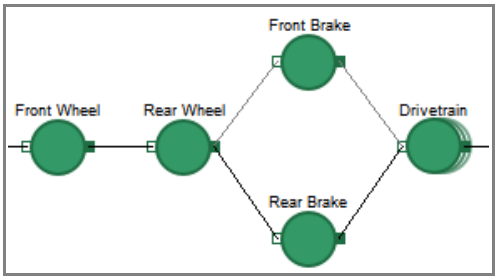
The model worksheet has several icons, called blocks, seven of which are connected together.

- The set of connected blocks, starting with the *Start Node* and terminating at the *End* node, is the RBD. The blocks between the Start and the End are called *Components*.
- The other blocks (*Executive*, *Distribution Builder*, and *Event Builder*) perform specific modeling tasks in reliability models.

These blocks are discussed more starting on page 13.

Paths and redundancy

Notice that in this RBD the Front and Rear Wheels are in series but the Front and Rear Brakes are parallel to each other. The brakes provide two paths through the model—either the upper path that goes through the front brake or the lower path that goes through the rear brake.



Placing the front and rear brakes in parallel to each other provides *redundancy*—if one of the brakes is down, there is still an alternate path through the RBD so the entire system isn't down.

Conversely, if even one of the other components (e.g. either the front or the rear wheel) are down, the system is down and the bicycle won't work.

Run the model

- Run the model by clicking the Run Simulation button or using the Run > Run Simulation command.

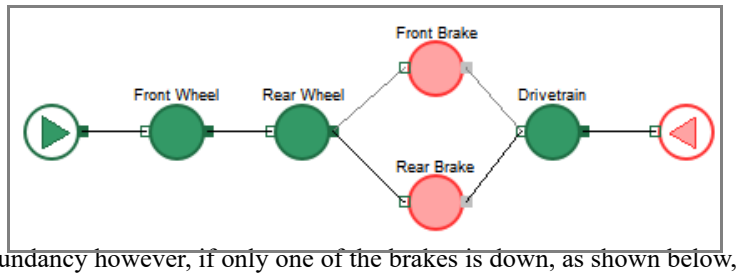
To explore variability, the model is set to run 5 times (run numbers 0-4) each time you give the Run command, and each run is set for 1,095 simulated days (3 years).

☞ If the animation runs too quickly for you to see what is happening, use the toolbar's Animation Slider to reduce the speed. Or click the Pause button (which replaces the Run Simulation button during the run), then click the Resume button to continue the run.

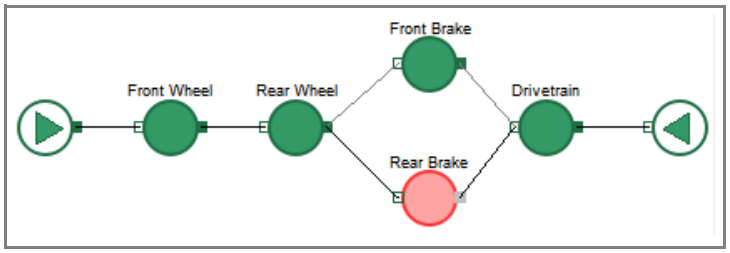
Watch the RBD system go up and down

If a Component is down for any period of time during the simulation, its icon will change from green to red. Whenever the End node is red, the entire system is down.

At the point in the simulation shown below, both brakes are down so the entire system is down, as reflected in the red End node.



Due to redundancy however, if only one of the brakes is down, as shown below, the RBD is still up.



Structure of the model

Before exploring the results of running the RBD, it's a good idea to learn about its structure and components. A reliability model is composed of *blocks* that provide the visible layout of the RBD on the model worksheet and *databases* that work "behind the scenes" and are saved with the model.

☞ This model only has one RBD but a reliability model can have multiple RBD's.

Blocks

Each of the icons on the model worksheet represents one *block*. A block is composed of an icon, a dialog that is accessed by double-clicking the icon, and code that determines how the block behaves over time as the model is run.

As seen in the screen capture on page 11, the model worksheet has 10 blocks:

- The seven *nodes* in this model (a Start Node, five Components, and an End Node) provide the actual layout of the Bicycle RBD. In this model, the Component blocks represent the mechanical parts of the bicycle described earlier.
- The Executive, Distribution Builder, and Event Builder blocks are used to manage aspects of the RBD, such as scheduling events. Every reliability model has one of each of these blocks.

The model's blocks are discussed in more detail starting on page 13.

RBD Databases

This model also contains RBD databases:

- A “visible” database that can be accessed at the bottom of the Database menu.
- A “hidden” database that is reserved by the developers of the Reliability library.

The RBD databases are discussed on page 20.

Blocks used for the stand-alone RBD model

There are 6 different blocks in the model of the bicycle RBD, described below.

Block	Library	Block Function	See Page
 Start Node	Reliability	Performs most of the RBD’s critical tasks, manages the RBD’s behavior over time, and can be thought of as the brain of the RBD. Can alternate between up and down states due to event cycles that have been defined in the Event Builder.	14
 Component	Reliability	Represents resources. Over time, Components alternate between up and down states due to event cycles that have been defined in the Event Builder.	14
 End Node	Reliability	Ends the RBD and reports its status.	15
 Executive	Item	Manages the events that have been scheduled by the Start Node.	16
 Event Builder	Reliability	Creates classes of event cycles for use by the Start Node and Component blocks to model an RBD’s up and down states over time.	16
 Distribution Builder	Reliability	Creates, stores, and provides statistical distribution definitions for time-between-downs (TBD), time-to-down (TTD), and time-to-up (TTU) to be used in the Event Builder block.	18

Descriptions of the blocks in the model

There is nothing fundamentally different about the structure of these different blocks. Any block may create, modify, or present information, and many blocks perform more than one of these functions.

Start Node

A model can have multiple RBD's; however, the Start Node is always the first node in each RBD. It performs most of the RBD's critical tasks, manages the RBD's behavior over time, and can be thought of as the brain of the RBD.

Icon

The icon of the Start Node has three modes. If the Start Node has:

1) None or 1 event cycles, the icon is a single circle as shown at the top of the screenshot at right
2) Two or more event cycles in *series*, the icon is stacked to the right as the middle icon shows
3) Two or more event cycles in *parallel*, the icon is stacked downward as seen in the icon at the bottom

☞ Any node with multiple event cycles will have its icon similarly stacked.

Functions

The Start Node is responsible for the following functions:

- Documenting the RBD's structure in a hidden database while the user builds the diagram
- Documenting input and output information in the RBD visible database
- Associating event cycle instances with nodes in the diagram
- Scheduling up and down events for all nodes in the diagram during the simulation run
- Calculating all paths through the diagram
- Collecting all run results for the entire diagram

Block dialog

▸ Double-click the icon of the Start Node to open its dialog.

☞ When its dialog is opened, a node's icon turns yellow; the icon stays yellow until the dialog is closed.

The Start Node has many tabs, which are described in detail on page 52. For now:

▸ Go to the *Event Cycles* tab. Depending on the popup choice, this tab displays either all the event cycles that are associated with the RBD or just the event cycles for specific nodes.

▸ In the *Add/remove event cycle instances* frame, select **Show event cycles for: Front Brake**

Node	Event Name	Comp-DE Interrupts	RBD-DE Interrupts	Address	Edit Event Class
Front Brake[28]	Brake Cycle	Preserve	Preserve	1:12:0:3	

As seen here, the Component representing the Front Brake has a block number of 28. It uses only one event cycle—the Brake Cycle—which has been defined in the Event Builder block.

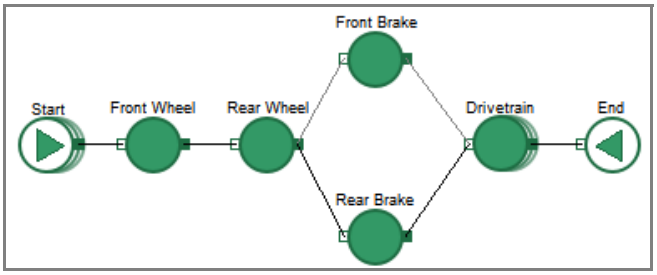
Components

Every RBD must have at least one Component located between its Start Node and its End Node. Components represent resources and are placed in series and/or parallel to each other.

Over time, Components alternate between up and down states due to event cycles that have been defined in the Event Builder block.

The RBD's Components

As seen here, there are five Components in this RBD:



- Front Wheel
- Rear Wheel
- Front Brake
- Rear Brake
- Drivetrain

Each Component represents a mechanical part of the bicycle and each has its own event cycles to specify its up/down behavior.

The Component for the Drivetrain represents the bicycle's five subcomponents—chain, crank, dérailleur, freewheel, and pedals—as discussed on page 9. Each of the subcomponents has its own event cycle to specify its up/down behavior. The Drivetrain icon is stacked to the right, indicating that the block has multiple event cycles in series.

☞ You can assign one or more event cycles directly to an individual Component in that Component's Event Cycle tab. Or, especially if the event cycle causes the RBD to go down, in the Start Node. For example, it is common to assign down events for maintenance in the Start node.

Block dialog

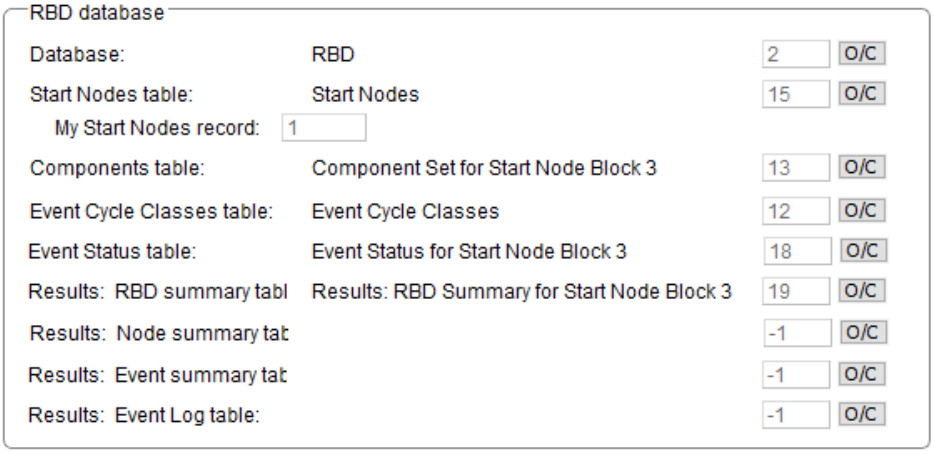
- Open the dialog of the Component labeled **Front Brake.**
- Go to the block's Event Cycles tab. Notice that the Brake Cycle event for the Front Brake, as shown in the Start Node block earlier, is also displayed in this dialog.

Dialog options

The tabs for a Component block (Around Me, Current State, etc.) are the same as for the Start Node block; for detailed information see page 52.

End Node

Each RBD must end at an End Node. This block reports RBD status on its output connectors. And, as is true for the other nodes, its Database tab displays which database tables are associated with this RBD, as shown below.



Executive

The Executive block (Item library) does event scheduling and provides for simulation control, item allocation, attribute management, and more. Each reliability model must have one Executive which must be present on the leftmost side of the model worksheet.

For a reliability model, the Executive manages the events that have been scheduled by the Start Node. Since the Executive handles this automatically, there are no settings to make in the dialog of the Executive block.

Event Builder

Each reliability model has one and only one Event Builder block (Reliability library), regardless of how many RBD's are in the model. This block is used to create classes of event cycles which are stored in a table in an RBD database. The event classes are used by the Start Node and Component blocks to model their up and down states over time.

How it works

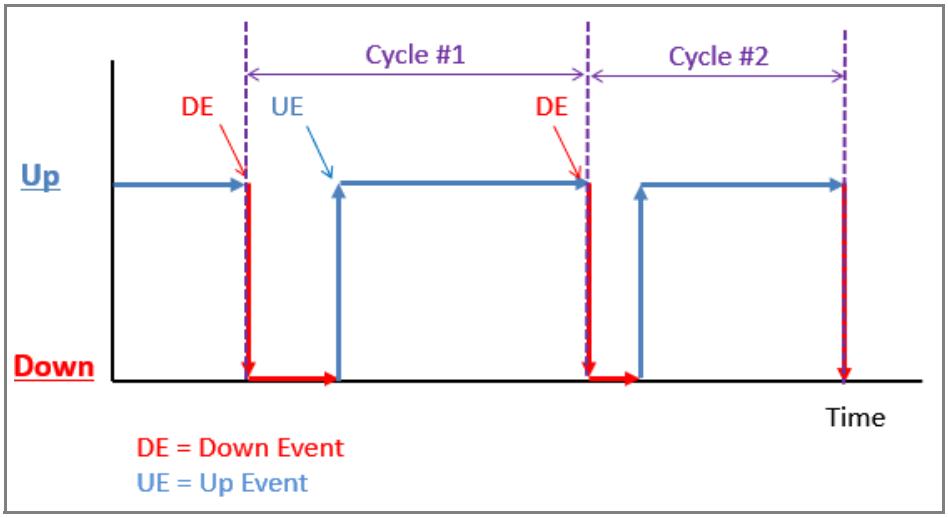
Discrete events are the mechanism ExtendSim uses to cycle Start Node and Component block's through their up and down states over time. The event cycle classes for an entire model are defined in the single Event Builder block.

When you add an Event Builder to the model, ExtendSim auto-creates an RBD database table named *Event Cycle Classes* to store event cycles defined for the model. They are stored as class definitions so that each event cycle can be used as one or more instances by the Start Node and Component blocks.

Event cycles

An Event Cycle is composed of up/down state transitions over time. This cycling behavior is shown below.

☞ Down events are discussed more at "Event cycles" on page 48.



Each record in the Event Cycle Classes database table contains enough information to model up/down event cycling over time.

☞ The distributions that specify the duration of the up and down states are defined in the Distribution Builder block, discussed on page 18.

The block's dialog

▸ Double-click the Event Builder's icon to open its dialog.

The Event Builder block is for creating new event cycle classes or modifying existing ones.

The database

The top part of the Event Builder's dialog indicates that:

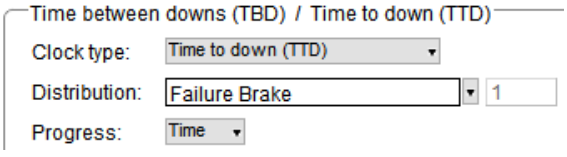
- The database for storing the Bicycle RBD's event cycle definitions is named *RBD*
- Definitions for the event cycles are stored in the database table named *Event Cycle Classes*
- The definitions of the distributions used to characterize the event cycles come from the *Distributions* table of the RBD database

An event cycle

Each bicycle component has one or more corresponding event cycles that were created in the Event Builder block. To see the event cycle for the brakes:

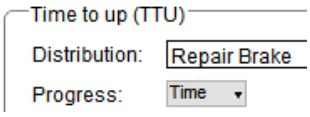
▸ In the *Create/modify event cycle class* dialog frame, choose **Event cycle name: Brake Cycle**. The Brake Cycle event is defined by two distributions:

1) A distribution that specifies how long the brakes will be up—the period of time from when the brakes recovered after the last failure until they will fail again (the *time to down* or TTD). For the



brake this distribution is named *Failure Brake*. It is a Weibull distribution with Scale = 80, Shape = 50, and location = 0.

2) A second distribution that specifies how long it will take to repair the brake (the *time to up* or TTU). As shown here, the Brake Cycle uses a distribution named *Repair Brake*, a Normal distribution with a Mean of 7 and a standard deviation of 0.25.



These distributions have been defined in the Distribution Builder block, discussed below.

☞ There are three types of distribution classes that can be created in the Distribution Builder: time to down (TTD), time to up (TTU), and time between downs (TBD). TBD-based event cycles schedule down events independent of repair and is typically used for calendar-based events such as Maintenance; it is described in detail on page 49.

The bicycle RBD's event cycle classes

To see the event cycles for all of the bicycle's components:

- In the Event Builder's dialog, within the *Location for event cycle classes* frame, click the **Open** button at the right of the *Event cycle classes* field.

A portion of the database table that stores the event cycle definitions for the model is shown below.

Event Name[2]	Event Type[3	Shift[4]	TBD/TTD: Dist Name[5]	TBD/TT D: Cloc	TBD/TT D: Progr	TBD/TTD : Prob > 0	TTU: Dist Name[9]	TTU: Prog ess Type[1
Annual Maintena…	Dist…		TBD - Annual …	TBD	Time	1.000000	TTU - Annual M…	Time
Bi-Annual Maint…	Dist…		TBD - Bi-Annu…	TBD	Time	1.000000	TTU - Bi-Annual…	Time
Brake Cycle	Dist…		TTD - Failure …	TTD	Time	1.000000	TTU - Repair Br…	Time
Drivetrain Chain …	Dist…		TTD - Failure …	TTD	Time	1.000000	TTU - Repair Dr…	Time

☞ As will be seen in the tutorial, classes of event cycles can be defined in the Event Builder block or in an external application, such as Excel, and imported into the model using the database table named Event Cycle Classes.

Distribution Builder

Each reliability model has one Distribution Builder block (Reliability library) which is used to create, store, and provide statistical distribution definitions for time-between-downs (TBD), time-to-down (TTD), and time-to-up (TTU). The Event Builder block can use those definitions when creating event cycles.

How it works

No matter how many RBD's are in a model, all the distributions for the model are defined in a single Distribution Builder block. When you add a Distribution Builder to the model, ExtendSim auto-creates an RBD database table named *Distributions* that is maintained by the block.

As distributions are created, they are stored in the database table as class definitions—one record for each class definition. Each class definition can have many instances, where each instance is created when the class definition is associated with a particular event cycle. This

allows for a distribution to be defined once but used by the Event Builder in multiple event cycles.

The block's dialog

▸ Double-click the Distribution Builder block's icon to open its dialog.

The Distribution Builder block is for creating and modifying classes of distributions for use when creating event cycles in the Event Builder block.

The database

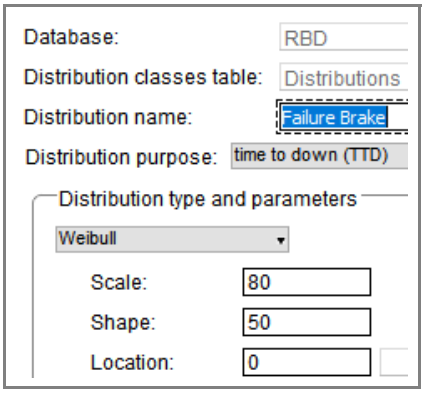
▸ The top part of the block's dialog indicates that:

- The database for storing distribution definitions is named *RBD*
- The definitions of the distributions are stored the *Distributions* table of the RBD database

A distribution

This is where the distribution classes for the Brake Cycle were created. To view an existing distribution:

▸ In the *Create/modify distribution classes* frame, select **Distribution name: Failure Brake**. Its parameters are shown here.



☞ As will be seen in the tutorial, classes of distributions can also be defined in an external application, such as Excel, and imported into the model using the Distributions database table.

The Bicycle RBD's distribution definitions

To see all the distribution classes for the Bicycle RBD:

▸ In the Distribution Builder's dialog, click the **Open** button at the right of the Distributions field.

A portion of the database table that stores the distribution definitions for the Step 1 model is shown below.

	ID[1]	Name[2]	Purpose[3]	Distribution Type[4]	Parameter 1[5]	Parameter 2[6]
1	*TTD - Failure Brake*	Failure Brake	TTD	Weibull	80.00000	50.00000
2	*TTD - Failure Drivetrain Chain*	Failure Drivetrain Chain	TTD	Weibull	90.00000	10.00000
3	*TTD - Failure Drivetrain Crank*	Failure Drivetrain Crank	TTD	Weibull	365.00000	100.00000
4	*TTD - Failure Drivetrain Derailleur*	Failure Drivetrain Derailleur	TTD	Weibull	90.00000	20.00000
5	*TTD - Failure Drivetrain Pedal*	Failure Drivetrain Pedal	TTD	Weibull	365.00000	100.00000
6	*TTD - Failure Drivetrain Freewheel*	Failure Drivetrain Freewheel	TTD	Weibull	365.00000	100.00000
7	*TTD - Failure Wheel*	Failure Wheel	TTD	Weibull	180.00000	100.00000
8	*TTU - Repair Brake*	Repair Brake	TTU	Normal	7.00000	0.25000

RBD databases

As this model was created, an RBD database was automatically created and populated with data. The structure, current state, and results from running the model get stored in tables in this database.

For example, in the Step 1 RBD model random distributions were created in the Distribution Builder to specify the TTD's, TBD's, and TTU's for each component. They are stored in the database's Distributions table as shown above.

Results of running the Bicycle RBD

- Run the simulation

When its run, a reliability model can generate a lot of information as you will see in the next chapter. For now, just look at availability.

Availability

At the end of the simulation run, notice that the cloned dialog item (top right side of the model) indicates that the average availability of the entire RBD was just short of 100%. (The distributions are random so your numbers will be slightly different than shown here.)

Note however that the availability number is essentially meaningless because the bicycle messenger doesn't know what the impact on her business will be. For example, the stand-alone RBD doesn't report how long it takes to deliver messages, how many messages were undelivered, what the revenues and costs are, and so forth. However, by integrating this stand-alone RBD with a process model, she can start to understand what the availability really means. You'll see that in the Tutorial 2.

Next steps

The next chapter, Tutorial 1, shows how to build the RBD you've just explored. That's lot more fun than reading about it!

Reliability
Tutorial & Reference

Tutorial 1: Creating an RBD

Overview

This chapter shows how to build the stand-alone bicycle RBD shown in the previous chapter.

⚠ The tutorials assume you know how to launch ExtendSim, open a library, place a library block on the model worksheet, and connect blocks. If you don't already know how, see either the Discrete Event or Discrete Rate Quick Start Guide.

Steps

1) Start a new model worksheet, then add and connect the blocks
2) Create statistical distribution classes
3) Create event cycle classes
4) Associate the event cycles with the nodes
5) Run the RBD

Start a new model

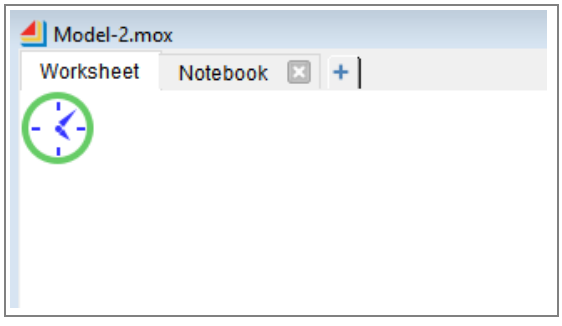
- Launch ExtendSim

By default, when you launch ExtendSim the Getting Started model opens as well as the major ExtendSim libraries and their library windows.

Open a new model worksheet

- Use the toolbar button or the File menu to open a new model.

By default the model opens with an Executive block on the left side, as shown here.

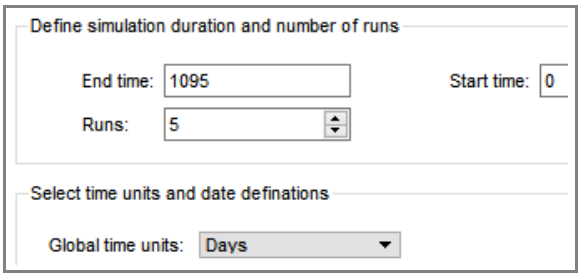


☞ Opening the major libraries on launch and inserting an Executive on the worksheet are default options in the Edit > Options menu.

Set the simulation parameters

The Simulation Setup command opens a window for entering global settings for the model, such as how long and how many times the simulation will run.

- Select the command Run > Simulation Setup
- In the dialog's Setup tab, enter the simulation parameters:
  - **End time: 1095**
  - **Start time: 0** (default)
  - **Runs: 5**
  - **Global time units: Days**



- Leave the other Simulation Setup settings at their defaults
- Click **OK** to close the window

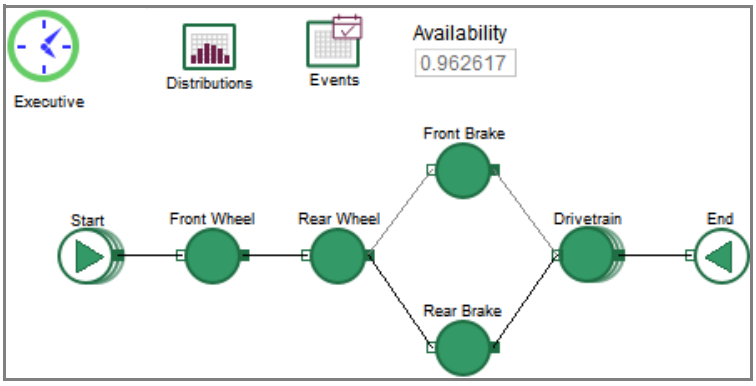
With these settings, the model will run five times, each for a simulated time of 1,095 days.

Save the model

- Choose File > Save Model As and name the file *My RBD*.

Place the blocks on the worksheet

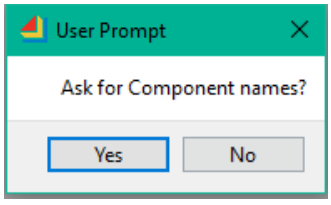
The goal is to create the model shown below.



Add the first three blocks

- Go to the **Reliability** library
- Place the following three blocks on the worksheet:

1)**Start Node**. When the Start Node asks if it should ask for Component names, say **Yes**.



☞ After asking about the Component names, the Start Node will automatically place the other two blocks (Event Builder and Distribution Builder) on the worksheet so you may not need to do the next two steps.

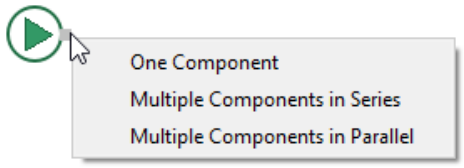
2)**Event Builder**. If it isn't already on the worksheet, add it. Note that if you place this block on the worksheet before placing the Start Node and Distribution Builder block, ExtendSim will automatically put the Distribution Builder on the worksheet as well.

3)**Distribution Builder**. If it isn't already there, place this block on the worksheet.

☞ **Don't add any of the other nodes yet!** You can always manually add, connect, and name the nodes in your RBD. However, using the right-click connect procedure shown below is easier and more fun.

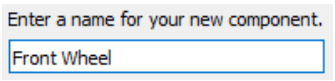
Add the first two Components

As shown in the model worksheet on page 22, the first two Components in your model should be in series.



- To add the first two Components, **right-click on the Start Node's output connector**
- In the dialog that appears, select **Multiple Components in Series**
- In the next dialog, choose to add **2** Components

As ExtendSim adds each Component to the model worksheet, it asks you to enter a name for the node:

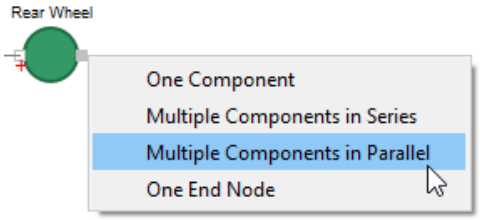


- Enter **Front Wheel** for the first Component
- Enter **Rear Wheel** for the second Component

☞ Notice that the Start Node's icon now has a question mark. It will stay that way until the layout of the RBD has been completed.

Add the second set of components

According to the model assumptions, the next two Components (the Front and Rear Brakes) are in parallel with each other:



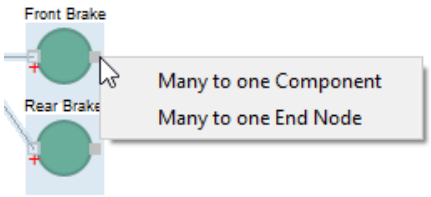
- To add the two Components in parallel, **right-click on the Rear Wheel's output connector**
- In the dialog that appears, select **Multiple Components in Parallel**
- In the next dialog, choose to add **2** Components
- When asked for the names of these new Components:
  - Enter **Front Brake** for the first Component
  - Enter **Rear Brake** for the second Component

Add the Drivetrain

The final Component is the Drivetrain.

- Select the icons for **both the Front Brake and Rear Brake**, as shown below

- *With both icons selected*, right-click on the output connector of **either** the Front Brake or Rear Brake
- From the dialog, select **Many to one Component**
- Name the new Component **Drivetrain**



Add the End Node

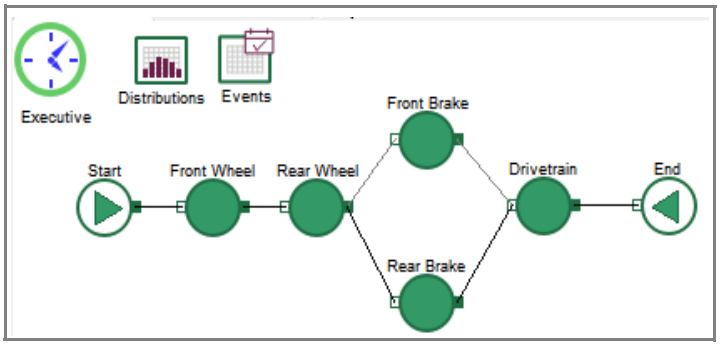
The last step in building an RBD is to add an End Node:

- Right-click on the Drivetrain's output connector and select **End Node**
- Save your model

At this point the RBD layout is complete and the question mark on the Start Node is gone.

Label the other blocks

- Label the blocks as indicated below:
  - Executive
  - Distributions
  - Events
  - Start
  - End



Label the Start and End nodes in the name field at the top of their dialog's General tab.

☞ So that their names can be used by the database, label the Start and End nodes in the frame at the top of their dialog's General tab rather than in the label field next to the Help button. If you check the checkbox, the label will appear above the icon as shown above.

When finished, your model should now look similar to the one shown above.

- Save your model

Distribution classes

The Distribution Builder is used to define different classes of statistical distributions. The definitions are stored in a Distributions database table that is created and maintained by the block. Each record in the database table is a class definition for a particular distribution and each definition can have multiple instances when the class definition is associated with event cycles.

In this model, the event cycles primarily use random distributions to define the TBD's (time between downs), TTU's (time to up), and TTD's (time to down). This section uses distribution classes than have been created in Excel and discusses how to make those distributions available to the Event Builder for use in creating event cycles.

Copying the definitions into the model

If you were only going to use one or two distributions, it would be easiest to just define them in the dialog of the Distribution Builder. However, it is more likely that you will want many dif-

ferent distributions. The best way to do that is to define the distributions in an external application, such as what was done here in Excel, then copy them into a database table in ExtendSim.

☞ See xxx on xxx for how to create a distribution class using the Distribution Builder's UI.

Copy the definitions from Excel

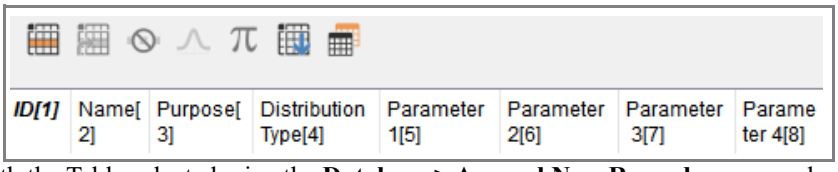
- Launch Excel and locate and open the Excel file named **Data for RBD Tutorial.xlsx**. The file is located at Documents/ExtendSim/Examples/Tutorials/Reliability.
- Go to the workbook's **Distributions** worksheet, a portion of which is shown below.

DISTRIBUTION NAME	PURPOSE	DISTRIBUTION	PARAM 1	PARAM 2
Failure Brake	TTD	Weibull	80	50
Failure Drivetrain Chain	TTD	Weibull	90	10
Failure Drivetrain Crank	TTD	Weibull	365	100
Failure Drivetrain Derailleur	TTD	Weibull	90	20
Failure Drivetrain Pedal	TTD	Weibull	365	100
Failure Drivetrain Freewheel	TTD	Weibull	365	100
Failure Wheel	TTD	Weibull	180	100

- Copy all the cells from **rows 2 - 19** and **columns A - G only**. DO NOT copy the row numbers or the headers.

Open the Distributions database table

- In ExtendSim, open the dialog of the *Distribution Builder* block
- At the bottom of the dialog, click the **Edit Distribution Classes Table** button; this opens the Distributions table of the RBD database as shown below.



ID[1]	Name[2]	Purpose[3]	Distribution Type[4]	Parameter 1[5]	Parameter 2[6]	Parameter 3[7]	Parameter 4[8]

- With the Table selected, give the **Database > Append New Records** command or click the **Append New Records** button in the table's toolbar
- In the Append Records dialog, enter the number **18** and click **OK**; this results in a database table that can hold 18 records

Paste the definitions into the Distribution Builder

⚠ Notice that the Distribution table's first column (ID) is pink. Do not paste any data into that first column as it is reserved for internal use.

- In the database table, click in the cell of the **second column (Name)**, at **row 1**
- Give the **Paste/Paste Cells** command, causing all the definitions to be copied into that database table

ID[1]	Name[2]	Purpose[3]	Distribution Type[4]	Parameter 1[5]	Parameter 2[6]	Parameter 3[7]	Parameter 4[8]
	Failure Brake	TTD	Weibull	80.00000	50.00000	0.00000	0.00000
	Failure Drivetrain Chain	TTD	Weibull	90.00000	10.00000	0.00000	0.00000
	Failure Drivetrain Crank	TTD	Weibull	365.00000	100.00000	0.00000	0.00000
	Failure Drivetrain Derailleur	TTD	Weibull	90.00000	20.00000	0.00000	0.00000
	Failure Drivetrain Pedal	TTD	Weibull	365.00000	100.00000	0.00000	0.00000
	Failure Drivetrain Freewheel	TTD	Weibull	365.00000	100.00000	0.00000	0.00000

- At the bottom of the Distribution Builder's dialog, click the **Commit Distributions Class Changes** button. This saves the data to the RBD's other data structures.

- Verify that the ID field has been written to as shown below, then close the Distributions database table.

	ID[1]	Name[2]	Purpose[3]	Distribution Type[4]	Parameter 1[5]	Parameter 2[6]	Parameter 3[7]	Parameter 4[8]
2	TTD - Failure Drivetr...	Failure Drivetra...	TTD	Weibull	90.00000	10.00000	0.00000	0.00000
3	TTD - Failure Drivetr...	Failure Drivetra...	TTD	Weibull	365.00000	100.00000	0.00000	0.00000
4	TTD - Failure Drivetr...	Failure Drivetra...	TTD	Weibull	90.00000	20.00000	0.00000	0.00000

⚠ Use the Commit button anytime you add records to the database. If you don't click the Commit button, the ID field won't be updated, causing database errors.

- Click the OK button to close the Distribution Builder's dialog
- Save your model; this saves the database changes with your model

The distribution classes have now been entered and the distributions will be available for use by the Event Builder.

Event cycle classes

The Event Builder is used to define different classes of event cycles, which are discussed more fully on page 50. The definitions are stored in an Event Cycles database table that is created and maintained by the block. Each record in the database table is a class definition for a particular event cycle and each definition can have multiple instances when it is associated with a particular Start Node or Component block. This allows for an event cycle to be defined once but used multiple times.

Event cycle table

As was true for the Distribution Builder, if you were only going to have one or two event cycles you could just define them in the dialog of the Event Builder. However, in most cases you will have multiple event cycles and it would be more efficient to define them in an external application as was done for this tutorial.

☞ See xxx on xxx for how to create an event cycle class using the Event Builder's UI.

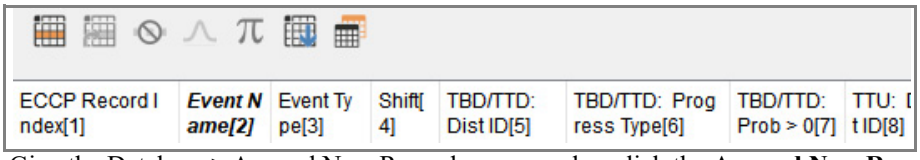
Copy the definitions from Excel

- If it isn't already open, launch Excel then locate and open the Excel file named **Data for RBD Tutorial.xlsx**. The file is located at Documents/ExtendSim/Examples/Tutorials/Reliability.

- Go to the Excel workbook's **Event Cycles** worksheet.
- Copy all the cells from **rows 2 - 10** and **columns A - I only**. DO NOT copy the headers or row numbers.

Open the Event Cycles database table

- In ExtendSim, open the dialog of the *Event Builder* block
- At the bottom of the dialog, click the **Edit Event Cycle Classes Table** button; this opens the Event Cycles Classes table of the RBD database as shown below.



ECCP Record Index[1]	***Event Name[2]***	Event Type[3]	Shift[4]	TBD/TTD: Dist ID[5]	TBD/TTD: Progress Type[6]	TBD/TTD: Prob > 0[7]	TTU: t ID[8]

- Give the Database > Append New Records command or click the **Append New Records** button in the table's toolbar
- In the Append Records dialog, enter the number **9**; this results in a table that can hold 9 records

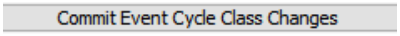
Paste the definitions into the Event Builder

Notice that the Event Cycle Classes table's first column (ECCP Record Index) is pink. Do not paste any data into that first column as it is reserved for internal use.

- In the database table, click in the cell of the **second column (Event Name)**, at **row 1**
- Give the **Paste/Paste Cells** command, causing the definitions to be copied into that table

ECCP Record Index[1]	***Event Name[2]***	Event Type[3]	Shift[4]	TBD/TTD: Dist ID[5]
0	*Annual Maintenance*	Distribution		TBD - Annual Maintenance
0	*Bi-Annual Maintenance*	Distribution		TBD - Bi-Annual Maintenance
0	*Brake Cycle*	Distribution		TTD - Failure Brake

- At the bottom of the Event Builder's dialog, click the **Commit Event Cycle Class Changes** button. This saves the data to the RBD's other data structures.



- Verify that the ECCP Record Index field has been written to, then close the Event Cycles database table

Use the Commit button anytime you add records to the database. If you don't click the Commit button, the ECCP Record Index won't get updated, causing database errors.

- Click the OK button to close the Event Builder's dialog
- Save your model

The event cycles classes will now be available for use as instances by the Start and Component nodes.

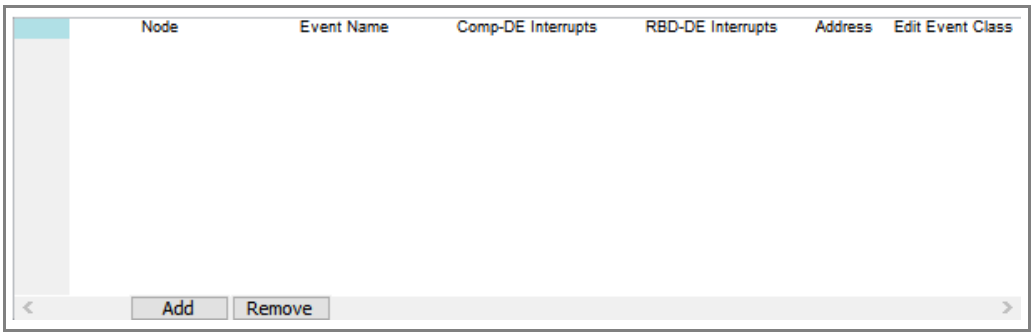
Associate the event cycles with the nodes

The only thing left to do is to specify which event cycles each node will use during the simulation. An event cycle class can have multiple instances, such as the Wheel Failure used by both the Front Brake and the Rear Brake. And each node can use more than one event cycle; for example, the Drivetrain has five event cycles—one for each subcomponent of the drivetrain.

You can associate an event cycle with a Component either directly using the Component's dialog or using the Start Node's dialog. This example uses the Start Node to add event cycle instances for the Start Node as well as for the Components.

Open the Start Node

- Open the dialog of the **Start Node**
- Select the block's **Event Cycles** tab
- Go to the **Add/remove event cycle instances** frame, shown below



Bicycle maintenance

The bicycle RBD has two event cycles that will cause the bicycle to be unavailable for a period of time, no matter what else is happening:

1) The annual maintenance on the drivetrain.
2) The bi-annual maintenance of the front and rear wheels.

The event cycles for these maintenance events will be associated with the Start Node.


If an event cycle will take the entire RBD down, it should be specified as a Start Node event.

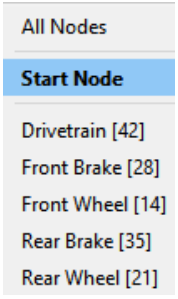

The maintenance of the wheels and the drivetrain impacts when their next down events will occur. For example, once the drivetrain is serviced, the event cycles for its subcomponents need to be reset to use new TTD's. This will be shown in "Event cycle induced interrupts" on page 31.

Drivetrain's annual maintenance

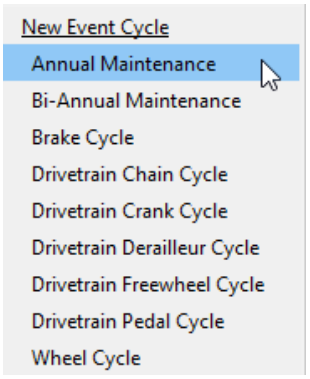
The event cycle for the drivetrain is named Annual Maintenance.

- In the Start node's *Add/remove event cycle instances* frame:

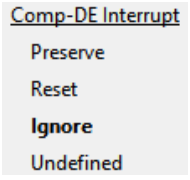
- Choose to **Show event cycles for: Start Node**



- At the bottom of the *Add/remove event cycle instances* frame, click the **Add** button
- In the popup menu, select **Annual Maintenance**



- For the two messages that appear (Component-DE Interrupt and RBD-DE Interrupt) select **Ignore**.



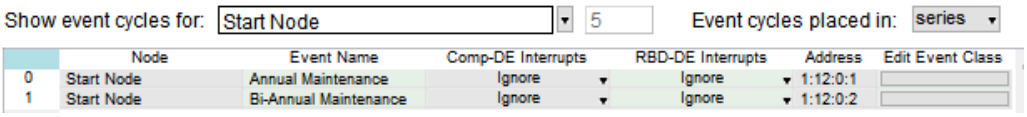
☞ As discussed in "Associate the interrupts" on page 31, these settings define when and how an event cycle's normal down event scheduling can be interrupted. The bicycle's maintenance events are going to occur independent of the RBD's changing states, so you choose to ignore any interrupts.

Wheels' bi-annual maintenance

Both wheels get checked twice a year. Their event cycle is named Bi-Annual Maintenance.

- Repeat the above steps, adding **Bi-Annual Maintenance** as a Start Node event cycle.
- For the two interrupt messages that appear (Component-DE Interrupt and RBD-DE Interrupt) select **Ignore**.

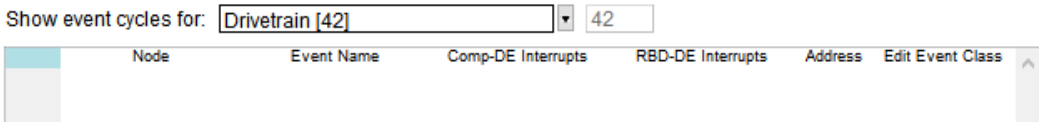
The Start Node should now have 2 instances of event cycles, as shown below. Notice that since the event cycles are placed in series rather than in parallel, in the model the Start Node's icon is stacked to the right. This indicates that when either of those event cycles occur, the Start Node (and hence the entire RBD) will go down.



Drivetrain non-maintenance event cycles

You've associated the Drivetrain's annual maintenance with the Start Node. However, the Drivetrain also has 5 subcomponents—Chain, Crank, Dérailleur, Freewheel, and Pedals—and each subcomponent has its own event cycle. Those event cycles need to be associated with the Drivetrain.

- In the Start Node's *Add/remove event cycle instances* frame, choose to **Show event cycles for: Drivetrain**



- Click the **Add** button each time to add the five Drivetrain event cycles—**Chain Cycle, Crank Cycle, Dérailleur Cycle, Freewheel Cycle, and Pedal Cycle**—to the Drivetrain.
- For each Component-DE Interrupt message, select **Preserve**
- For each RBD-DE Interrupt message, select **Preserve**

☞ Some of the event cycles need to keep track of their progress towards their next down events, even if the RBD goes down. Choosing Preserve means that, when the RBD comes back up, those event cycles will pick up where they left off.

The event cycle instances for the Drivetrain should look as shown here:

Node	Event Name	Comp-DE Interrupts	RBD-DE Interrupts	Address
Drivetrain[42]	Drivetrain Chain Cycle	Preserve	Preserve	1:12:0:4
Drivetrain[42]	Drivetrain Crank Cycle	Preserve	Preserve	1:12:0:5
Drivetrain[42]	Drivetrain Derailleur C...	Preserve	Preserve	1:12:0:6
Drivetrain[42]	Drivetrain Freewheel ...	Preserve	Preserve	1:12:0:7
Drivetrain[42]	Drivetrain Pedal Cycle	Preserve	Preserve	1:12:0:8

- Click OK to close the Start Node's dialog.
- Save the model to save your changes.

Notice that the Drivetrain icon is stacked to the right, indicating that its associated event cycles are in series. If even one of those event cycles occur, the Drivetrain will go down. And since there is no redundancy or load sharing for the Drivetrain, the entire RBD will also go down.

☞ The same event cycles shown for the Drivetrain in the Start Node will be displayed in the Drivetrain's Event Cycles tab.

Brakes

- In the Start Node's *Add/remove event cycle instances* frame, choose to **Show event cycles for: Front Brake**
- Click the **Add** button and add the **Brake Cycle**

- For the Component-DE Interrupt and RBD-DE Interrupt popups, select **Preserve**
- Duplicate the above steps to add the **Brake Cycle** to the **Rear Brake**

Wheels

- In the Start Node's *Add/remove event cycle instances* frame, choose to **Show event cycles for: Front Wheel**
- Click the **Add** button and add the **Wheel Cycle**
- For the Component-DE Interrupts and RBD-DE Interrupts popups, select **Preserve**
- Duplicate the above steps to add the **Wheel Cycle** to the **Rear Wheel**
- Save the model

Associate the interrupts

An event cycle's normal down event scheduling can be interrupted when:

1) The RBD goes down
2) A Component goes down
3) One event cycle affects another event cycle

The options you selected for the interrupt settings in "Drivetrain's annual maintenance" on page 28, (Component-DE Interrupt and RBD-DE Interrupt) determine what happens if the Component or the RBD goes down.

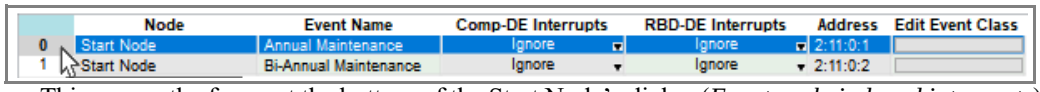
You might expect, for example, that servicing the bicycle's front wheel would cause that wheel's next down event to be postponed. That is an example of one event cycle affecting the timing of another event cycle.

Event cycle induced interrupts

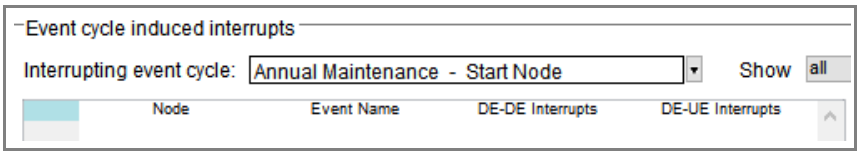
The occurrence of a down in one event cycle can interrupt the normal scheduling of other event cycles' down events. In the bicycle RBD, maintenance down events result in the drivetrain and wheel event cycles needing to be "reset" to use a fresh set of TBD's or TTD's.

A table in the Start node is used to specify when and how these types of interrupts affect specific event cycles.

- In the Start Node, go to the Event Cycles tab
- Choose to show event cycles for the **Start Node**
- Select the first row in the cycles table by clicking row heading #0, as shown below



This causes the frame at the bottom of the Start Node's dialog (*Event cycle induced interrupts*) to display the table below. Notice that the interrupter is the Annual Maintenance.

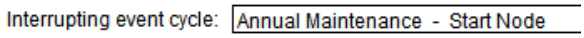


This table is where you will:

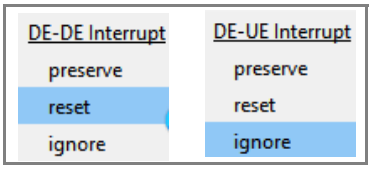
- Choose which event cycles will be affected (interrupted) by the occurrence of a maintenance down event
- Specify how and when the interrupt will affect the event cycle

Annual maintenance and the Drivetrain Chain Cycle

With the Annual Maintenance event cycle selected as the interrupter:



- In the *Event cycle induced interrupts* table, click the **Add** button
- From the popup menu that appears, select the **Drive-train Chain Cycle**
- In the DE-DE Interrupt popup that appears, select **reset**
- In the DE-UE Interrupt popup that appears, select **ignore**



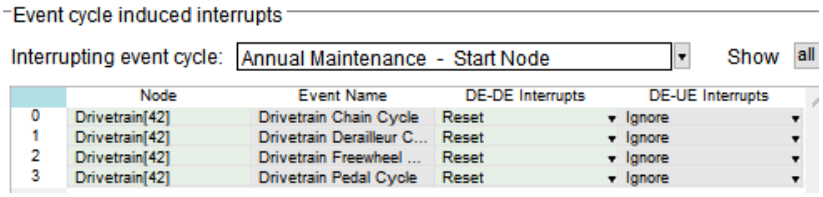
Other drivetrain subcomponent event cycles affected by the annual maintenance

As mentioned in the assumptions, the Crank is not serviced as part of the annual maintenance. However, the other subcomponents are serviced, which affects their next down events.

- **Skipping the Crank**, repeat the above steps for 3 of the 4 remaining drivetrain subcompo-nents:
  - Drivetrain **Dérailleur** Cycle
  - Drivetrain **Freewheel** Cycle
  - Drivetrain **Pedal** Cycle
- Save your model

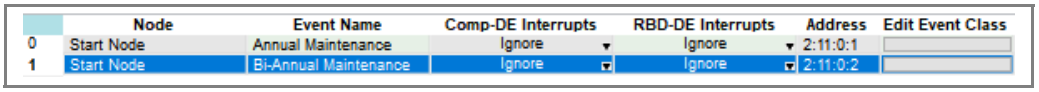
⚠ Since it is not serviced as part of the annual maintenance, do not add the Drivetrain Crank Cycle to the table. For Tutorial 2 you will simulate what happens when the crank needs repair.

The interrupts should appear as below:



Bi-annual maintenance and the wheels

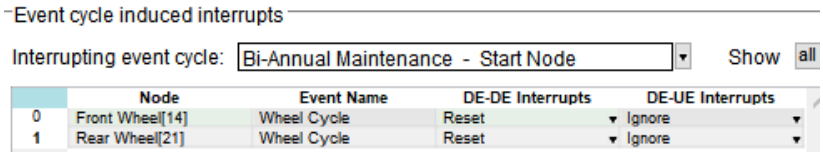
- Select the second row in the cycles table by clicking row heading #1, as shown below



- In the *Event cycle induced interrupts* table, click the **Add** button
- From the popup menu that appears, select **Wheel Cycle - Front Wheel**

- From the popups, choose **reset** for the DE-DE Interrupt and choose **ignore** for the DE-UE Interrupt.
- Repeat the above steps to add **Wheel Cycle - Rear Wheel**

The table should appear as below:



- Save your model

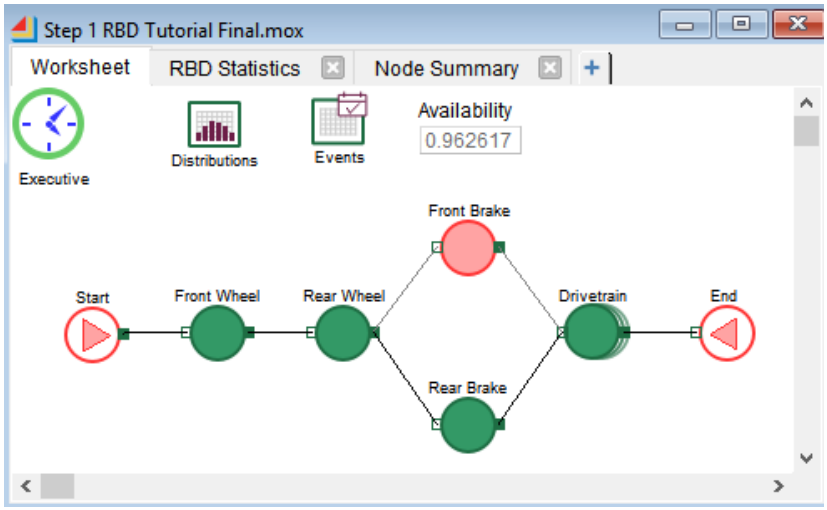
Availability

At this point you've entered all the information needed for the bicycle RBD. So that your model looks more like the example model on page 10, clone the availability field from the Results tab of the Start Node onto the model worksheet. This is the average availability of the entire RBD as calculated for the 5 runs.

Conclusion

Your creation of the RBD is complete.

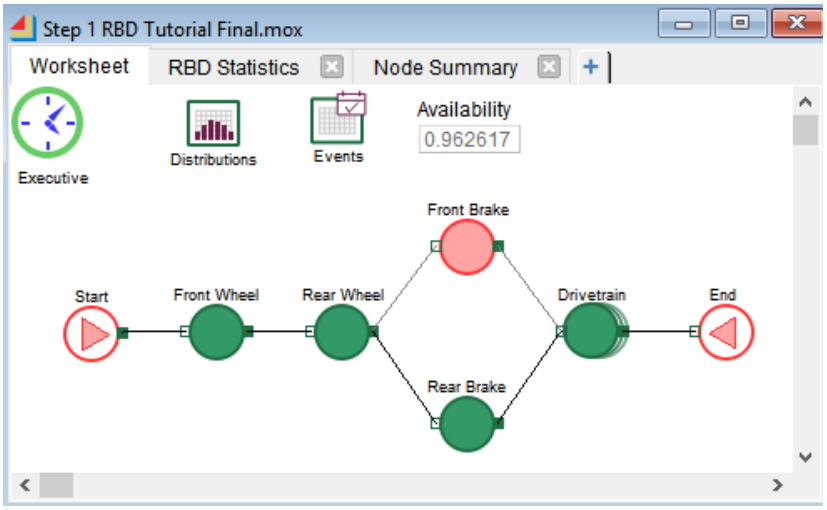
If you've followed all the steps, your model should be similar to the Step 1 RBD Tutorial Final model shown below and located at Documents/ExtendSim/Examples/Tutorials/Reliability.



Run the model

Even though it runs five times for a simulated period of three years, the model may run too fast for you to see changes. You can slow the model speed considerably by setting animation speed

at the slowest. This allows you to see when the nodes are up or down and when the RBD is up or down.



Results

As mentioned in the Introduction, an RBD graphically and statistically describes when scheduled and unscheduled downs occur for individual resources and what impact that has on the availability of entire system. You see the graphical description when you run the simulation with animation on. The statistical description is reported in the Start Node.

RBD statistics section

- Double-click the icon of the Start Node to open its dialog
- Go to its **Results** tab.

☞ When a node's dialog is opened, its icon turns yellow; it stays yellow until the dialog is closed.

The **RBD statistics** frame at the top of the Results tab will be similar to the following. Since this model uses random numbers for its distributions, your results will differ.

RBD statistics

	Down Events	Total Downs	Up Time	Down Time	Availability
Average:	44	31.6	1048.861	46.13944	0.956796
Std Dev:	0.894427	2.727636	14.97233	14.97233	0.013686

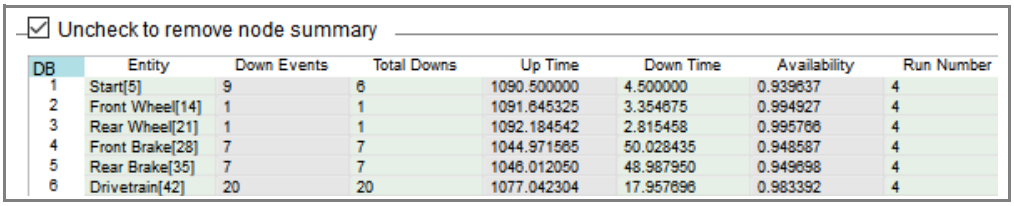
DB	RBD	Down Events	Total Downs	Up Time	Down Time	Availability	Run Number
1	Start[5]	43	30	1053.698812	41.301388	0.959943	0
2	Start[5]	45	34	1045.520850	49.479150	0.954155	1
3	Start[5]	44	27	1074.482334	20.517666	0.980720	2
4	Start[5]	43	33	1040.871293	54.128707	0.949526	3
5	Start[5]	45	34	1029.729713	65.270287	0.939637	4

The simulation is run five times and the results for each run are shown in the table in the RBD statistics frame. The statistics at the top of the frame report the average for all the runs.

Notice that the sum of the Up Time and Down Time for each of the runs is 1095. This corresponds to the model's simulation time of 1095 days.

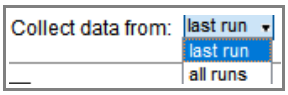
Node summary section

- The Start Node's Results tab also has a **Node summary** frame that should look similar to this one:



DB	Entity	Down Events	Total Downs	Up Time	Down Time	Availability	Run Number
1	Start[5]	9	6	1090.500000	4.500000	0.939637	4
2	Front Wheel[14]	1	1	1091.645325	3.354675	0.994927	4
3	Rear Wheel[21]	1	1	1092.184542	2.815458	0.995766	4
4	Front Brake[28]	7	7	1044.971565	50.028435	0.948587	4
5	Rear Brake[35]	7	7	1046.012050	48.987950	0.949698	4
6	Drivetrain[42]	20	20	1077.042304	17.957696	0.983392	4

- This table provides information for the Start Node and the five Components in this RBD. As selected in this frame's popup menu, the information is for the last run (row 5 in the RBD Statistics table) rather than for all runs. (Simulation runs are numbered starting at 0, so the last run is run number 4.)



Things to notice

- The Start Node's RBD Statistics frame gives information about the entire RBD while the Node Summary frame gives information about each node.
- The Down Events for each run in the RBD Statistics frame (45 for the 5th run as shown above) will be the same as the sum of the Down Events for that run in the Node Summary frame.
- However, as seen for the 5th run in the two frames, the sum of the actual Total Downs for the RBD (34) is less than the Total Downs for all the nodes (42). This makes sense due to the redundancy of the brakes; the RBD doesn't go down unless both of the brakes are down so there are fewer downs for the RBD itself than for the nodes.
- When there are downs for the Front and Rear Wheels it is due to failures, not to their bi-annual maintenance. Since anytime maintenance is done the entire bicycle is brought off line, the event cycles for bi-annual maintenance have been associated with the Start Node rather than with the wheels.
- The simulation runs for 3 years. Each year there is an annual maintenance and two bi-annual maintenances, resulting in three maintenance events per year. However, since the second bi-annual maintenance occurs at the same time as the annual maintenance, there are only 2 Total Downs per year associated with the Start Node, for a total of 6.

Next steps

The next chapter shows how to integrate the RBD you just built with a discrete event model.

Reliability
Tutorial & Reference

Tutorial 2: Adding PSS to RBD

⚠ If you aren't already familiar with the ExtendSim Item library, see the Discrete Event QSG (Quick Start Guide) located at Documents/ExtendSim/Documentation.

The purpose of this tutorial is to demonstrate how to interface process simulation with an RBD. In particular, it shows how the RBD you built in the previous tutorial can be integrated with an item-based process model of the message delivery business.

What integration adds to the model

Adding process simulation to RBD leads to a better understanding of the impact that availability has on overall system performance in a way that dedicated RBD tools simply cannot do. For example, integration can answer some business-related questions that the RBD's availability number can't answer, such as how changes in availability impact the business's ability to deliver messages.

In addition, the occurrence of down events in the stand-alone RBD must by definition be based on the progression of time, since there is no other information available. However, when RBD is integrated with process simulation, the process model can supply the RBD with usage information that reflects the wearing of the components. This allows for a more realistic representation of the process, where how much a component is used affects when its next down event will occur.

☞ Integrating process simulation with RBD allows the RBD's components to progress to a down event based on factors other than the mere advance of time.

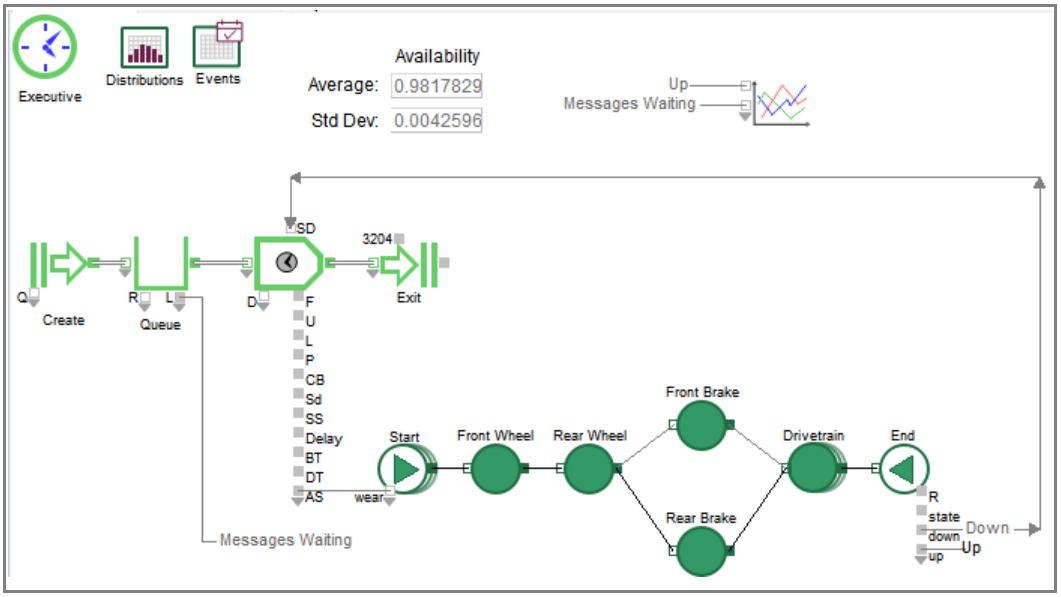
The tutorial models in this chapter

There are two parts to this tutorial:

1) Integrate an item-based process model (2A) with the RBD you created in the previous tutorial.
2) Explore what happens if the wait for delivering the messages gets too long and messages renege (2B).

☞ Rather than have you build the item-based portion of the model, both sections of the tutorial start with a pre-built model.

The 2A model



- Open the model *Step 2A RBD Tutorial* located at Documents/ExtendSim/Examples/Tutorials/Reliability.)
- Save the model as MyRBDItems

What the model does

This model integrates the RBD you built in the previous tutorial with a simple MM1 queuing model that represents the message delivery process.

- In the process section of the model, messages that need to be delivered arrive randomly from a Create block, sit in a Queue while they wait to be delivered, and are processed by an Activity block. The Activity represents the bicycle's operation and it tells the RBD whether the bicycle is being used or not.
- The RBD determines the bicycle's availability (its up and down states) based on the wearing of the Components. It reports that information to the Activity block, letting it know if and when the bicycle is available to deliver messages.
- At the end of the simulation, the down events and the messages waiting for delivery are displayed on the graph.

Assumptions for the process portion

As you can see in the dialogs of the Create, Queue, and Activity blocks:

- Messages arrive randomly, represented by an exponential distribution with a mean of 0.35 days
- They are stacked in a first-in-first-out (FIFO) order as they wait for delivery
- The bicycle messenger can only deliver one message at a time. The delivery time is specified by a triangular distribution with the following parameters:

- Minimum: 0.10 days
- Maximum: 0.50 days
- Most likely 0.25 days

So that you can focus on the integration, the process portion of the model has already been completed.

The RBD portion

Since the maintenance events are calendar-based and will occur at set times no matter what the following TBD/TTD event cycles are the only ones that would be affected by wearing:

- Brake Cycle
- Drivetrain subcomponents: Chain, Crank, Dérailleur, Freewheel, and Pedal cycles
- Wheel Cycle

☞ Wearing can affect progress towards the next down event and thus the duration of the time-between-downs (TBD) or time-to-down (TTD). However, it never impacts the time to up (TTU). Thus only the TBD/TTD Progress Types need to be changed.

Integrating the two portions

Your task for this tutorial is to:

1) Change the TBD/TTD progress type for the affected event cycles from *time* to *wear*
2) Connect the RBD and process simulation sections so they can exchange data with each other

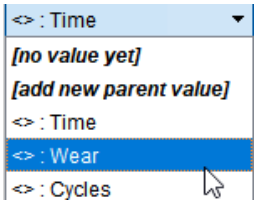
Change the event cycle progress type

There are two ways you can change the TBD/TTD Progress Type from Time to Wearing:

1) Select an event cycle in the Event Builder's dialog. In the *Time between downs (TBD)/Time to down (TTD)* frame, change the event cycle's Progress from Time to Usage, select Wear as the Usage type, and save the change. Repeat for each event cycle and save the model.

2) Or, open the Event Cycle Classes table in the dialog of the Event Builder, make all the changes there, and save your model.

- In the dialog of the Event Builder, open the Event Cycle Classes table. Notice that, in the TBD/TTD Progress Type column, all event cycles are set to Time.
- In the TBD/TTD Progress Type column, for the **Brake Cycle**:
  - Use the popup menu to change its progress type from **Time** to **Wear** as shown here
  - After the cell has switched from Time to Wear, copy the word **Wear** as it is displayed in the cell
- For the Drivetrain Chain Cycle, paste into the Progress Type cell so that Time is replaced by **Wear**.



☞ Although copy/paste is fast, you could alternatively use the popup to select Wear for each event cycle.

- Paste **Wear** into the Progress Type cells for the five remaining event cycles:
  - Drivetrain Crank Cycle
  - Drivetrain Dérailleur Cycle
  - Drivetrain Freewheel Cycle
  - Drivetrain Pedal Cycle
  - Wheel Cycle

The table should now show all the event cycles, except for the Annual and Bi-Annual Maintenance, having a progress type of Wear.

TBD/TTD: Dist ID[5]	TBD/TTD: Progress Type[6]
TTD - Annual Maintenance	Time
TTD - Bi-Annual Maintenance	Time
TTD - Failure Brake	Wear
TTD - Failure Drivetrain Chain	Wear
TTD - Failure Drivetrain Crank	Wear
TTD - Failure Drivetrain Derailleur	Wear
TTD - Failure Drivetrain Freewheel	Wear
TTD - Failure Drivetrain Pedal	Wear
TTD - Failure Wheel	Wear

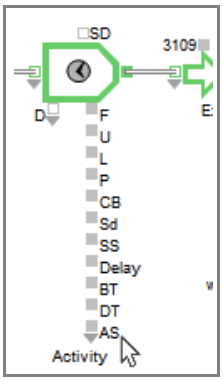
- Close the database table, click OK in the Event Builder's dialog to close it, and save the model to save the changes you've made to the database.

☞ Since you haven't added any records, you don't need to Commit Event Cycle Class Changes, you just have to save the model.

Connect the process model to the RBD

Activity

- On the Activity block's Shutdown tab:
  - Check the box to **Enable shutdown**.
  - Leave the other options as they are and click OK to close the dialog and save changes.
- Notice that there is now an SD (shutdown) input connector at the top of the block's dialog. This will allow the RBD to control when the Activity (the bicycle) is able to deliver messages and when it is not.
- At the bottom of the Activity block's icon, drag down on the variable output connector to reveal the AS (Advanced Status) output as shown here.



End Node

- In End Node's General tab, choose to **Show outputs**, then save and close the dialog. This puts a variable output connector at the bottom of the End Node's icon.
- Drag down on the variable connector to reveal the **up** output connector.

Start Node

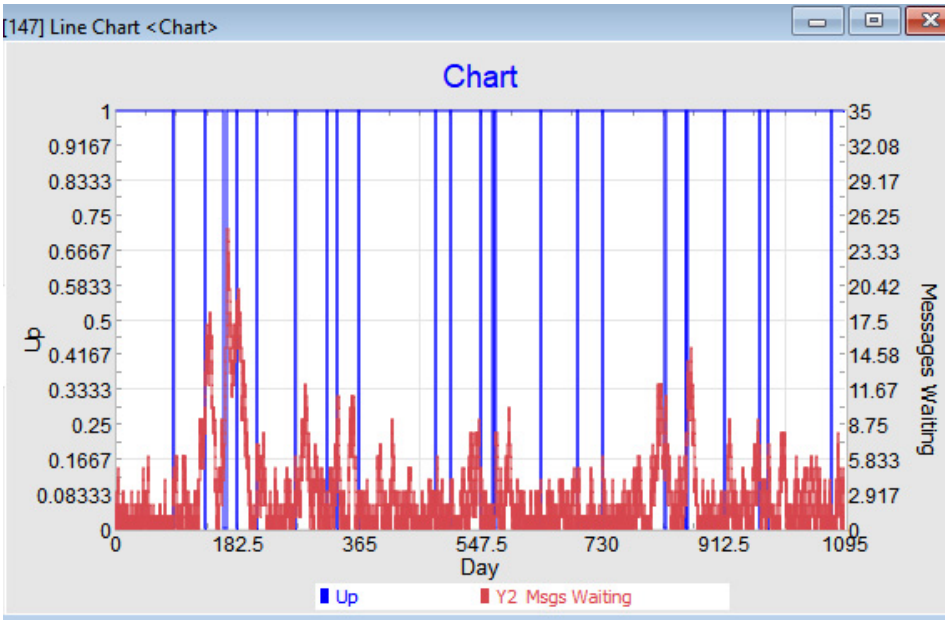
- In the Start Node's Connectors tab, select to **Show input connectors**, then save and close the dialog.
- Notice that this puts a variable input connector at the bottom of the Start Node's icon and the first input is **wear**.
- Draw a connection line from the Start Node's **wear** input to the **AS** output on the Activity.

Make named connections

- Draw a connection line (or make a named connection) from the *down* output of the End Node to the *SD* input on the Activity
- Create a named connection named Up and use it in 2 places:
  - Connected to the *up* output of the End Node
  - Connected to the top input on the Chart block

Run the simulation

The graph should appear similar to the one shown below. This shows the affect that the bicycle's unavailability has on the buildup of messages in the Queue.



The 2B model

THIS SECTION IS NOT FINISHED

Reliability
Tutorial & Reference

Tutorial 3: Add Reliability to a Rate Model

THIS CHAPTER IS NOT FINISHED

Reliability
Tutorial & Reference

Reference

THIS CHAPTER IS NOT FINISHED

RBD terminology

Term	Definition
Availability	Percentage of time a resource is in an “up” state and available to perform work.
Component	The nodes located between a Start Node and an End Node in the interior of an RBD. Components represent resources and are placed in series and/or parallel to each other. Over time, Components alternate between up and down states due to event cycles.
Distribution	A description of a random phenomenon that specifies the length of time a resource will be in an up or a down state.
Down Event	A Down event is the point in time when an event cycle switches from the up state to the down state. Down events are either scheduled or unscheduled and can be due to failure, off-shifting, maintenance, or repair.
Edges	Edges describe how nodes in an RBD are related to each other. In ExtendSim, edges are represented by the connection lines between nodes.
Event Cycle	Describes an alternating behavior of cycling through up and down states over time. Event cycles are sometimes called Failure modes, although that is a more limiting term because not all downs are caused by failures.
Failure Mode	The manner in which a design, process, product, or service will cycle through failure and non-failure states over time.
k of N	The “k” defines the minimum number of upstream parallel components (N) that need to be in an Up state in order for the “one downstream component” to be up.
Load Sharing	A form of redundancy in which there is a parallel structure that supports the workload. If one of the load sharing components fails, it causes a higher share of the workload for the remaining components, increasing the wear rate.
Nodes	The interconnected shapes that form the structure of the RBD. They consist of a Start Node, an End Node, and one or more Components.

Term	Definition
Parallel Nodes	A set of Component nodes placed in parallel to each other indicates redundancy. Unless the entire set of parallel nodes is down, that section of the RBD will be available to perform work.
PSS	Process simulation software. Event-based tools such as the ExtendSim Item library for discrete event simulation or Rate library for discrete rate simulation.
RBD	Reliability block diagram. A network of interconnected nodes that alternate over time between up and down states in order to model the availability of the system as a whole. A model may have more than one RBD; each RBD has a Start Node, an End Node, and one or more Components.
Reliability	The probability the RBD will remain in its up state for the entire duration of the run.
Resources	The means by which process activities and operations are performed. Their lack of availability can cause constraints on the system.
Scheduled Downs	A planned Down event such as for maintenance or off-shifting.
Serial Nodes	A set of Components that are placed one after the other. The entire set must be up for that part of the RBD to be available to perform work.
Standby	The case in which a component has a backup component in a Down (idle) state until it is needed. Note that the backup component could have the same failure rate while down as it does while up, causing it to be a “hot standby”.
TBD	Time-Between-Downs. The period of time between the start of a Down event and the start of the following Down event.
TTD	Time-To-Down. The period of time between the start of an Up event and the start of the following Down event.
TTU	Time-To-Up. The period of time between a Down event and the following Up event.
Unscheduled Downs	An unexpected Down event such as a failure.
Up Event	An Up event is the point in time when the event cycle switches from the down state to the up state.

For a pictorial representation of many of these terms, see “Basics” on page 47.

☞ By default ExtendSim assumes the system is in an up state at the start of the simulation.

Example models

☞ The following models are located at Documents/ExtendSim/Examples/Reliability/RBD Bicycle. They are variations on the RBD model of the bicycle discussed in the tutorial chapters.

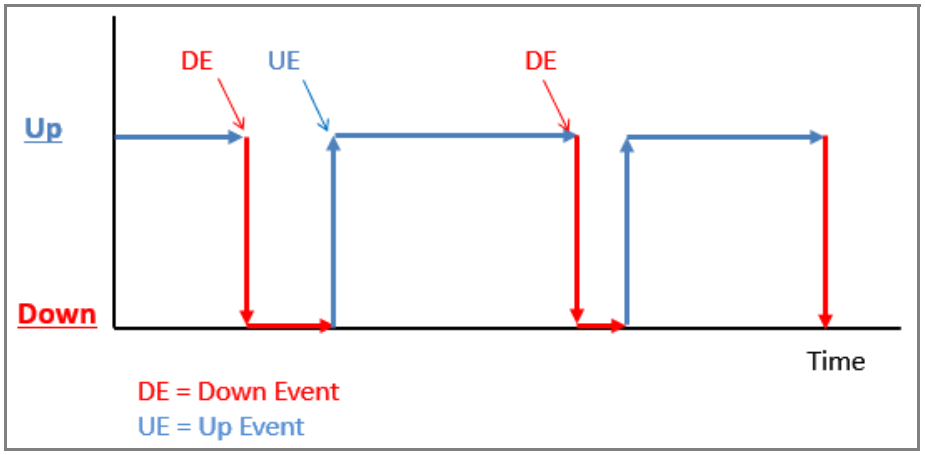
Model	Description
Shift Blocks	Same as the Step 1 RBD model discussed in Tutorial 1 except it uses Shift blocks (Item library) to specify both the annual and bi-annual maintenance event cycles.
Item-Based Repair	Demonstrates how to translate repair events into items that must travel through a repair process before returning to the RBD in a fixed state. By default, repairs are modeled using a random distribution to define how long the repair will take. However, that black boxing of the repair process might not provide a high enough level of fidelity. When the availability of the critical resources needed to make repairs affects repair duration in meaningful ways, Reliability users have the option to use the ExtendSim process modeling capabilities to model the repair process with a higher level of fidelity.
Standby (A)	Uses an Equation block (Value library) that is wired directly to the RBD to model "Standby". Standby is a general purpose reliability concept where a component sits idle until it is called into service. In the model the rear brake is not used until the front brake fails.
Standby (B)	The same as the Standby "A" model except the Equation block is not wired to the RBD with connection lines. Instead the ExtendSim internal database and Link Alerts notify the Equation when the front or rear brake has gone into the failed state. The Equation then issues the Standby directive (discussed above) to the RBD via the database instead of via connection lines.
Load Share (A)	Load share is another general reliability concept where two or more components (in this case the front and rear brakes) share the load. When one of the components goes down, the wear rate on the other component(s) speeds up. This model uses an Equation directly wired to the RBD.
Load Share (B)	The same as the Load Share "A" model but wireless. as discussed above for the Standby (B) model.
k of N (A)	The "k of N" is a general reliability concept used to define how many (k) of the upstream parallel components (N) need to be in an up state in order for the "one downstream component" to be up. In this model, k = 1, the upstream parallel components are the front and rear brakes, and the "one downstream component" is the Drivetrain. That is, the drivetrain will be considered "up" if at least 1 of the 2 brakes is up. This model uses an Equation block wired directly to the RBD to determine how many of the parallel components are up and whether the "one downstream component" should be up or down.
k of N (B)	The same as the k of N "A" model but wireless, as discussed above for the Standby (B) model.

Basics

Up events and Down events

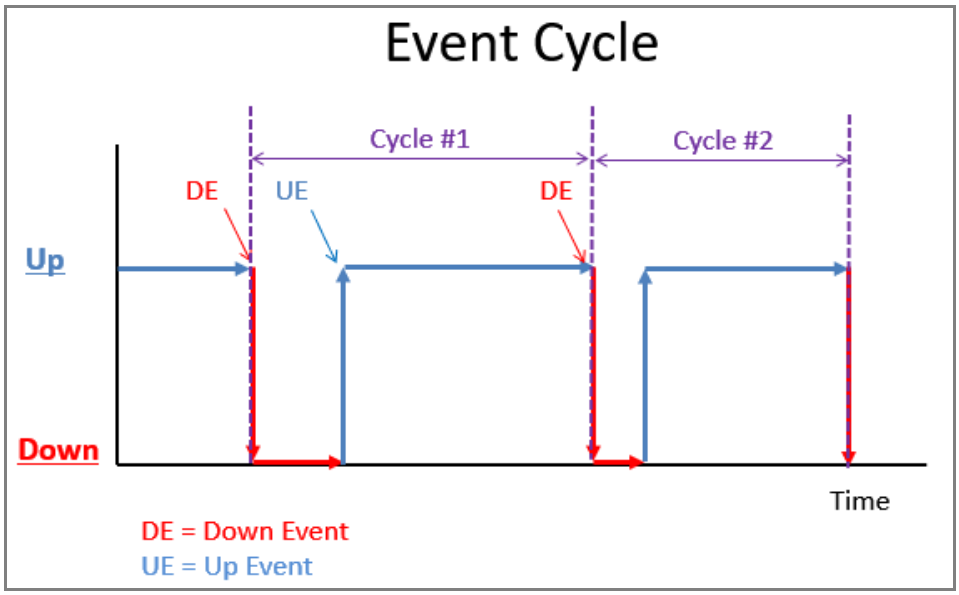
An Up event (UE) is the point in time when a resource becomes available. By default, ExtendSim assumes that all resources are available at the start of the simulation.

A Down event (DE) is the point in time when a resource becomes unavailable. Down events are either scheduled or unscheduled and can be due to failure, off-shifting, maintenance, or repair.



Event cycles

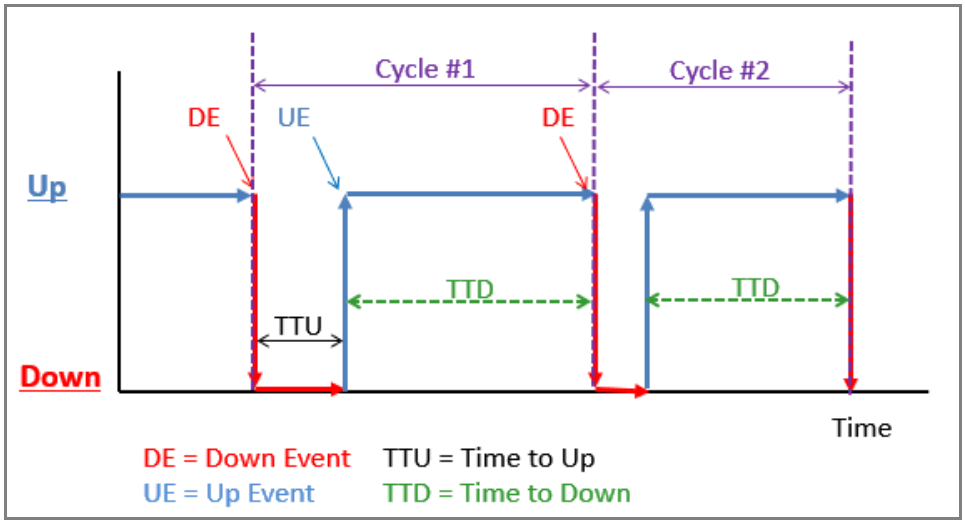
An event cycle describes how a resource alternately cycles through its up and down states over time. As shown below, this event cycle has multiple up and down cycles.



TTU's and TTD's

As seen in the graphic below:

- There is a period of down time that starts with a Down event (DE) and ends with an Up event (UE). The time between those two events is determined by a distribution that specifies the Time-to-Up (TTU).
- There is a period of up time that starts with an Up event and ends with a Down event. The time between these two events is determined by a distribution that specifies the Time-to-Down (TTD).


Cycle #1
Cycle #2
DE
UE
DE
Up
TTD
TTD
TTU
Down
Time
DE = Down Event
TTU = Time to Up
UE = Up Event
TTD = Time to Down

Index

Step 1 RBD Tutorial Final model 33

T-V

User Reference

How To chapters 8