

# Getting Started with ReThink

## Tutorials

Version 5.1 Rev. 1



Getting Started with ReThink Tutorials, Version 5.1 Rev. 1  
September 2014

The information in this publication is subject to change without notice and does not represent a commitment by Gensym Corporation.

Although this software has been extensively tested, Gensym cannot guarantee error-free performance in all applications. Accordingly, use of the software is at the customer's sole risk.

Copyright (c) 1985-2014 Gensym Corporation

All rights reserved. No part of this document may be reproduced, stored in a retrieval system, translated, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior written permission of Gensym Corporation.

Gensym®, G2®, Optegrity®, and ReThink® are registered trademarks of Gensym Corporation.

NeurOn-Line™, Dynamic Scheduling™ G2 Real-Time Expert System™, G2 ActiveXLink™, G2 BeanBuilder™, G2 CORBALink™, G2 Diagnostic Assistant™, G2 Gateway™, G2 GUIDE™, G2GL™, G2 JavaLink™, G2 ProTools™, GDA™, GFI™, GSITM, ICPTM, Integrity™, and SymCure™ are trademarks of Gensym Corporation.

Telewindows is a trademark or registered trademark of Microsoft Corporation in the United States and/or other countries. Telewindows is used by Gensym Corporation under license from owner.

This software is based in part on the work of the Independent JPEG Group.

Copyright (c) 1998-2002 Daniel Veillard. All Rights Reserved.

SCOR® is a registered trademark of PRTM.

License for Scintilla and SciTE, Copyright 1998-2003 by Neil Hodgson, All Rights Reserved.

This product includes software developed by the OpenSSL Project for use in the OpenSSL Toolkit (<http://www.openssl.org/>).

All other products or services mentioned in this document are identified by the trademarks or service marks of their respective companies or organizations, and Gensym Corporation disclaims any responsibility for specifying which marks are owned by which companies or organizations.

Gensym Corporation  
52 Second Avenue  
Burlington, MA 01803 USA  
Telephone: (781) 265-7100  
Fax: (781) 265-7101  
510

Part Number: DOC077-



# Contents

---

## Preface ix

About this Guide ix

Audience x

Conventions x

Related Documentation x

Customer Support Services xii

## Chapter 1 Introduction to ReThink 1

What is ReThink? 1

    Graphical Design Environment and Process Simulator 2

    Business Process Modeling 2

    Animation 2

    Automatic Performance Analysis 2

    Simulation Management 3

    Online Deployment 3

Benefits of Using ReThink 3

Process Modeling Paradigm 4

    Discrete Simulation Clock 6

    Work Objects 6

    Work Object Types 7

    Blocks 8

    Paths 9

    Resources 10

    Work Backups 11

    Inputs and Outputs 11

Required Knowledge for ReThink Users 12

    ReThink End Users 13

    ReThink Modelers 14

    ReThink Developers 16

## Chapter 2 Building a Simple Model 17

Introduction 18

Running ReThink 18

Running ReThink	<b>18</b>
Understanding ReThink Modules	<b>19</b>
Organizing ReThink Applications	<b>20</b>
Creating a Project	<b>20</b>
Creating a Model	<b>21</b>
Using the ReThink Toolbox	<b>22</b>
Using Scenarios	<b>24</b>
Creating a Scenario Tool	<b>24</b>
Activating a Scenario	<b>25</b>
Creating a Simple Model	<b>26</b>
Creating and Connecting Blocks	<b>26</b>
Labeling Blocks	<b>27</b>
Configuring Blocks	<b>28</b>
Configuring the Work Objects to Process	<b>28</b>
Configuring Duration	<b>29</b>
Saving the Model	<b>30</b>
Running the Simulation	<b>31</b>
Running the Simulation in Step Mode	<b>31</b>
Stepping Through the Process	<b>31</b>
Resetting the Simulation	<b>32</b>
Running the Simulation in Jump Mode	<b>33</b>
Opening an Existing Model	<b>33</b>
Exiting ReThink	<b>34</b>
<b>Chapter 3 Modeling a Complete Process</b>	<b>35</b>
Introduction	<b>35</b>
Opening the Model	<b>36</b>
Viewing the Model	<b>36</b>
Using Hierarchical Views in a Model	<b>38</b>
Creating Detail for a Task	<b>38</b>
Modeling the Order Processing Task	<b>39</b>
Viewing the Details of the Order Processing Task	<b>40</b>
Modeling the Payment Task	<b>42</b>
Viewing the Details of the Payment Task	<b>43</b>
Using Resources to Constrain the Model	<b>44</b>
Viewing the Clerks Resource	<b>44</b>
Viewing the Clerks Pool	<b>45</b>
Viewing Definitions for Work Objects and Resources	<b>46</b>

Observing the Process Under Normal Conditions	47
Running the Simulation	47
Analyzing the Process	50
Testing the Model Under High Capacity	51
Running the Simulation	51
Analyzing the Process	53
Automating the Payment Process	55
Running the Alternative Simulation	55
Analyzing the Process	57
<b>Chapter 4 Building Process Models</b>	<b>61</b>
Introduction	61
Organizing Models	62
Building Your Model	62
Project	62
Models	62
Scenario Tool	63
Organizer Tools	63
Model Detail	64
Model Definitions Organizer	64
Resources Organizer	64
Storage Pools Organizer	64
Viewing the Navigator	65
Saving the Model	65
Determining the Process and Building the Model	65
Identifying Metrics	66
Building the Top-Level Model	66
Modeling Task Details	67
Initial Task Detail	68
Adding Connection Posts to the Detail	68
Configuring the Detail	69
Modeling Resources	69
Creating Resource Pools	70
Creating Resource Managers	70
Configuring Duration	71
Configuring Costs	71
Annotating Your Model	72
Creating Work Objects	72
Computing Metrics and Obtaining Outputs	73
<b>Chapter 5 Turbine Blade Model</b>	<b>75</b>
Introduction	76

Opening the Model	76
Running the Simulation	78
Comparing Processes	78
Exploring the As Is Model	79
Exploring the High-Level View	79
Exploring the Engineering Design Detail	80
Creating Management Reports	81
Recording When Designs are Complete	81
Exploring the Engineering Analysis Detail	83
Determining Whether Problems Exist	84
Recording the Problem	85
Exploring the Manufacturing Detail	86
Summary	87
Exploring the Vision Model	87
Exploring the Engineering Design Detail	87
Exploring the Engineering Analysis Detail	89
Exploring the Manufacturing Design Detail	90
Observing the Resources	91
Viewing the Results	92
Organizing Versions	93
Using Customizations	95
Viewing the Object Hierarchy	96
Typical Questions and Answers	96

## **Glossary** 97

## **Index** 105

# Preface

---

*Describes this guide and the conventions that it uses.*

About this Guide	ix
Audience	x
Conventions	x
Related Documentation	x
Customer Support Services	xii



## About this Guide

This guide provides an introduction to the basic features of ReThink and basic modeling practices. The guide describes:

- The features and benefits of using ReThink, and the overall modeling paradigm that ReThink uses.
- How to build and run a simple model.
- The process you might go through to build a complete model of an existing process and to reengineer the process by creating a “what-if” scenario.
- The high-level modeling tasks that you need to perform when modeling using ReThink.
- A full-scale model of an engineering design and manufacturing process, which compares the results of an “As Is” model and a “Vision” model.

# Audience

This manual is written as an introduction to modeling using ReThink. For complete documentation of all ReThink features, see the *ReThink User's Guide*.

If you are a ReThink developer who wants to customize the definition of ReThink objects, see the *Customizing ReThink User's Guide*.

# Conventions

This tutorial uses the following typographic conventions:

Example	Description
true	Parameter and metric values
<b>task</b>	Glossary terms
<i>c:\Program Files\Gensym\g2-2011\rethink\kbs\rethink.kb</i>	Pathnames and filenames

# Related Documentation

## ReThink

- *Getting Started with ReThink*
- *ReThink User's Guide*
- *Customizing ReThink User's Guide*

## G2 Core Technology

- *G2 Bundle Release Notes*
- *Getting Started with G2 Tutorials*
- *G2 Reference Manual*
- *G2 Language Reference Card*
- *G2 Developer's Guide*
- *G2 System Procedures Reference Manual*
- *G2 System Procedures Reference Card*
- *G2 Class Reference Manual*

- *Telewindows User's Guide*
- *G2 Gateway Bridge Developer's Guide*

## **G2 Utilities**

- *G2 ProTools User's Guide*
- *G2 Foundation Resources User's Guide*
- *G2 Menu System User's Guide*
- *G2 XL Spreadsheet User's Guide*
- *G2 Dynamic Displays User's Guide*
- *G2 Developer's Interface User's Guide*
- *G2 OnLine Documentation Developer's Guide*
- *G2 OnLine Documentation User's Guide*
- *G2 GUIDE User's Guide*
- *G2 GUIDE/UIL Procedures Reference Manual*

## **G2 Developers' Utilities**

- *Business Process Management System Users' Guide*
- *Business Rules Management System User's Guide*
- *G2 Reporting Engine User's Guide*
- *G2 Web User's Guide*
- *G2 Event and Data Processing User's Guide*
- *G2 Run-Time Library User's Guide*
- *G2 Event Manager User's Guide*
- *G2 Dialog Utility User's Guide*
- *G2 Data Source Manager User's Guide*
- *G2 Data Point Manager User's Guide*
- *G2 Engineering Unit Conversion User's Guide*
- *G2 Error Handling Foundation User's Guide*
- *G2 Relation Browser User's Guide*

## **Bridges and External Systems**

- *G2 ActiveXLink User's Guide*
- *G2 CORBALink User's Guide*

- *G2 Database Bridge User's Guide*
- *G2-ODBC Bridge Release Notes*
- *G2-Oracle Bridge Release Notes*
- *G2-Sybase Bridge Release Notes*
- *G2 JMail Bridge User's Guide*
- *G2 Java Socket Manager User's Guide*
- *G2 JMSLink User's Guide*
- *G2 OPCLink User's Guide*
- *G2-PI Bridge User's Guide*
- *G2-SNMP Bridge User's Guide*
- *G2-HLA Bridge User's Guide*
- *G2 WebLink User's Guide*

## **G2 JavaLink**

- *G2 JavaLink User's Guide*
- *G2 DownloadInterfaces User's Guide*
- *G2 Bean Builder User's Guide*

## **G2 Diagnostic Assistant**

- *GDA User's Guide*
- *GDA Reference Manual*
- *GDA API Reference*

# **Customer Support Services**

You can obtain help with this or any Gensym product from Gensym Customer Support. Help is available online, by telephone, by fax, and by email.

### **To obtain customer support online:**

➔ Access G2 HelpLink at [www.gensym-support.com](http://www.gensym-support.com).

You will be asked to log in to an existing account or create a new account if necessary. G2 HelpLink allows you to:

- Register your question with Customer Support by creating an Issue.
- Query, link to, and review existing issues.

- Share issues with other users in your group.
- Query for Bugs, Suggestions, and Resolutions.

**To obtain customer support by telephone, fax, or email:**

→ Use the following numbers and addresses:

	Americas	Europe, Middle-East, Africa (EMEA)
<b>Phone</b>	(781) 265-7301	+31-71-5682622
<b>Fax</b>	(781) 265-7255	+31-71-5682621
<b>Email</b>	<a href="mailto:service@gensym.com">service@gensym.com</a>	<a href="mailto:service-ema@gensym.com">service-ema@gensym.com</a>



# Introduction to ReThink

---

*Provides an overview of ReThink, describing its benefits, users, basic components, and requirements.*

What is ReThink? 1

Benefits of Using ReThink 3

Process Modeling Paradigm 4

Required Knowledge for ReThink Users 12



## What is ReThink?

ReThink is a graphical simulation, analysis, and automation tool that enables decision-makers within complex organizations to visualize how their business process works, to analyze its performance, and to deploy it in real time.

Once the current process is understood, decision-makers can experiment with alternative work-flow models to redesign the business process in order to:

- Reduce costs.
- Reduce time to market.
- Improve quality.
- Enforce standards.

ReThink has numerous features including process modeling, workflow analysis, "what-if" analysis, simulation management, and online deployment.

# **Graphical Design Environment and Process Simulator**

ReThink supports interactive business process design by using discrete event simulation modeling techniques. It provides a graphical environment for designing business processes and an object-oriented simulator for testing those designs, as well as for measuring cycle time, cost, and other metrics.

## **Business Process Modeling**

You can model any type of business process, using ReThink. For example, you can use ReThink to model and analyze the following types of business processes:

- Order fulfillment processing.
- Manufacturing and distribution.
- Sales cycle and performance.
- Research and development cycle.
- Purchasing and accounts receivable.

## **Animation**

ReThink automatically animates your process designs, making it easy for you to visualize the workings of any complex business process. You can also turn animation off to speed up the models when all you need to see are the outputs.

## **Automatic Performance Analysis**

ReThink automatically measures the performance of proposed business process designs. With it, you can easily experiment with alternative structures, determining how each will impact cost and cycle time. It includes a wide range of tools for “**what-if**” analysis. For example, you can quickly determine how adding resources in key areas will affect product delivery time.

## Simulation Management

You can run multiple simulations sequentially, using a script. You use this feature to compare different simulation runs, using different input parameters. You can also use this feature to change parameter values during a simulation, for example, to model improved manufacturing times or variable consumer demand.

## Online Deployment

Once you have created a model of your business process, you can deploy that model in real time to support online transaction processing and decision support. In online deployment mode, ReThink works as a workflow engine, managing decisions and coordinating activities.

## Benefits of Using ReThink

ReThink offers numerous benefits to your business, including:

- Animation and interactive graphics to allow modelers to capture and communicate effectively the dynamics of your current business process.
- “What-if” analysis to enable managers to evaluate and understand process behavior and performance under different sets of conditions.
- Top-down modeling to provide the high-level view all the way down to the details.
- Object-oriented representation to model the various aspects of your business process, which you can then use to help implement the information systems of your reengineered process.
- Real-time, client/server environment to allow users to put a business process model online where it can continue to help manage business processes.

# Process Modeling Paradigm

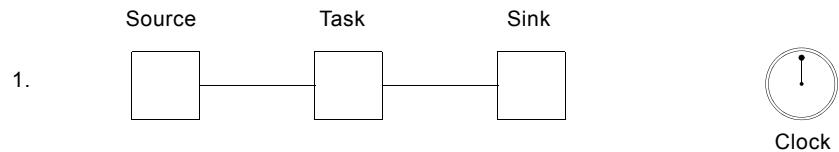
ReThink uses **discrete event simulation** as the basic paradigm to perform process modeling. Each change in the system that occurs in real time is called a discrete event. Each discrete event has a start event and a stop event. The discrete event simulation clock advances with each new stop event. The amount by which the clock advances represents the **work time** of each task in the process model.

You represent value-added tasks in ReThink by using **blocks**. A ReThink model consists of a set of connected blocks, which create and process work objects. Work objects flow from block to block on directed **paths**. Work objects and blocks accumulate performance statistics at each point in the process.

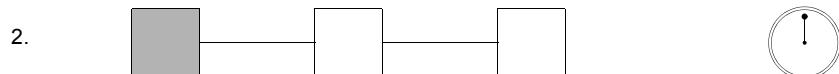
When the appropriate work objects arrive at a block, the block becomes active, which marks the beginning of the start event. The block then performs its particular operation on its input work objects and passes them to the next block in the process. After the block performs its tasks, ReThink advances the simulation clock, which marks the end of the stop event. Thus, a **task** represents the set of events that occur between the start and stop events, where each individual event is called an **activity**.

A special type of task called a source creates work objects at a given rate. Another type of task called a sink deletes work objects at the end of the process. The process continues indefinitely until you stop the simulation or until no more work objects remain to be processed.

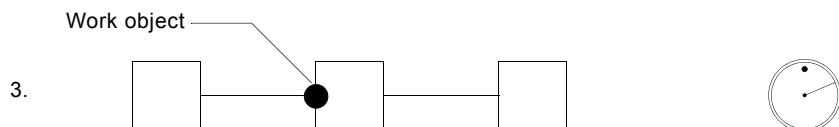
This figure shows several steps in a simple process model, which contains a source, a task, and a sink. The model shows the work objects that the blocks create, process, and delete. A shaded task in the figure means the task is actively processing work. The figure also shows a representation of a simulation clock as it advances with each new discrete event.



Simulation clock and tasks are idle.



When you start the simulation running, the source becomes active, which marks the beginning of the first discrete event.



During the second discrete event, the clock advances, and the source creates a work object and passes it to the downstream task.



The next event begins when the work object arrives at the task block.



When the task completes, the clock advances and the work object passes to the sink.

The following sections describe each aspect of the modeling paradigm, as well as some additional features of a ReThink model.

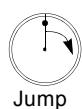
## Discrete Simulation Clock



The heart of ReThink is its discrete event simulation clock, which you control by using a **scenario**.

You can run the simulation clock continuously, or you can pause the simulation clock with each new discrete event. You can also run the simulation in a synchronized, time-scaled mode so you can visualize the delays in the process. If you have licensed ReThink Online, you can run the simulation in real time, based on the wall clock, to perform online transaction processing. These modes are called **jump**, **step**, **synch**, and **online mode**, respectively.

Any task can have a duration, which represents the amount of **simulation time** that it takes to process the task's input work objects. The clock keeps track of the current simulation time by advancing each time a task completes.



Jump

You can run the model so that the clock advances continuously with each new discrete event.



Step

You can run the model so that ReThink pauses the model at each discrete event.



Synch

You can run the model so that the clock advances continuously with each new discrete event, and the length of time between events is proportional to the duration of each event.



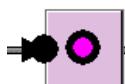
Online

You can run the model so that the simulation clock advances in real time for online transaction processing.

You model the duration of each event in seconds, minutes, hours, or days. The minimum time unit that the simulation clock can handle is one second.

For information on how to use ReThink scenarios, see [Using Scenarios](#).

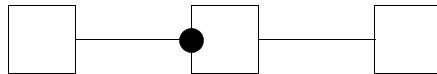
## Work Objects



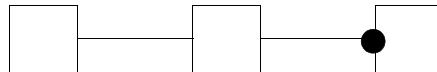
When a simulation is running, work objects flow on directed paths from one task to another in the process. **Work objects** represent physical or intangible objects that the model processes, for example, sales calls, widgets, boxes, or designs.

Tasks process work objects according to the requirements of the particular task.

Work objects keep track of the amount of time they have spent in the simulation and the total cost of all the tasks that have been applied to them.



The work object accumulates duration and cost from each task that processes it as it moves through the simulation.



At the end of the process, the work object knows how long it has been in the process, how long it was actively being processed, and how much it cost to process.

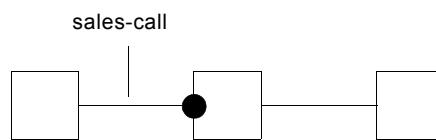
For information on how to specify the type of work a model processes, see [Configuring the Work Objects to Process](#).

## Work Object Types

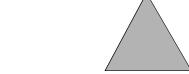


ReThink determines the type of work object that a block outputs by looking at the output path type specification of each block. When a work object arrives at a block, ReThink passes the object to the output path only if the output path type matches the work object's type. ReThink considers it a "match" if the output path type has the same class as or is a superior class of the input work object type. If the output path does not match the work object's type, ReThink deletes the input work object and creates a new work object whose type matches the output path type specification.

If ReThink does not already know about a particular class of work object, it creates a class definition during processing. You can extend the definition of a work object according to the needs of the process by editing the class definition of which the object is a type.

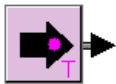


The source creates a work object of the type specified on its output path.



sales-call class definition

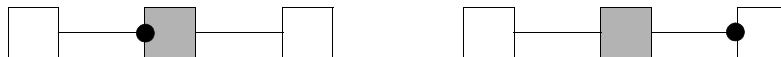
## Blocks



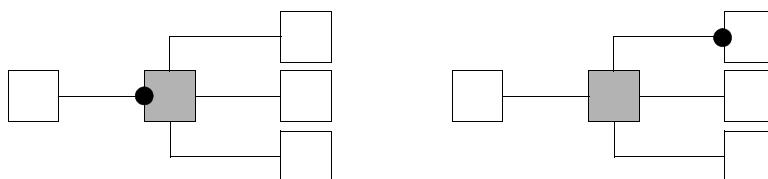
You represent tasks in ReThink with blocks. Blocks perform operations on objects, such as creating objects, applying value-added by adding simulation time or cost to objects, associating and reconciling objects, and deleting objects.

The most basic type of block is a Task block, which waits for all of its input work objects to arrive at the block before passing its outputs to the next block. Other blocks do not require all of their input work objects to arrive at the block before passing its outputs. For example, a block that merges multiple streams of work into a single stream does not wait for all of its inputs before passing its outputs. Most ReThink blocks perform specialized processing such as branching work objects based on some criteria, associating and reconciling work objects, or storing and retrieving work objects to and from a database, file, or resource pool.

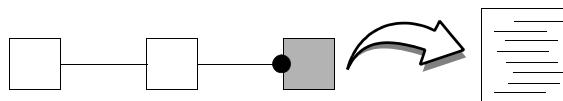
At the beginning of the process, a source block creates work objects at a rate you determine. At the end of the process, the model deletes the work objects, stores their data in an external database or file, or stores the work objects in a resource pool.



The most basic type of block simply passes work objects to the downstream block.



Another type of block branches work to different downstream blocks based on some criteria.



Other blocks store work object data to external files or databases for further processing.

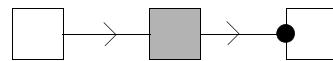
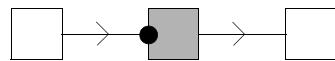
For information on how to create models with ReThink blocks, see [Creating a Simple Model](#).

For usage and reference information about ReThink blocks, see the *ReThink User's Guide*.

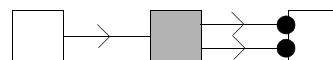
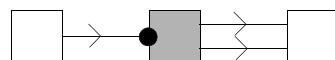
## Paths

With the exception of a Source block, each type of block has at least one input path. Depending on the type of block, the block either processes its input work objects immediately or waits until all of the inputs arrive at the block. When all of the necessary inputs have arrived at the block, the discrete event begins.

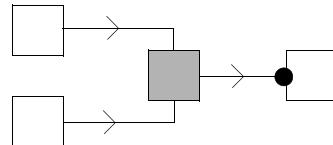
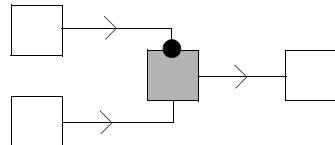
Most blocks have one or more directed output paths on which work objects flow when the discrete event completes. Some blocks have multiple output paths, which allow you to create or retrieve new work objects during the simulation. You can create feedback loops in a process by merging the output path of a downstream block into an upstream block.



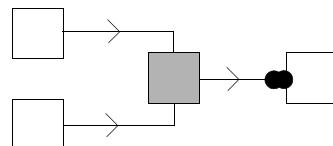
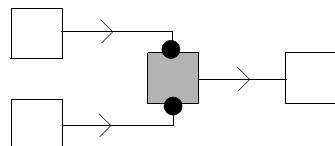
Some blocks have a single input and a single output path.



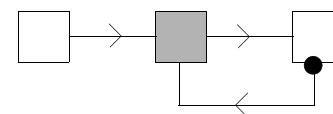
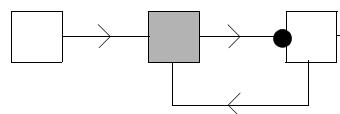
Other blocks have a single input and two output paths.



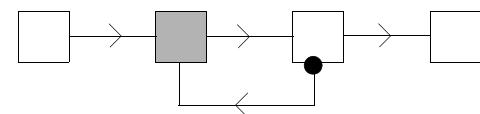
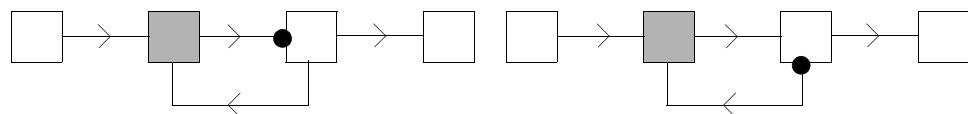
Some blocks require only a single input before processing.



Other blocks require multiple inputs before processing.



Some blocks require multiple inputs before processing.



Some blocks allow you to create feedback loops.

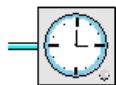
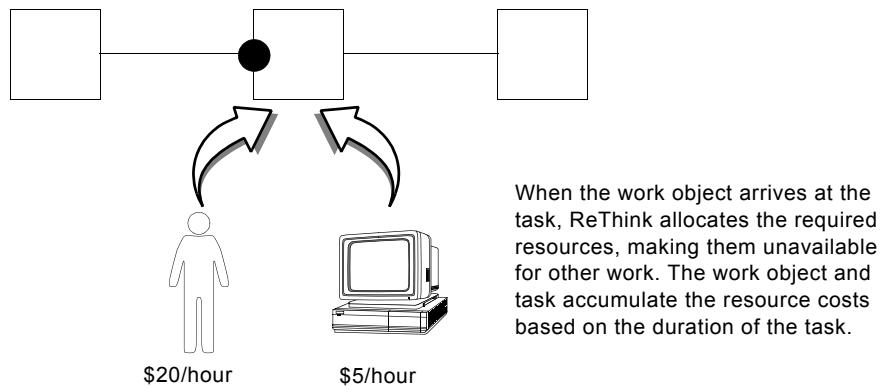
## Resources



A particular task can require one or more **resources**, which are physical or abstract objects that a task requires to process its work objects. For example, a filing task might require a clerk, a delivery task might require a driver and a truck, a loading task might require two loaders, and a financial task might require working capital.

When a work object arrives at a task and the discrete event begins, ReThink allocates the required resources to the task. When a resource is allocated, the task uses up the specified amount of the resource, making it unavailable for other tasks while it is allocated. When the task finishes, the resource is deallocated, making it available again for other tasks. Thus, resources constrain the model so that a particular task can simultaneously process only as many work objects as available resources exist.

You use resources to model costs in a process by applying fixed and variable costs to individual resources. As a task processes its work objects, ReThink applies the resource costs to the work objects, as well as to the task.



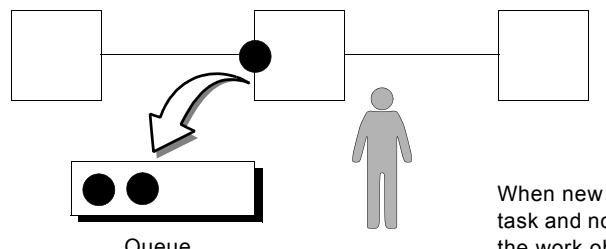
You can also use **temporal constraints** to specify the availability of resources on a monthly, weekly, and daily basis. You connect a constraint to resources in your model to constrain its availability.

For general information on applying constraints to the model with ReThink resources, see [Using Resources to Constrain the Model](#).

For specific information on constraining the model, using resources, and constraining the availability of resources, using ReThink constraints, see the *ReThink User's Guide*.

## Work Backups

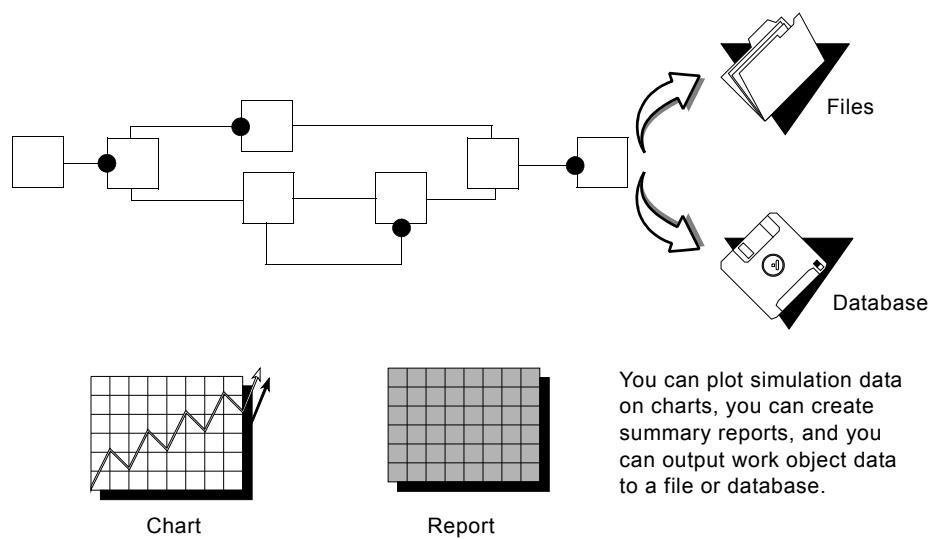
Work objects flow between blocks on paths, which can carry one or more work objects at a time. If a block is busy when a work object arrives, for example, because all of its resources are currently being used, ReThink places the work object in a **path queue** until the block is available. You can obtain statistics about how long work objects wait in the queue due to **work backups**.



When new work objects arrive at a task and no resources are available, the work objects back up on the input path and wait in a queue until a resource becomes available.

## Inputs and Outputs

Because a ReThink model processes objects, you can obtain a variety of information about the current simulation by querying the objects. For example, you can obtain statistics relating to the process time and total cost of work objects, or the utilization and total cost of a resource. Because the tasks themselves are also objects, you can obtain additional information about the total process time or total cost of particular tasks, or the number of discrete events themselves. You can chart these statistics, generate reports, or write report data to Excel spreadsheets, CSV files, or databases.



You can plot simulation data on charts, you can create summary reports, and you can output work object data to a file or database.



You obtain statistics from your model and feed key inputs into your model by using ReThink **instruments**. ReThink provides two kinds of instruments: **feeds** and **probes**.



You can also generate **reports**, which summarize all the statistics that ReThink computes for all types of ReThink objects. You can create reports for blocks, paths, probes, resources, and work objects. You can also create charts from those reports.

For detailed information using instruments and reports, see the *ReThink User's Guide*.

## Required Knowledge for ReThink Users

The three types of ReThink users are:

- **End users**, who apply knowledge about their business to existing ReThink models to obtain critical information about their business process.
- **Modelers**, who create graphical models of their business process by cloning ReThink objects from a palette, configuring the objects for their specific needs, and connecting these objects together to create a running model.
- **Developers**, who customize the definition of ReThink objects to perform tasks that are specific to a particular industry or business and that are not part of the basic tool set provided by ReThink. ReThink developers use G2 to customize ReThink.

ReThink runs as a layered application product on top of G2, Gensym's graphical object-oriented environment for building and deploying intelligent real-time systems. ReThink provides all the power of G2's discrete event simulation capability without requiring you to be a G2 expert. You create simulation models directly in ReThink, using ReThink's own discrete event simulation clock, which is separate from G2.

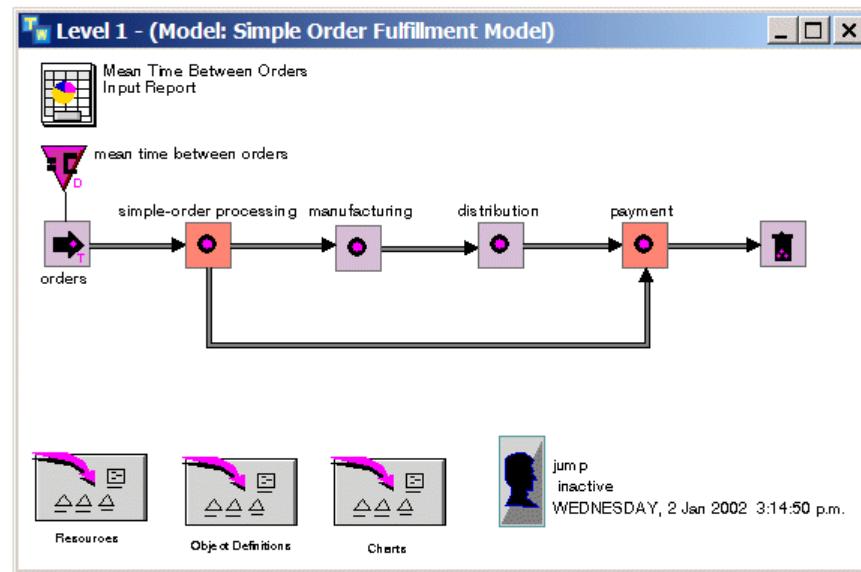
Because ReThink runs within the G2 environment, however, ReThink requires differing amounts of G2 knowledge, depending on the type of user.

## ReThink End Users

ReThink users need minimum information about the ReThink operating environment to run existing simulations. Specifically, ReThink end users need to know how to:

- Navigate the various workspaces that comprise the model.
- Control the status of a running scenario, for example, whether it is running or paused.
- View static information about the model, such as the duration of activities, path probabilities, and so on.
- View dynamic information about the model during or after a simulation, such as the utilization of resources or the waiting time of activities.

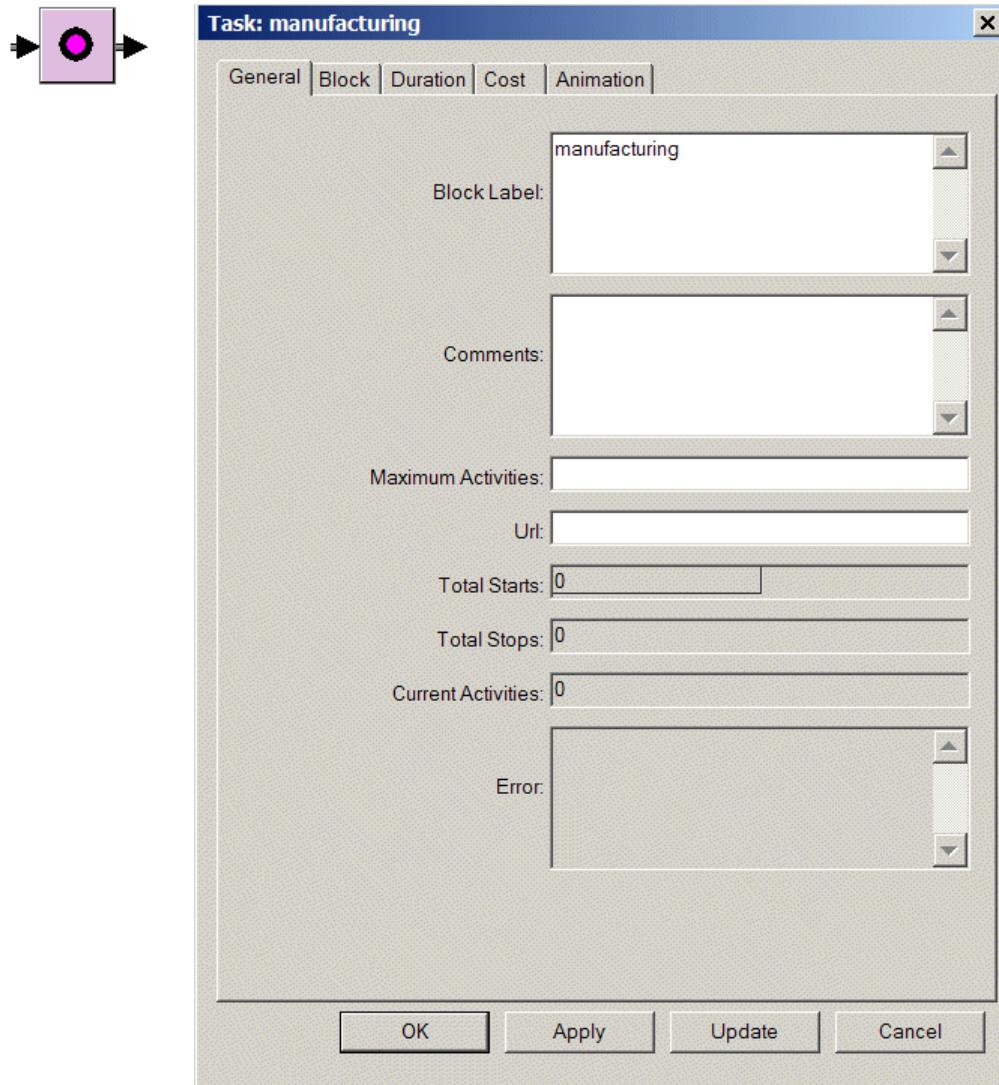
End users interact with graphical models that consist of blocks, instruments, reports, and various types of ReThink tools. Here is a typical model that an end user might see:



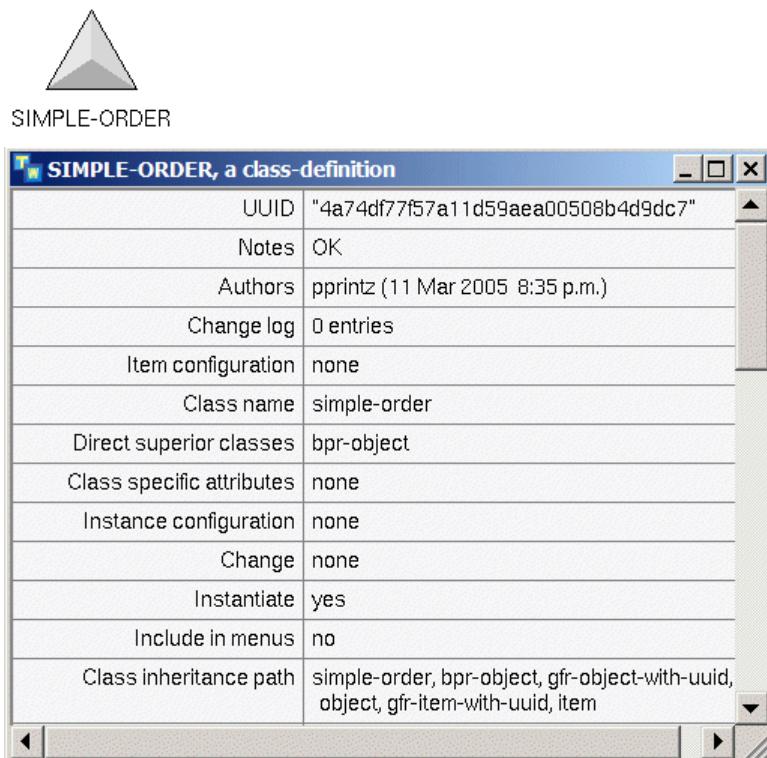
## ReThink Modelers

ReThink modelers need to know how to use all of ReThink's features.

Modelers can use properties dialogs or input reports to configure objects in a model. Here is the General tab of the properties dialog for a ReThink Task block:



ReThink modelers create **class definitions**, which inherit from built-in ReThink classes. These class definitions form the basis of information systems that you use to implement your models. ReThink creates **instances** of these classes when you run the model. Here is a class definition for a ReThink work object and its associated table:



## ReThink Developers

ReThink developers use G2's procedural programming language and subclassing features to customize ReThink objects. In writing custom methods and procedures, they need to know something about ReThink's internal processing and how to use ReThink's built-in system procedures or API.

Here is a method that the ReThink developer can customize to control the default behavior of a block

bpr-task::bpr-stop-method



Text Editor for the text of BPR-TASK::BPR-STOP-METHOD, a method

```
1 bpr-stop-method(task: class bpr-task, activity: class bpr-activity, ui-client-item: class ui-client-item)
2 object: class bpr-object;
3 path: class bpr-path;
4 counter: integer;
5 source-object: class bpr-object;
6 scenario: class bpr-scenario;
7 begin
8 {
9 This method defines the default stop behavior for the Task block. The Task block first tries to post all of its input objects on the output paths of the block. It then creates and posts objects for all of the output paths which do not already have an input object posted. Optionally it may also copy attributes from the input object to newly created output objects.
10 }
```

# Building a Simple Model

---

*Teaches the basic features of ReThink by using a step-by-step tutorial that loads ReThink and creates a simple model of a billing process.*

- Introduction **18**
- Running ReThink **18**
- Organizing ReThink Applications **20**
- Using the ReThink Toolbox **22**
- Using Scenarios **24**
- Creating a Simple Model **26**
- Configuring Blocks **28**
- Saving the Model **30**
- Running the Simulation **31**
- Opening an Existing Model **33**
- Exiting ReThink **34**



# Introduction

This chapter presents a step-by-step tutorial that teaches the basic operating environment of ReThink. You will create a simple model of a billing process, which generates orders and processes invoices.

In this tutorial, you will learn how to:

- Run ReThink.
- Use the ReThink menus.
- Organize your model, using Model tools and Scenario tools.
- Create and activate a Scenario tool.
- Create and connect blocks to create a simple model of a billing process.
- Configure the type of work the model processes and the timing of events.
- Save and load the model.
- Run the model to observe the simulation.
- Exit ReThink.

## Running ReThink

ReThink is a client/server application, where the server runs as a hidden process. To run ReThink, first you must start the server, then you can start the client. The server and client can be running on the same computer or on different computers.

ReThink is distributed as a set of modularized KB files, with a top-level module that automatically requires all the ReThink modules. The top-level module is called `rethink-online`.

When you first start ReThink, it is running in Modeler mode, which means that all ReThink functionality is available to you. To access additional functionality required to customize ReThink, you must switch to Developer mode. For information about switching user modes, see the *ReThink User's Guide*.

## Running ReThink

### To run ReThink:

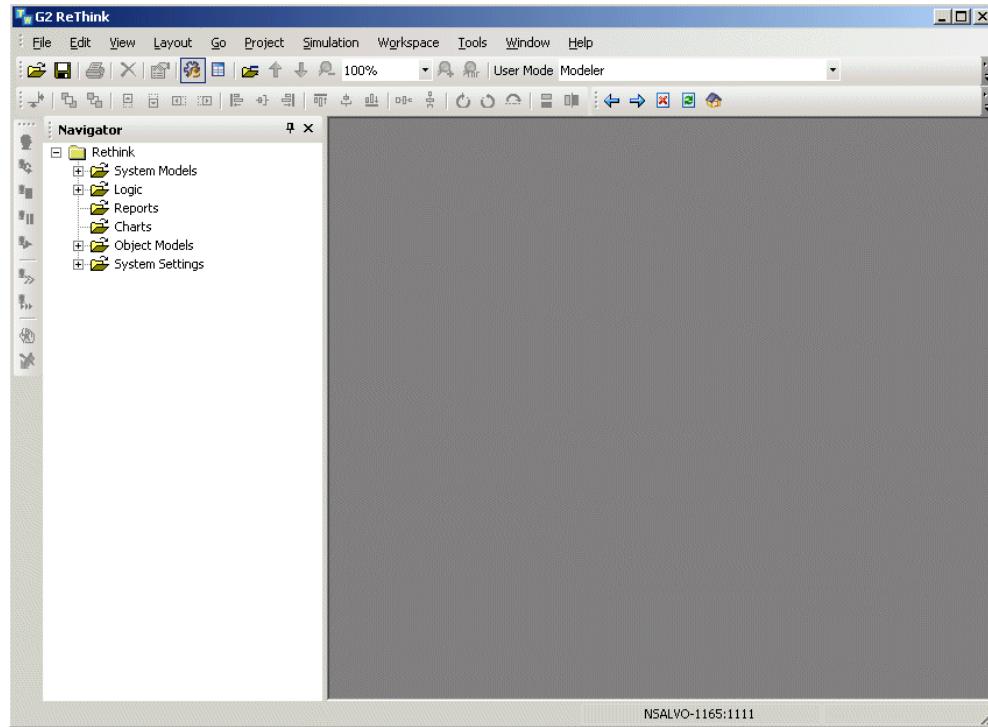
- 1 Choose Start > Programs > Gensym G2 2011 > G2 ReThink > Start G2 ReThink Server.

This launches the server on the local machine.

- 2 Choose Start > Programs > Gensym G2 2011 > Telewindows Next Generation.

This launches the Telewindows client on the local machine and connects to the server.

When ReThink has been loaded, you will see this window:



For more information about running ReThink, see [Running ReThink](#) in the *ReThink User's Guide*.

## Understanding ReThink Modules

Every ReThink application consists of one or more files, each of which typically contains a single module with the same name as the file. A **module** is a set of related knowledge in an application. A ReThink application is also called a **knowledge base** or **KB**; thus, the files that make up an application all have the `.kb` extension.

When you run ReThink, you are actually loading a number of related KB files and modules. ReThink contains numerous KB files, each of which contains a module of the same name as the file. The top-level module is called `rethink-online.kb`.

When you create a ReThink project, ReThink creates a new top-level module and associated KB file with the name you specify.

For more information about modules, see:

- [Saving the Model](#) and [Opening an Existing Model](#).
- [Working with Models](#) in the *ReThink User's Guide* for information about displaying the module hierarchy and merging, creating, renaming, and deleting modules.

## Organizing ReThink Applications

In this tutorial, you will create a model of a simple billing process, which creates and processes orders, generates invoices from those orders, then deletes the invoices. To create this model, you will clone, configure, and connect blocks. To run the model and observe the process, you will need a Scenario tool.

Typically, you place the blocks and associated Scenario tool together to create a **stand-alone model**, which is a model that can run independently of any other models in your knowledge base.

### Creating a Project

The first step in creating a ReThink model is to create a new project. ReThink creates and loads a new top-level module with the project name you assign. The project is saved as a `.kb` file in the `projects` directory of your ReThink installation directory.

#### To create a project:

- 1 Choose File > New.
- 2 Choose the ReThink library.
- 3 Choose any other library your application requires for connecting to external systems.
- 4 Enter the name of your project, for example, `getting-started`.

The project name cannot contain spaces.

- 5 Click OK.

ReThink creates a new project and associated `.kb` file with the name you specify, then loads it. You must wait until you see the top-level menu bar before continuing.

## Creating a Model

You create a business process **model** to contain your ReThink application. You place your model on the **detail**, which is an unnamed workspace associated with the model.

You can create as many ReThink models as you need. For example, you might create an “As Is” model, which represents your current process, and a “To Be” model, which represents the process after it has been reengineered.

### To create a model:

**1** Do one of the following:

- Choose Project > System Models > Business Processes > Manage, and click the New button.  
or
- In the Navigator, expand System Models in the tree, mouse right on the Business Processes node, and choose New Instance.

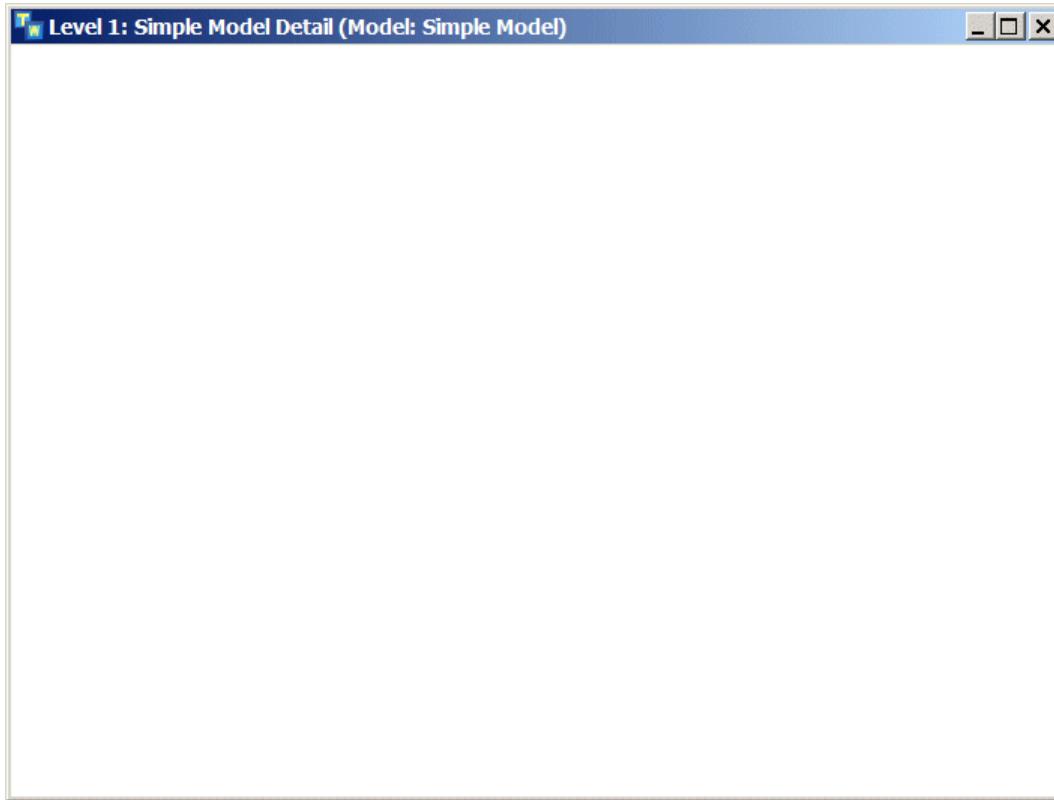
**2** In the dialog that appears, configure the Label for the model, for example, Simple Model.

The model appears in the Manage dialog and Navigator.

**3** Select the model in the Manage dialog and click the Model button or mouse right on the model in the Navigator and choose Show Detail.

**4** If the Manage dialog is open, close it.

This is where you will create your model:



## Using the ReThink Toolbox

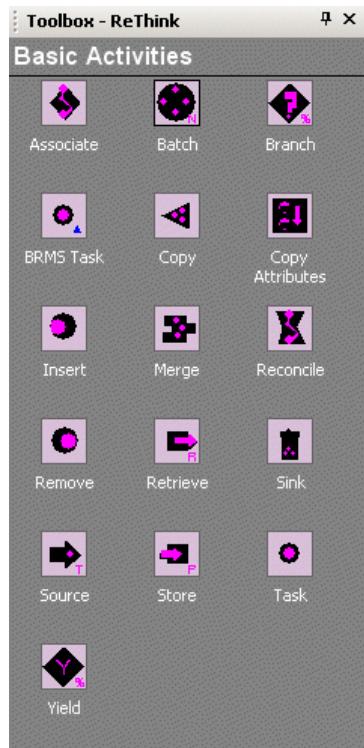
You create objects in a ReThink model by cloning objects from the various palettes in the ReThink toolbox. In this tutorial, you will use these palettes in the ReThink toolbox:

- The **Tools** palette contains a number of tools for organizing your model and running the simulation, the most important of which is the Scenario tool, which controls the simulation clock, the simulation mode, and the status of the simulation. The Model and Organizer tools help you organize the different components of your model.
- The **Basic Activities** palette contains all the blocks that you need to create ReThink models. Most blocks have input and output **stubs**, which you use to connect to other blocks. You can also add and delete stubs on a block once you clone it from the palette.

**To display the ReThink toolbox:**

- 1 Choose View > Toolbox - ReThink.

The Basic Activities palette appears, by default:



To display the various palettes of the ReThink toolbox, click the buttons at the bottom of the toolbox.

- 2** Click the Tools button (right-most button) to display the Tools palette:



## Using Scenarios

The heart of ReThink is its discrete event simulation engine, which tracks events by using a simulation clock. You control this clock by using a ReThink Scenario tool.

A Scenario lets you control and display various aspects of an individual model, including:

- The status of a model, that is, whether it is running, stopped, or paused.
- The mode in which the model is running, for example, continuously or step-by-step.

The two modes that you will use are called:

- Jump mode, which runs the model continuously.
- Step mode, which runs the model step-by-step.

## Creating a Scenario Tool

You typically place the Scenario tool on the model detail to control a single model. However, keep in mind that a single model might consist of one or more Model tools, where each model detail might consist of several levels of detail. Thus, a Scenario tool controls the model on the current workspace, as well as any models on a detail of the current workspace.

**To create a Scenario tool:**

- Display the Tools palette of the ReThink toolbox, click a Scenario tool, and place it on your model detail.

The Scenario tool has two attribute value displays: the current mode (**jump**), the current status (**inactive**). When the simulation is running, it also shows the current simulation time.



## Activating a Scenario

By default, the scenario is inactive. To run a simulation, the scenario must be active.

---

**Note** In general, it is not necessary to activate a scenario explicitly; choosing Start All or Reset automatically activate the scenario before performing the action.

---

**To activate a scenario:**

- Choose Simulation > Activate or click the equivalent toolbar button.

The Scenario tool icon changes to blue, and the status of the scenario changes to **stopped**, indicating that it is active but not currently running:



# Creating a Simple Model

The first step in creating a model is to create blocks from the toolbar and connect them together. In this part of the tutorial, you will create and connect blocks to model a simple billing process.

## Creating and Connecting Blocks

All ReThink blocks are located in the Basic Activities palette of the ReThink toolbox. You connect the blocks together by using the stubs attached to the blocks.

The graphical objects in the Basic Activities palette are fundamental building blocks for modeling any process. They are domain independent; you can use them in a variety of industries. You can also customize their appearance and behavior to reflect your particular business or market.

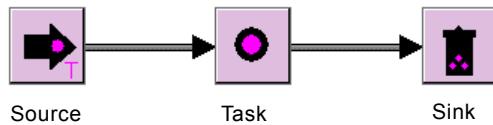
In this exercise, you will create a very simple model of a billing process, which uses a Source block, a Task block, and a Sink block:

- A **Source block** creates work objects at the beginning of a process.
- A **Task block** represents any activity that processes work objects in a model.
- A **Sink block** signals the end of a process by deleting the work objects it receives.

### To create and connect blocks:

- 1 Display the detail of the Simple Model.  
The Simple Model should already have a Scenario tool on its detail.
- 2 Display the Basic Activities palette of the ReThink toolbox.
- 3 Create a Source block and place it on the left side of the Simple Model detail.
- 4 Create a Task block and place it directly to the right of the Source block.
- 5 Create a Sink block and place it to the right of the Task block.
- 6 Select all the blocks by dragging the mouse around the objects or by selecting an object and adding to the selection by using the Shift key.
- 7 Choose Layout > Align or Distribute > Distribute Horizontally to space the objects equally or click the equivalent toolbar button.
- 8 With the objects still selected, choose Layout > Align or Distribute > Align Tops or click the equivalent toolbar button.
- 9 To connect the three blocks together, click the output stub connected to a block to attach it to the mouse, move the stub to the input stub of another block, and click to connect the stubs.

Your model should look like this:



## Labeling Blocks

You can label each block as to its specific function in the model. You can move a label by dragging it to a new location. The label moves with the block. You can also hide the label.

### To configure the label for the Task block:

- 1 Choose Properties on the Task block to display its properties, double-click the block, or press F4 with the block selected.

Each block has a properties dialog with various tabs for configuring parameters and viewing metrics, as follows:

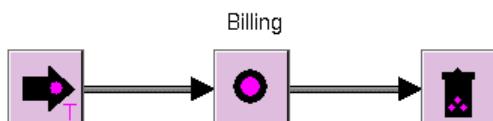
- The General tab contains attributes common to all blocks.
- The Block tab contains block-specific attributes.
- The Duration tab contains parameters and metrics related to the timing of activities of the block.
- The Cost tab contains parameters and metrics related to fixed and variable costs.
- The Animation tab contains parameters for customizing block colors.

- 2 On the General tab, configure the Block Label to be Billing.

The attribute display appears near the Task block.

- 3 Drag the label to just above the block.

Here is the simple model with a labeled Task block:



# Configuring Blocks

Now, you will configure the blocks you just created to model a simple billing process, which involves two basic steps:

- Configuring the work objects that the block will create and process.
- Configuring the timing of events in the process.

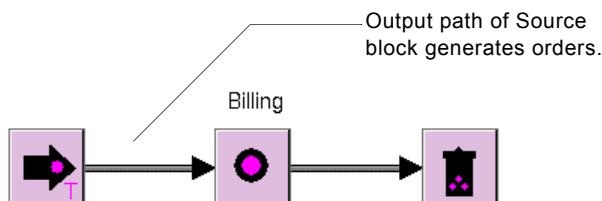
Using our example, the Source block will create orders once an hour, and the Task block will take three hours to process those orders and produce invoices.

## Configuring the Work Objects to Process

A ReThink model creates and processes **work objects**. You specify the type of work object that a block processes by configuring the Type attribute of the block's output path(s), or its **path type**. You configure the block's path type by configuring the path between two connected blocks.

The default output path type is **bpr-object**, which is a built-in ReThink class that represents a work object. Your particular model can create and process specific types of work objects, which are subclasses of **bpr-object**. A **subclass** is a class of object that inherits its definition from another object.

In the model you have created so far, you will configure the path type of the Source block so that it generates orders:



### To configure the type of work that a block processes:

- 1 Choose Properties on the Source block's output path or double-click the output path.

The Type attribute of a path determines the type of work object the block processes. The default value is **bpr-object**.

- 2 Edit the Type attribute to be **order**.

When the model is running, the Source block will generate orders and pass them to the Billing block.

- 3 Display the properties dialog for the Billing block's output path and edit its Type to be **invoice**.

When the simulation is running, the Billing block will delete the orders it receives and generate invoices. The Billing block will then pass the invoices to the Sink block to signal the end of processing.

The first time you run the simulation, ReThink automatically creates object definitions for the work objects named `order` and `invoice`. Thereafter, ReThink uses the existing definitions.

## Configuring Duration

Next, you must configure the frequency with which a Source block emits orders and the amount of time it takes for the Billing block to process invoices. To do this, you configure the **duration**, which represents the amount of simulation time that a particular event takes.

By default, most blocks use a random normal distribution to compute duration; by default, the Source block uses a random exponential distribution. You can configure blocks to use a variety of mathematical distributions to compute duration, including fixed, random exponential, and random triangular.

For an explanation of the available distributions, see [Using Blocks](#) in the *ReThink User's Guide*.

### To configure the duration of blocks in the model:

- 1 Display the properties dialog for the Source block and click the Duration tab.

This dialog contains several attributes related to the timing of the block. The default Distribution Mode for a Source block is **Random Exponential**, which causes the block to compute the delay by using a random exponential function.

- 2 Configure the Mean to be 1 hour, as follows: 000 000 01:00:00

The Source block will emit orders once an hour, using a random exponential distribution. This means that on average, the Source block will emit orders once an hour; however, 50% of the time the block will emit orders more frequently than once an hour, for example, every 30 minutes, and 50% of the time the block will emit orders less frequently than once an hour, for example, every 2 hours.

Occasionally, with a random exponential distribution, the block will emit orders significantly less frequently than once an hour, for example, once every 5 or 10 hours, depending on how far along the curve the sample point is.

Next, the Billing block will receive orders, process them for three hours, and produce an invoice for each order.

- 3 Display the properties dialog for the Billing task and click the Duration tab.

Notice that the default Distribution Mode for a Task block is **Random Normal**. A random normal duration type causes the block to compute the delay by

using a random normal function. When a block uses a random normal duration, you specify the mean time between events and an optional standard deviation. If you do not specify a deviation, the duration of the task is exactly the mean time you specify. You can also configure the Distribution Mode to use a fixed duration.

- 4 Configure the Mean to be 3 hours.
- 5 Configure the Standard Deviation to be 1 hour.

The duration of the Billing task has a mean time of three hours and a standard deviation of one hour. This means that, on average, the duration will be three hours, but it will vary from some amount less than three hours to some amount greater than three hours, based on a normal distribution.

## Saving the Model

Before you run the simulation, you should save the model. By default, when you save a model, ReThink saves it in the *projects* directory of your ReThink installation directory, using a *.kb* filename that corresponds to the name of the project you created.

Once you have saved your model, you can exit ReThink and open it again at any time.

For additional information on saving and opening models, see [Working with Models](#) in the *ReThink User's Guide*.

**To save your model:**

- ➔ Choose File > Save.

# Running the Simulation

You will now run the simulation in step mode to observe the behavior of the simple billing model. The model generates orders and processes the order into invoices. The model deletes the invoices at the end of the process.

## Running the Simulation in Step Mode

By default, the scenario is configured to run in jump mode, which advances the simulation clock continuously. First, you will run the simulation in step mode, which advances the simulation clock for each new event, then pauses the simulation.

### To run the simulation in step mode:

- 1 Choose Simulation > Step Mode or click the equivalent toolbar button.
- 2 Choose Simulation > Start All or click the equivalent toolbar button.

This button activates and resets the scenario, then starts all Source blocks associated with the scenario. You should see the start time for the simulation displayed next to the scenario. Also, the Source block should change to a different color indicating that it is now active.

---

**Tip** If you have multiple Source blocks associated with the scenario and you want to start them individually, choose Start on the Source block.

---

## Stepping Through the Process

Now, you will step through the process by advancing the clock to the start time of each sequential event.

### To step through the process:

- 1 Take a step in the process by choosing Simulation > Continue or clicking the equivalent toolbar button.

You should see a work object at the input of the Billing task. The work object is black, indicating that it is idle; it is not currently being processed.

Notice that a triangular icon also appears on the workspace. This icon represents a class definition for the order work object. ReThink automatically creates class definitions for work objects that the model processes when those definitions do not already exist. A class definition is an **object-oriented representation** of the type of information that an object contains.

Notice too that the clock now shows a new simulation time. Every time an event occurs that has a duration, the clock advances to the stop time of that

event. The amount of time that the clock advances depends on the duration of the event. In this case, the clock advances by a number that is a random exponential function of the mean time of the Source block, which is one hour.

**2** Choose Properties on the work object.

The title bar indicates that it is an order. The Source block has created a work object that is a type of order, which is the type you specified on the path. The source produces the order and passes it to the Billing block.

**3** Take several more steps through the process until the Source block has emitted a number of orders.

Because the Source block emits orders on average once an hour, and the Billing block takes three hours on average to process them, orders flow into the Billing block faster than invoices flow out. These orders stack up on top of each other on the input path of the Billing block, with the newest order on top.

When the Billing block is actually processing an order, the order object turns red, indicating it is being processed. When it is not being processed, it is black.

Notice that the simulation time has advanced by the duration of the events that have occurred.

**4** Continue stepping through the process until you see an invoice object on the input path of the Sink block.

ReThink automatically generates a class definition for the invoice object, just as it did for the order object.

**5** Display its properties dialog to verify its type.

ReThink has deleted the order and generated an invoice according to the output path type you specified for the Task block.

**6** Take several more steps through the process until the Sink block absorbs the invoice, indicating that the process is complete.

The Billing block continues to process orders and generate invoices.

## Resetting the Simulation

Now you will reset the model and run it again in jump mode.

**To reset the model:**

- ➔ Choose Simulation > Reset or click the equivalent toolbar button.

ReThink deletes all of the work objects that the model created and resets the simulation time of the scenario back to zero.

## Running the Simulation in Jump Mode

Now, you will observe the process by running the simulation in jump mode, which advances the simulation clock continuously with each new event.

### To run the simulation in jump mode:

- 1 Start the simulation running, as described in [Running the Simulation in Step Mode](#), except this time, first choose Simulation > Jump Mode or click the equivalent toolbar button.  
ReThink is creating, deleting, and transferring objects according to your specification in the blocks and the output paths.
- 2 Reset the simulation.

## Opening an Existing Model

Now that you have saved your model and run your simulation, you can exit ReThink and open the model again at any time. Opening an existing model replaces the current model and loads all of the required ReThink modules automatically.

Before you open an existing model, be sure you have saved the current model, as described in [Saving the Model](#).

### To load an existing ReThink model:

- 1 Choose File > Open.

---

**Note** Be sure you have saved your model before you load a new one, as described in [Saving the Model](#).

---

- 2 Select the *getting-started.kb* file located in the *projects* directory to open the top-level module you saved previously and all required modules.
- 3 Choose Project > System Models > Business Processes > Simple Model.

You should see the same model you just saved.

# Exiting ReThink

When you exit ReThink, you typically exit the server as well as the client. To prevent users from exiting the server by mistake, you can only exit the server in Developer mode. You can also exit the server from the icon in the system tray.

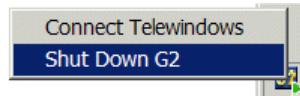
Before you exit, be sure you have saved the current model, as described in [Saving the Model](#).

**To exit ReThink and shut down the server:**

- 1** Choose Tools > User Mode > Developer.
- 2** Choose File > Exit.
- 3** Click Yes in the confirmation dialog that appears to shut down the client and the server.

**or**

- 1** Choose File > Close.
- 2** Choose Shut Down G2 on the G2 icon in the system tray:



# Modeling a Complete Process

---

*Provides a tutorial that compares the performance of a simple order fulfillment process model with an automated model that requires fewer resources.*

Introduction **35**

Opening the Model **36**

Viewing the Model **36**

Using Hierarchical Views in a Model **38**

Using Resources to Constrain the Model **44**

Viewing Definitions for Work Objects and Resources **46**

Observing the Process Under Normal Conditions **47**

Testing the Model Under High Capacity **51**

Automating the Payment Process **55**



## Introduction

This chapter provides you with a step-by-step tutorial that walks you through a model of a simple order fulfillment process. In this tutorial, you will learn how to:

- Create a high-level model of a process and model the details of various tasks.
- Model decision-making in a process by branching work.
- Associate and reconcile work that flows apart in a process.
- Constrain the model by using resources.

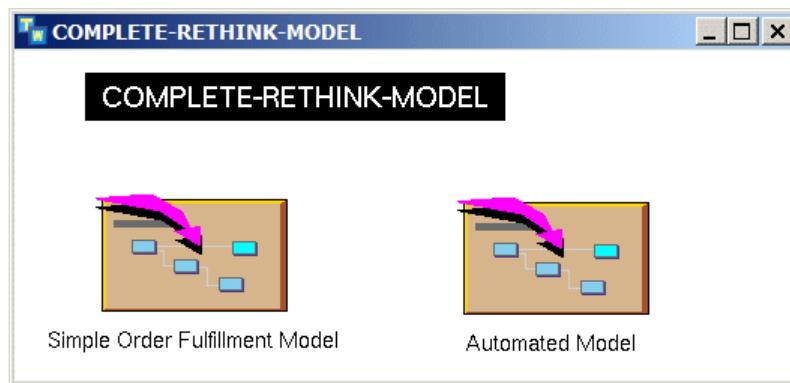
- Model database operations by storing objects in a resource pool.
- Use ReThink instruments to analyze key statistics in a model, such as the total cycle time and average utilization of resources.
- Supply different input parameters to the model to observe the process under a variety of conditions.
- Perform “what-if” analysis to experiment with different configurations of the model and to see the impact on performance.

## Opening the Model

Your ReThink product includes a KB file of a simple order fulfillment model, which you can open.

**To open this model:**

- 1 Choose File > Open and choose *chapter3.kb* in the *rethink\examples* directory of your ReThink installation directory.  
You can also choose Start > Programs > Gensym G2 2011 > Examples > G2 ReThink > Chapter 3 Model.
- 2 If the workspace is not already showing, choose Workspace > Get Workspace > complete-rethink-model to display this workspace:



First, you will look at the Simple Order Fulfillment model, then you will look at an alternative model for comparison, which automates the payment process.

## Viewing the Model

The simple order fulfillment model includes:

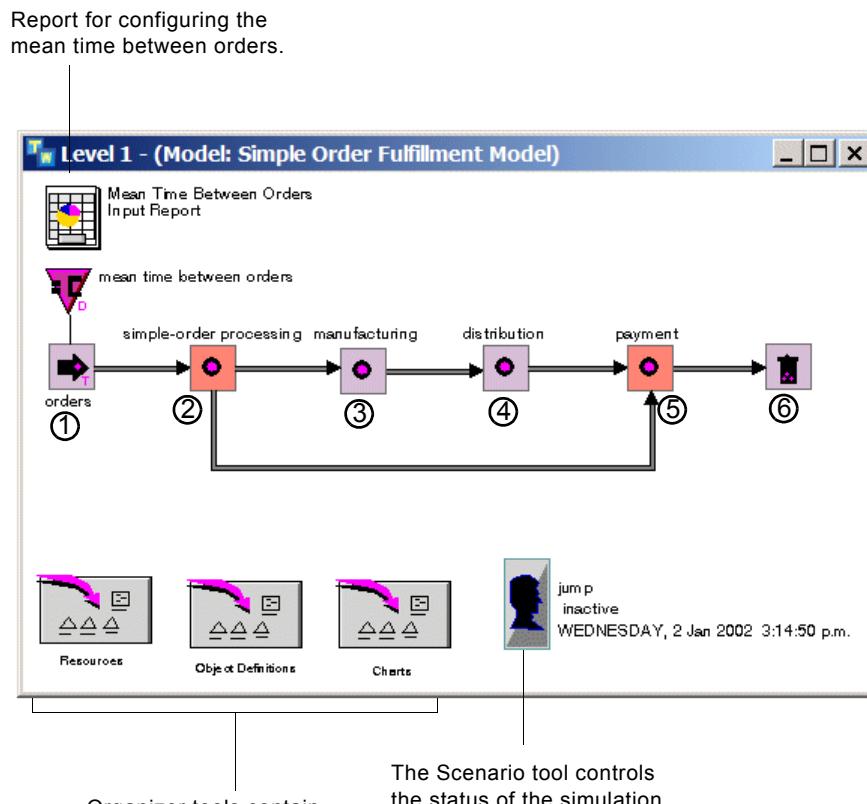
- Blocks that create, process, and delete orders and invoices.
- An instrument that supplies the model with the mean time between orders.

- A Scenario tool that controls various aspects of the model and displays performance statistics.
- Organizer tools that contain resources and class definitions that the model uses.

**To view the model:**

→ Choose Show Detail on the Simple Order Fulfillment Model.

ReThink displays this detail:



Organizer tools contain resources that constrain the model and class definitions for work objects.

The Scenario tool controls the status of the simulation and its run mode.

This model represents a high-level view of an order fulfillment process, which performs these tasks in this order:

- 1 Generates orders.
- 2 Processes orders and generates invoices.
- 3 Manufactures orders.
- 4 Distributes orders.

**5** Receives payment.

**6** Deletes orders.

You supply the mean time between orders as an input parameter to the model.

## Using Hierarchical Views in a Model

When modeling business processes, you typically start by modeling the high-level steps in the process to provide an overview. In the simple order fulfillment model, the high-level tasks include order processing, manufacturing, distribution, and payment.

Once you have a high-level view, you can break each macro step down into micro steps, which describe the details of each high-level process. You can refine the subtasks to as much level of detail as you desire, until you have created a complete model. In this way, you can create **hierarchical views** of a model.

Creating a model from the top-down provides these advantages:

- You can model separate, unrelated tasks independently and at different times.
- Different people can model different aspects of the process, applying their own expertise where needed.

In the order fulfillment model, for example, you might begin by modeling the order payment process to analyze the efficiency of the payment cycle. The Order Processing task might include performing a credit check on new customers and generating an invoice for all orders. Later, you can expand the model by adding details for the other top-level tasks.

### Creating Detail for a Task

To perform top-down modeling in ReThink, you create detail for a Task block, which is a subworkspace on which you place additional objects. The detail contains a copy of the top-level block, which is connected to the input and output paths of the top-level task through **connection posts**. You add blocks to the detail, as necessary, to model its details.

When you run the simulation, ReThink routes work objects to the detail for processing before passing them to the next high-level block. ReThink maintains summary statistics in the high-level task.

For more information on creating detail for a Task block, see [Modeling Task Details](#).

## Modeling the Order Processing Task

This section shows some typical questions you might ask the order processing manager to obtain information for your ReThink model. The responses determine the techniques you use for each step.

**Q When you receive an order from a customer, what do you do first?**

- A** First, we check to see if it is a new customer. If so, we do a credit check and register a new account. Then, we merge the orders for new customers with the orders for existing customers.

To model this, first, you need a Branch block with two output paths. The block branches orders based on whether the order is for a new or existing customer. For new customer orders, you need another Branch block, again with two output paths. This Branch block performs the credit check and either approves the customer for credit or not. If the customer is approved, you use a Task block to create an account. Then, you pass the new customer orders and the existing customer orders to a Merge block for prioritization.

**Q What percentage of the orders are for new accounts?**

- A** About one-quarter of the orders are from new customers.

The probability that the order will pass to the credit check approval process is 0.25, and the probability that the order will pass directly downstream is 0.75. You model this by specifying proportions on each output path of the Branch block.

**Q What percentage of the new orders get approved?**

- A** About two-thirds are approved.

The probability that a new order will pass to the Task block that creates an account is 0.7, and the probability that the new order will be lost is 0.3. You model this by specifying proportions on each output path of the Branch block that performs the credit check. You send the orders that are lost to a Sink block to delete them.

**Q What happens next?**

- A** Next, we make sure that all the order information has been entered on the order. If not, we have to send it back to the salesperson to get the missing information.

Here, we need another Branch block with two output paths that checks the order. One path sends the order to a rework block, which loops back into the Merge block for reprioritizing with the orders for new and existing customers. The other path sends the order directly downstream to the next task.

**Q How often do you need to send the order back for more information?**

- A** About two or three out of every 10 orders is missing information.

The probability that an order is passed to the rework loop is 0.25, and the probability that the new order is OK is 0.75.

**Q Now that the order has all the necessary information, what do you do?**

**A** We create an invoice for the order, which we identify with the order.

To model this, you use a Task block with two output paths, each with a different path type. One path passes the existing order downstream, and the other generates an invoice.

Next, you pass the order and the invoice into an Associate block, which associates the two work objects in the model. You identify the association by name in the block. You specify the path types for both the input and output paths of the Associate block. The order passes through the top connection post to the Manufacturing block, and the invoice passes through the bottom connection post directly to the Payment block.

**Q What resources do you require to process orders and generate invoices?**

**A** We have two clerks who create new accounts and generate invoices.

To model this, you use resources, which limit the number of orders that each task can process simultaneously. You create what is known as a “pool” of resources, which you place on the detail of the Resource organizer. You generate Resource Managers directly from the resource. You attach the managers to the blocks that require the resources.

For more information on modeling with resources and Resource Managers, see [Using Resources to Constrain the Model](#).

**Q Finally, about how long does each part of the process take?**

**A** To do a credit check and create a new account takes just about half an hour. Creating an invoice takes about twenty minutes, unless the computer is down, in which case it might take half an hour.

You specify the duration of the task that creates orders to be 30 minutes, with a deviation of 5 minutes. You specify the duration of the task that generates invoices to be 20 minutes, with a deviation of 10 minutes.

You are now finished modeling the details of the Order Processing task.

## **Viewing the Details of the Order Processing Task**

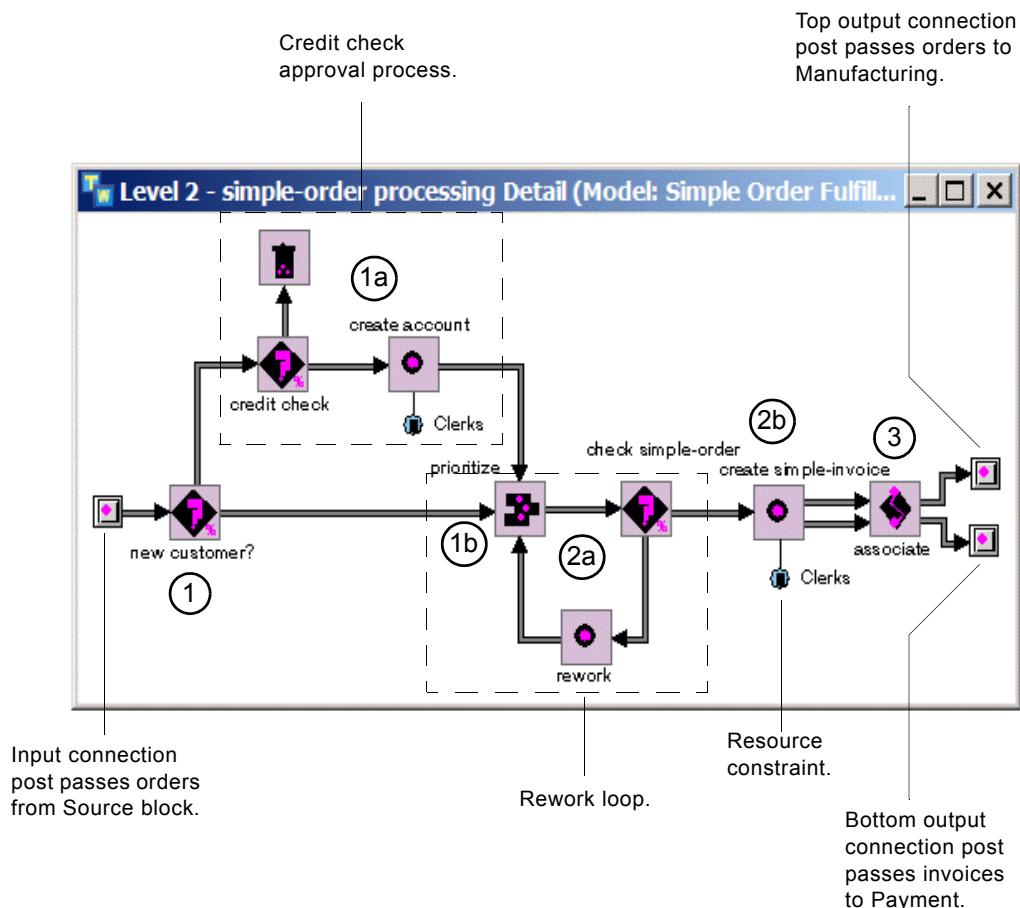
The Order Processing task’s detail contains:

- Blocks that model a credit check approval process, a rework loop, and an invoice creation process.
- Connection posts that receive orders from the previous block and pass orders and invoices downstream.
- Resource constraints that limit the number of new accounts that the model can create and the number of invoices that the model can generate simultaneously.

**To view the details of the Simple Order Processing task:**

→ Choose Show Detail on the Simple Order Processing task.

ReThink displays this detail:



The order processing detail performs these tasks in this order:

- 1 Checks to see if the order is from a new customer.
  - a If the order is from a new customer, performs a credit check on the customer.
    - If the credit check is not approved, deletes the order.
    - If the credit check is approved, creates an account for the customer, which requires a clerk resource, and sends the order downstream to be prioritized with existing customer orders.
  - b If the order is from an existing customer, sends the order directly downstream to be prioritized.

- 2** Checks both new and existing customer orders to see if all the necessary information exists.
  - a** If the order is missing information, sends the order through a rework loop, and reprioritizes the order.
  - b** If the order is OK, creates an invoice for the order, which requires a clerk resource.
- 3** Associates the order and the invoice and sends both objects downstream. The order goes to the Manufacturing task, and the invoice goes directly to the Payment task.

## Modeling the Payment Task

This section shows some typical questions you might ask the order processing manager to obtain information for your ReThink model. The responses determine the techniques you use for each step.

**Q** **What happens in your department when a customer makes an order?**

**A** Once the order has been delivered, we receive the order form and match it with the original invoice. Then, we record the invoice in our database.

To model this, you use a Reconcile block, which matches the orders that arrive from the Distribution task with invoices that arrive from the Order Processing task. The Reconcile block matches orders and invoices based on the association name you created in the Associate block on the Order Processing detail. You specify the input and output path types of the Reconcile block. The invoices then go to a Store block, which models the database operation.

**Q** **How long does it take to receive payment?**

**A** We take credit information over the phone with the order, so it usually only takes one business day for the funds to be transferred.

The reconciled orders pass to a Task block, which models the payment period. The duration of the task is 1 day.

**Q** **What happens then?**

**A** Right now, we are using the same clerks that process orders to process the payments. It only takes about 20 minutes to process the payment, once we have received it.

You create a Task block that processes the payment, which takes 20 minutes, with a deviation of 5 minutes. This task draws upon the same resources as the tasks on the Order Processing detail.

You have finished modeling the details of the Payment task.

## Viewing the Details of the Payment Task

The Payment task detail contains:

- **Blocks** that reconcile orders and invoices, store invoices in a database, model the payment period, and process the payment.
- **Connection posts** that receive orders and invoices, and process orders.
- **Resource constraint** that limits the number of payments for orders that the model can process at one time.
- **Resource pool** that models a database for storing invoices.
- **Instruments** that probe the model to compute cycle times for each order.

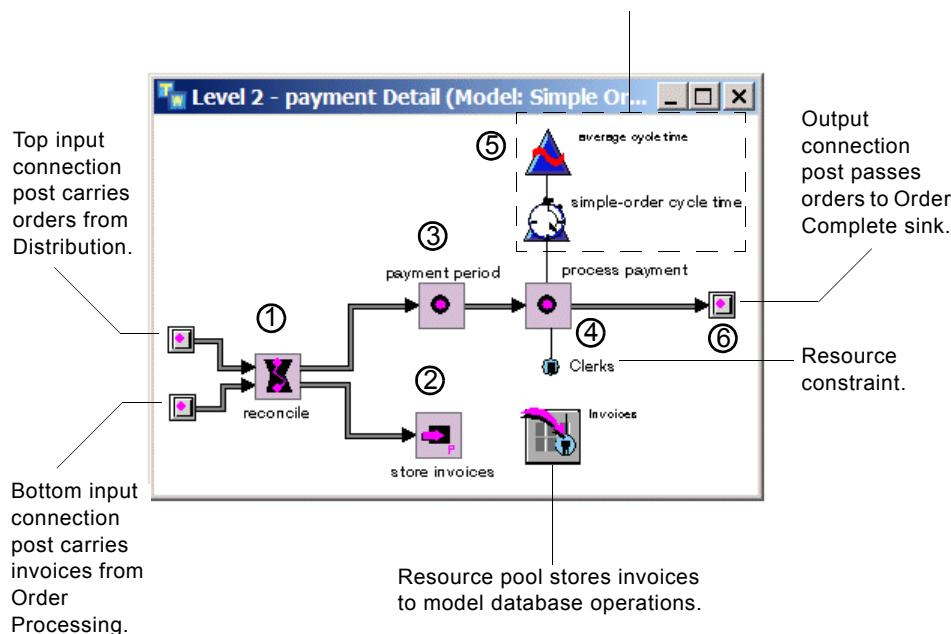
**To view the details of the Payment task:**

→ Show the detail of the Payment task.

ReThink displays this detail:

The Delta Time probe probes the Process Payment task to obtain the cycle time of each order from its creation time to the end of the payment task.

The Moving Average probe probes the Delta Time probe to compute a moving average of the cycle time.



The payment detail performs these tasks in this order:

- 1 The Reconcile block receives orders from the Distribution task and invoices from the Order Processing task.

The invoices must wait on the input path of the Reconcile block until the matching order arrives, at which point they are reconciled.
- 2 The invoices pass to the Store Invoices block, which stores the invoices in a resource pool to model a database operation.
- 3 The orders pass to the Payment Period task, which represents the waiting period for receiving payment.
- 4 Once the payment period has passed, the orders flow into the Process Payment task, which requires a clerk to process.
- 5 The model uses a Delta Time probe and a Moving Average probe to compute the current cycle time and average cycle time for processing each order from receipt to payment.

The Delta Time probe obtains the **creation time** of each order and compares this time to the current time at the end of the payment task. The Moving Average probe computes an average over time of the current cycle time.
- 6 The order passes through the output connection post to the Order Complete block, which deletes the order and signals the end of the process.

## Using Resources to Constrain the Model

The Order Processing detail and the Payment detail require clerk resources to create new customer accounts, generate invoices, and process payments.

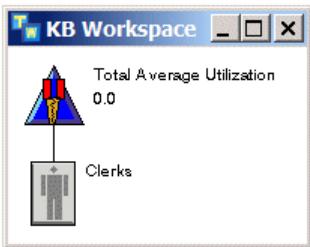
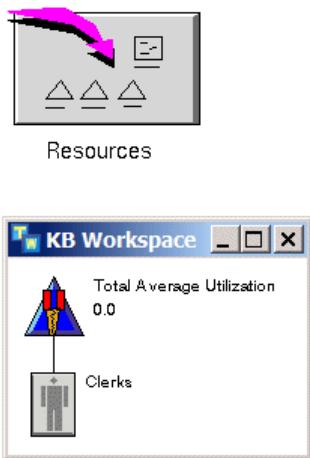
The resources for these tasks are located on the detail of the Resources Organizer tool.

### Viewing the Clerks Resource

**To view the available clerk resources:**

- 1 Display the top-level Simple Order Fulfillment Process Model workspace.
- 2 Show the detail of the Resources organizer.

ReThink displays this workspace:



This workspace contains a clerks resource and a ReThink instrument. The instrument probes the average utilization of all the resources in the pool. The average utilization of a resource measures the total amount of time that a resource is allocated to a task, compared to the total amount of time that the model has been processing. This statistic provides insight into how efficiently the process is using the resources over the duration of the simulation.

For example, if two resources are available and the total average utilization is approximately 1.6, the process is running very efficiently; each resource is used at about 80% capacity. However, if the total average utilization is less than one, this means that each resource is idle at least half the time, which means the model is not using them very efficiently. Similarly, if the total average utilization is close to 2, the process is probably overusing its resources.

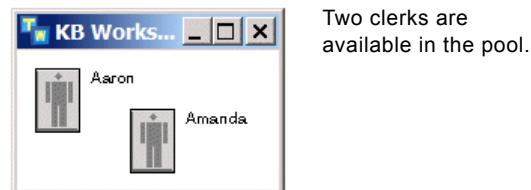
## Viewing the Clerks Pool

You store available resources in a pool, which is a detail subworkspace of the resource object.

### To view the available resources in the pool:

- ➔ Show the detail of the clerks resource.

ReThink displays the pool of available resources:



# Viewing Definitions for Work Objects and Resources

Each ReThink work object is built upon a class definition, which defines its type, its attributes, and its behavior. Work objects are built upon the `bpr-object` class, which is a built-in ReThink class.

The first time you run a model, ReThink automatically creates a class definition for each work object that the model creates, if none exists. You can also pre-define specific class definitions for each work object that a model processes.

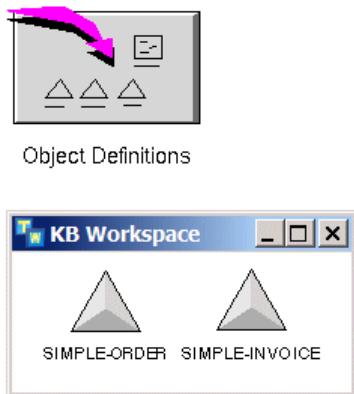
You can use work object class definitions to form the basis of the information systems that will implement your business processes reengineering plan.

In the simple order fulfillment model, class definitions exist for two types of work objects: one for orders and one for invoices. Each of these classes inherits its definition from the built-in ReThink work object class, `bpr-object`. In addition, a class definition exists for a custom resource pool.

**To see the class definitions for the work objects and resources in the model:**

- ➔ Show the detail of the Object Definitions organizer.

ReThink displays this workspace:



For more information on how to create work object class definitions, see [Creating Work Objects](#).

# Observing the Process Under Normal Conditions

To reengineer your business process, you typically start by analyzing the performance of your current process under normal conditions.

To do this, you will run the simulation in jump mode and observe the following charts:

- One chart measures order cycle time and average order cycle time.
- One chart measures average resource utilization over the duration of the simulation.

To simulate normal conditions, the model uses a random triangular distribution to compute the mean time between orders, where the minimum is 1 hour, the mode is 2 hours, and the maximum is 3 hours. You will observe the simulation for approximately ten days of simulation time. You will also observe orders flowing through the process to determine whether work backups exist due to resource constraints.

## Running the Simulation

### To run the simulation under normal conditions:

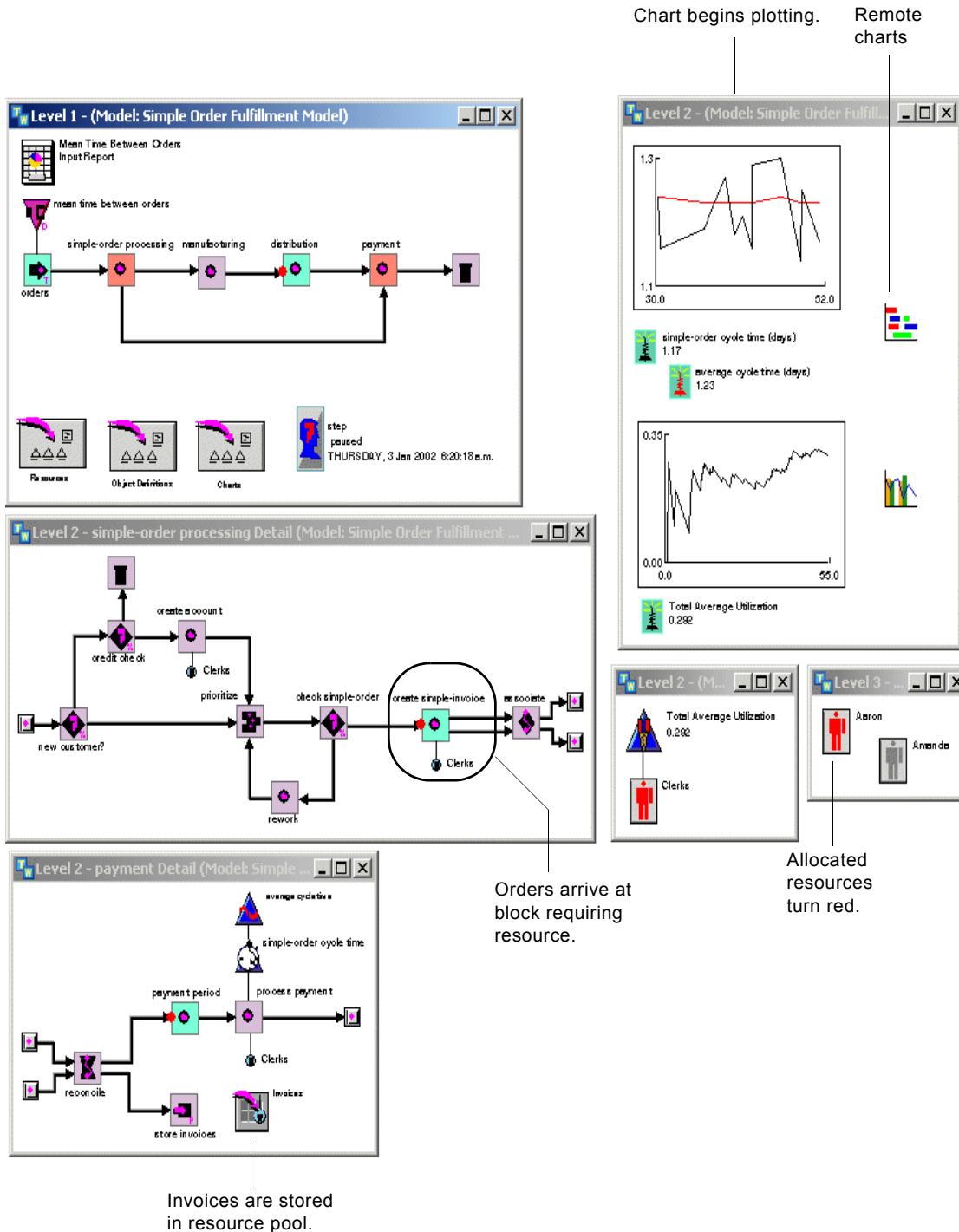
- 1 Display all workspaces in the model, including: the top-level workspace, the two details, the resource organizer detail, the clerks resource detail, and the charts detail.
- 2 Choose Simulation > Start All button or click the equivalent toolbar button.

You will see orders start flowing through the top-level tasks and onto the detail subworkspaces. Invoices are created on the Order Processing detail and flow to the Payment detail. The Reconcile block stores the invoices waiting for associated orders in an internal attribute of the block.

You will also see the resources in the pool turn red, indicating the model is allocating them to a task. The charts that plot the average utilization of the resources begin plotting as soon as the first resource is allocated. The charts that plot cycle time take longer to begin plotting due to the payment period, which is one day. When one day has elapsed, the cycle time also begins plotting.

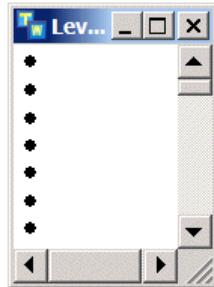
ReThink also provides Windows charts through the Cycle Time and Utilization chart remotes.

Here is the complete model after approximately three days of **elapsed time**, which is the amount of simulation time that has passed since the first work object was created:



**3** Display the detail of the Invoices pool.

ReThink dynamically adds invoices to the pool, which represents a database:



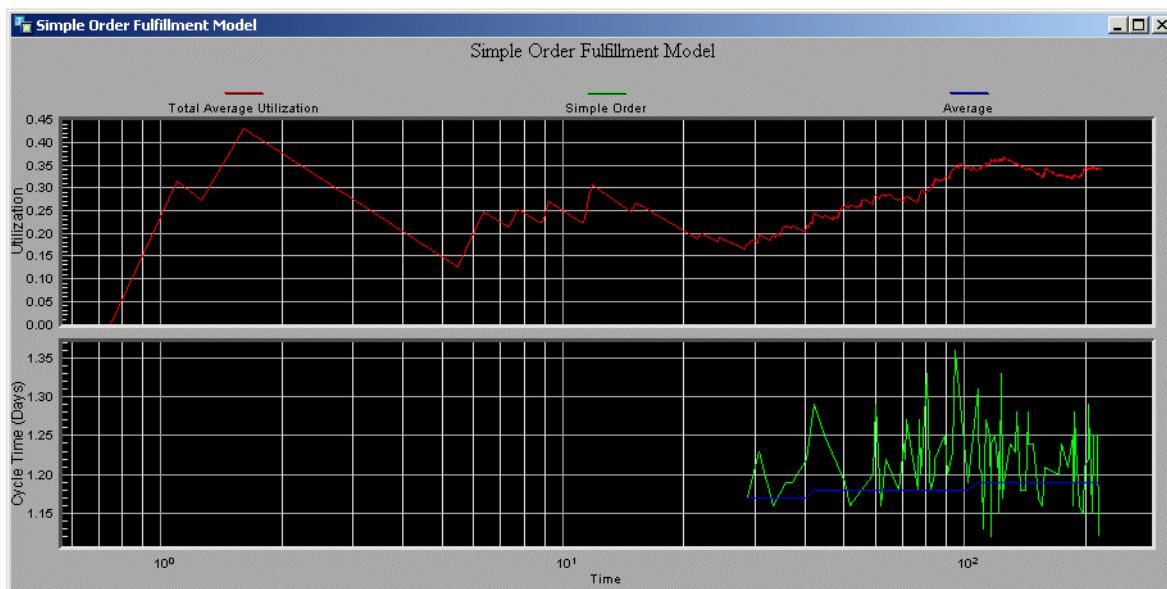
**4** Close the Invoices detail.

**5** Observe the process until the simulation time indicates that approximately ten days have passed.

**6** Pause the simulation by choosing Simulation > Pause or clicking the equivalent toolbar button.

**7** Choose Show Chart on the chart remote:

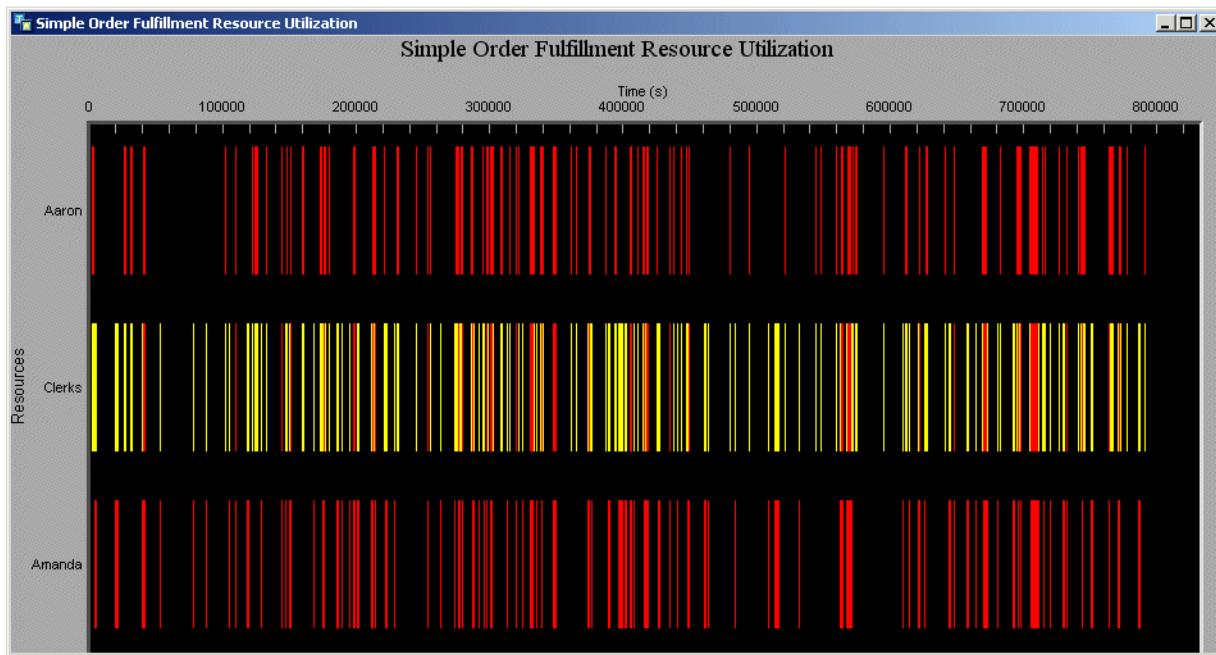
The Utilization and Average Cycle Time charts looks similar to this after 10 days of simulation time, where the resource utilization is approximately 30% and the average cycle time becomes steady over time:



- 8** Choose Show Chart on the resource utilization chart:



Here is the Resource Utilization gantt chart, which shows resource utilization for each resource as well as the overall pool:



## Analyzing the Process

While observing the process, you might ask the following questions, which you can answer by observing the following aspects of the model:

**Q Are there any bottlenecks in the process?**

**A** Work seems to flow through the process fairly smoothly. While observing the simulation, you should not see any green input paths that persist for any significant amount of time. A green input path indicates that work objects are waiting in the path queue for resources to become available, thus creating work backups.

**Q What is the cycle time for the process? Is the cycle time steady?**

**A** The last cycle time that the probe computed on the sample chart is approximately 1.2 days and the average cycle time is approximately the same.

While the cycle time fluctuates quite a bit, the average cycle time remains steady throughout the process, indicating that no significant work backups exist due to resource constraints.

- Q** What is the average utilization of the resources? Are they being used efficiently?
- A** The total average utilization of the resources starts out high. You can see this by observing the resources in the pool, which are mostly red for the first day of the simulation, indicating that they are allocated most of the time.
- Once the process accumulates data points, the total average utilization drops dramatically to a low level and gradually increases to a steady level of approximately 0.3. Again, you can see this by observing the resources in the pool, which are grey a much higher percentage of the time after the first payment period has elapsed.
- A total average utilization of 0.3 indicates that each resource is being used, on average, 15% of the time ( $0.3/2$ ). This percentage is an average over the entire course of the simulation. Thus, the resources are being used very inefficiently.

## Testing the Model Under High Capacity

Now that you have observed the process under normal conditions, the next step is to see how the process behaves operating under high capacity. For example, suppose orders flow into the process once every 15 minutes, on average, instead of once every two hours. Are there any work backups? Are resources used more efficiently?

To model this, you will configure the minimum, mode, and maximum mean time between orders through an input report. You will then observe the simulation until a total of 13 days have elapsed.

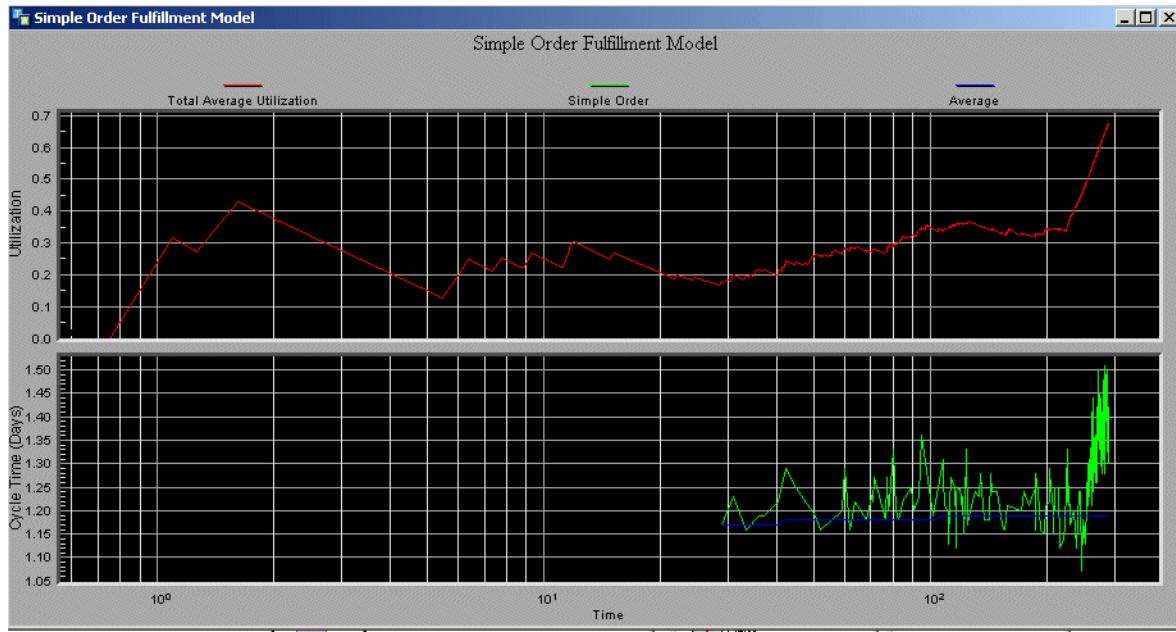
### Running the Simulation

#### To run the simulation under high capacity:

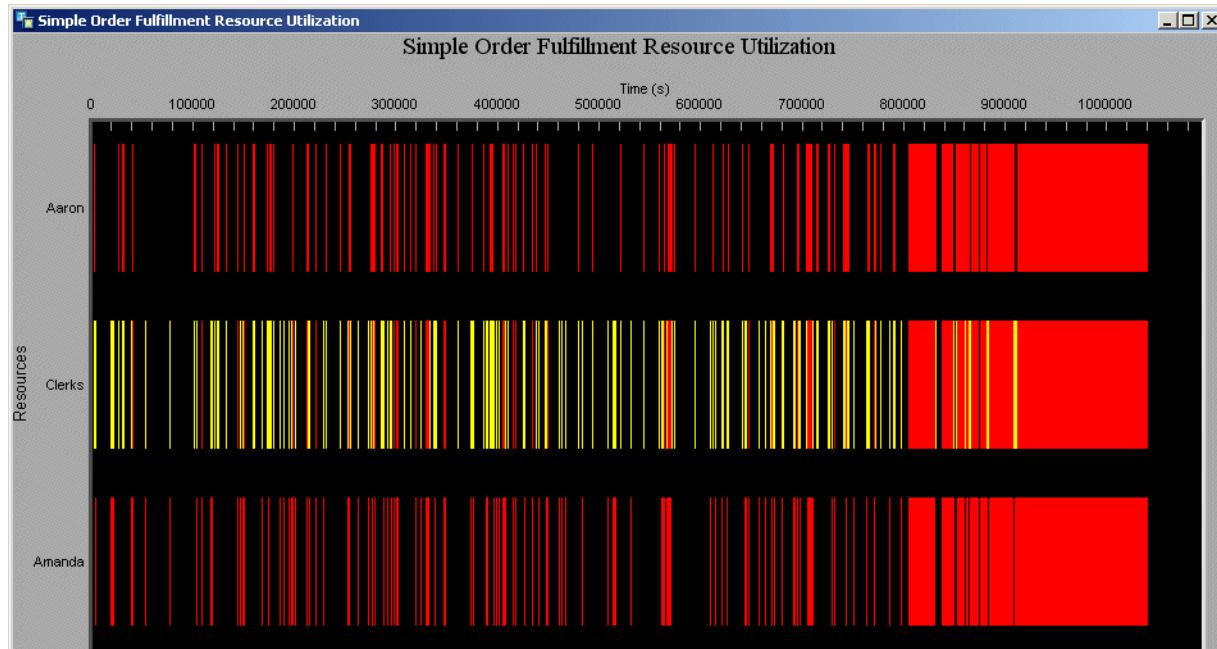
- 1 Select the Mean Time Between Orders Input Report and choose Reports > Show Report.
- ReThink displays an input report for configuring the Minimum, Mode, and Maximum of the Mean Time Between Orders Change feed, which sets the Mean parameter of the Orders source block. By default, the report is configured to accept values in units of one hour.
- 2 Configure the Duration Subtable.min, Duration Subtable.mode, and Duration Subtable.max to be .2, .25, and .3, respectively.
- These values represent units of an hour, the default time unit of the report; therefore, the Duration Min, Mode, and Max are 12, 15, and 18 minutes, respectively.
- 3 Click the Apply button in the report to apply the values to the model and close the report.

- 4** Continue the simulation choosing Simulation > Continue or clicking the equivalent toolbar button

After approximately 13 days, the average cycle time and average utilization both increase:



Here is the Resource Utilization gantt chart:



- 5 When you are finished analyzing the process as described in the following section, reset the simulation.

## Analyzing the Process

While the simulation is running under high capacity, you might ask the following questions, which you can answer by observing the following aspects of the model:

- Q Are there work backups now?**
- A** After about another day and a half of elapsed time, you should see work beginning to backup on the input paths of the Create Invoice block on the Order Processing detail and the Process Payment block on the Payment detail. Because the simulation is processing orders more frequently, many more orders will flow into these blocks, both of which require resources.

ReThink creates an activity object for each work object that flows into a block. An activity represents the amount of work time and cost required to process the inputs to a particular task. Each block keeps track of its current activities in an attribute of the block. If no resource constraints exist, the block can process any number of work objects simultaneously. If resource constraints exist, however, the number of concurrent activities is limited by the number of available resources.

For more information on the activities of blocks, see [Using Blocks](#) in the *ReThink User's Guide*.

**To observe the number of orders currently being processed:**

- ➔ Choose Snapshot Activities on the Payment Period task and on the Process Payment task on the Payment detail.

ReThink displays a temporary workspace that contains the activity objects the block is currently processing. The Payment Period task is currently processing a large number of activities. The Process Payment task, on the other hand, can process a maximum of two activities, due to resource constraints. If the work object on the input path of the block is red, the block will have one activity object, and if the work object is black or the block has no input work object, the block will have no activities.

When a day or so has elapsed, orders start flowing out of the Payment Period block into the Process Payment block, at which point orders start backing up on the input path to the Process Payment block, as indicated by the green input path.

**To see how many orders are waiting for resources at the Process Payment task:**

- ➔ Choose Snapshot Queue on the green input path to the Process Payment task on the Payment detail.

ReThink displays a temporary workspace that contains numerous orders waiting to be processed. The orders are waiting on the input path due to resource constraints.

At the same time, work objects on the input paths to the other tasks that require resources also begin to back up, as indicated by the green paths on the Order Processing detail. You can snapshot the path queue for these paths as well to observe the backups.

Notice that the input path to the Create Invoice task has approximately the same number of orders waiting to be processed as does the Process Payment task; by default, ReThink randomly allocates resources to all the tasks that are waiting for the same resource. However, you can configure how ReThink allocates resources between tasks that require the same resources.

**Q Which path has the longest wait time?**

**A** Choose Indicate Longest Wait Time Path on any workspace in the model.

The path with the longest wait time turns magenta and indicates the wait time. The wait time should be more than one week on the input path to the Create Invoice task. You can also visualize the most used and busiest paths in the model.

**Q What is the effect on cycle time?**

**A** After the model begins to process the initial backups, the order cycle time starts to go up dramatically to approximately 1.5 days.

**Q What is the effect on resource utilization?**

**A** Resource utilization also goes up to a total of approximately 0.6 after 13 days. This means that each resource is now being used almost 30% of the time.

Note that although the resources appear to be allocated almost all the time, as indicated by their color, the average utilization on the chart increases only gradually. This is because ReThink computes average utilization over the entire life of the simulation.

**Q Is the process more or less efficient?**

**A** While the model uses resources more efficiently, the process as a whole is less efficient because cycle time has gone up significantly.

# Automating the Payment Process

Now that you have observed the model under adverse conditions, you might want to consider alternative modeling scenarios and compare the performance results over time. This is one technique that ReThink provides for performing **work flow analysis** on a model to reengineer a process and make it more efficient.

For example, you might ask the question:

- Q** **Can I automate any task to reduce bottlenecks in the process when operating at maximum capacity?**

Suppose you want to test a slightly different version of the model while running at high capacity. For example, you might consider automating the payment process and reducing the number of available resources from two to one.

To do this, you create a duplicate model by copying the existing Model tool, you specify different resource constraints, and you run this simulation next to the simulation of the current process. In this way, you can do a side-by-side comparison of the performance results of the current process model and the automated process model.

---

**Caution** You can only copy a model when the simulation is reset. If you copy a model while the simulation is running or paused, the work objects that the original simulation is processing become permanent objects of the copied model, which you must delete manually.

---

## Running the Alternative Simulation

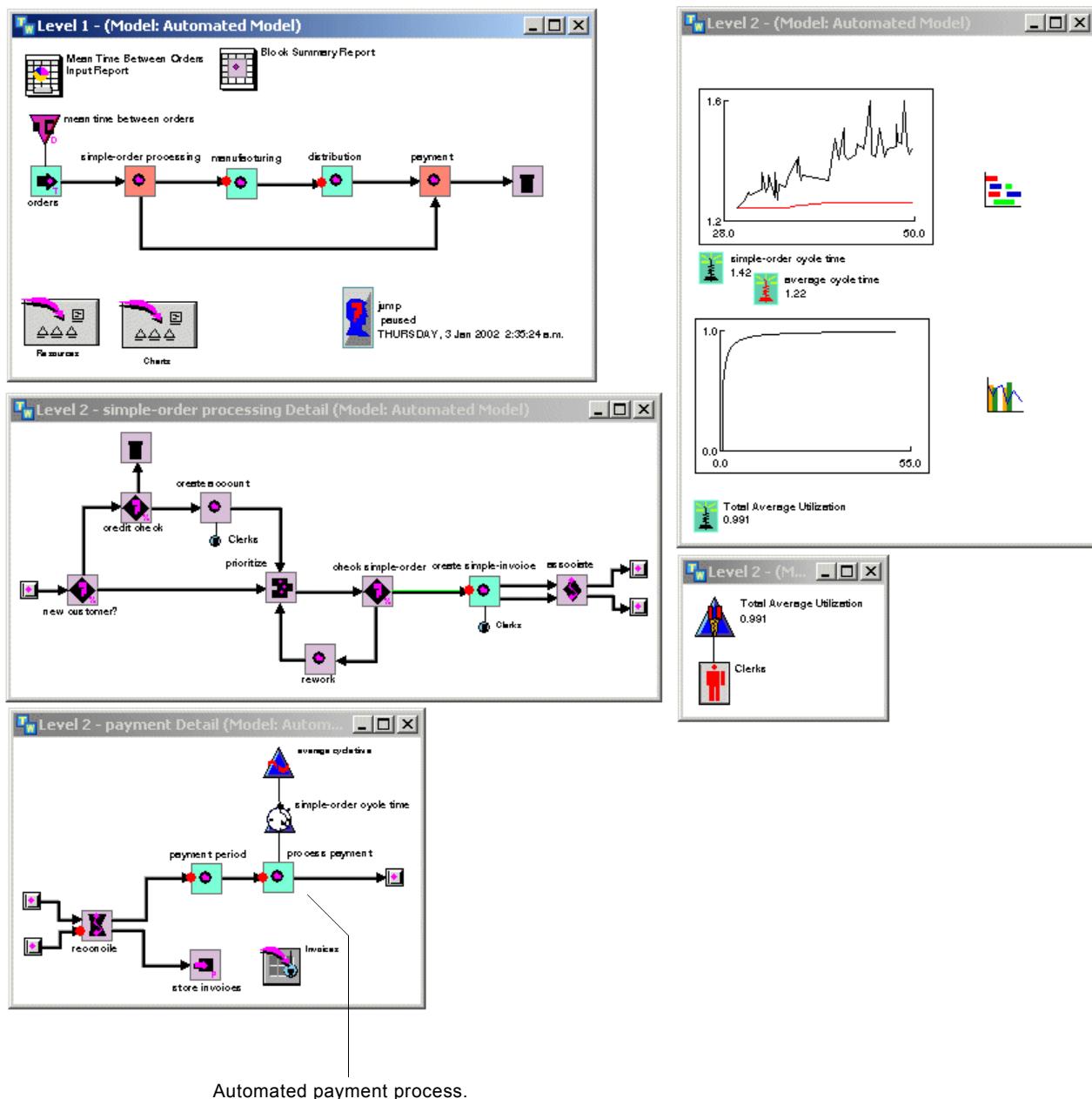
The Automated Model on the Complete ReThink Model workspace contains the alternative model.

**To run the alternative simulation:**

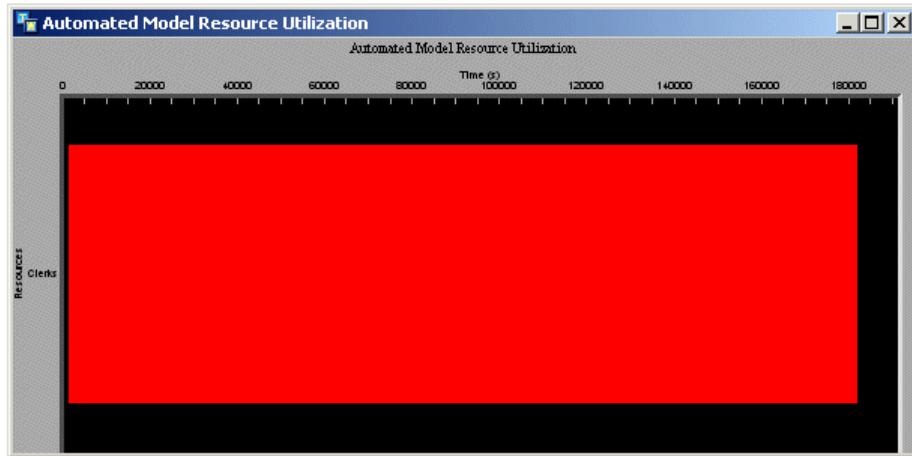
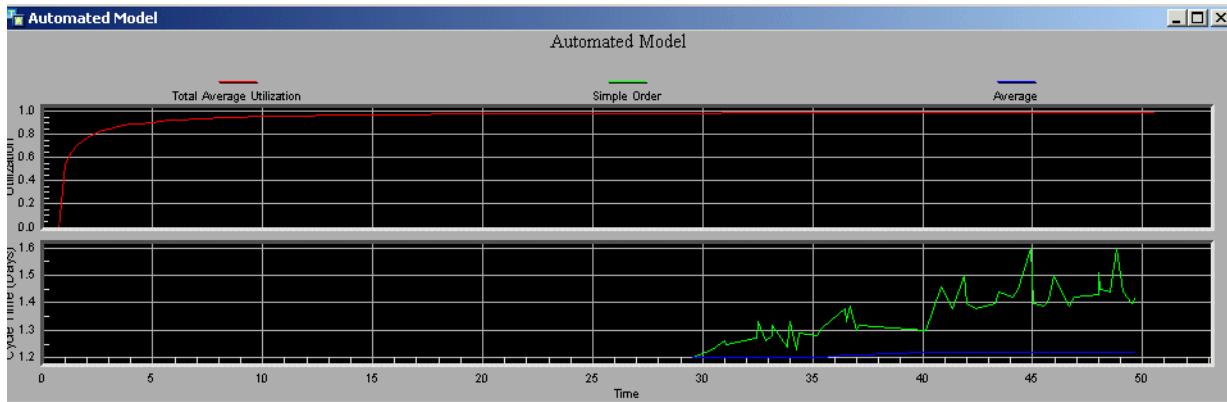
- 1** Display the Automated Model detail and display all the details.
- 2** Choose Simulation > Start All to start the simulation.

The simulation tests the model under maximum capacity, which uses 15 minutes as the mean time between orders.

Here is the complete model after approximately three days of elapsed time:



After approximately six days of elapsed time, the charts look something like this, where the cycle time for orders and the average utilization of resources are both excessively high:



## Analyzing the Process

To test the automated payment process model under maximum capacity, you might ask the following questions, which you answer by observing the following aspects of the model:

- Q** What is the cycle time of the automated model compared with the actual process model?
- A** To see this, display the charts of the original model, which is paused, and compare it with the alternative model.

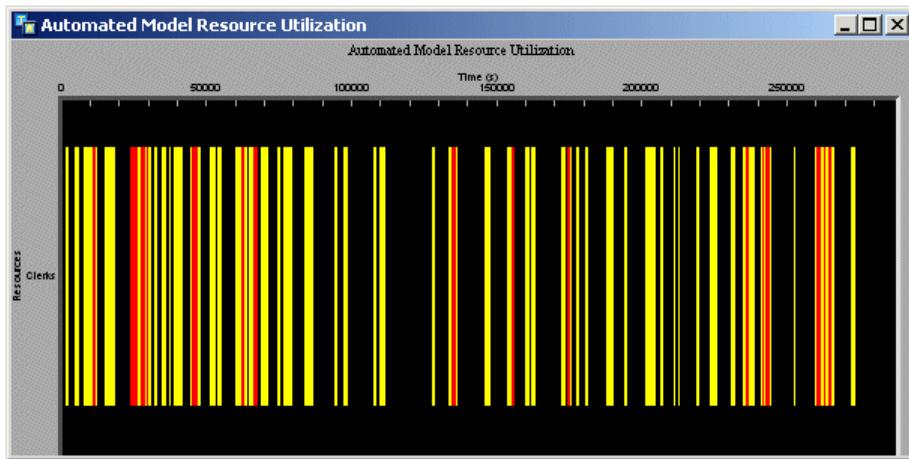
When the original model ran under maximum capacity, the cycle time was approximately 1.5 days and rising. In the alternative model, the cycle time is

approximately 2.5 days. Thus, the cycle time is still excessively high and the average cycle time is rising.

- Q What is the average resource utilization compared with the actual process model?**
- A** The average utilization of the single resource climbs to approach 100% utilization. While the alternative model uses resource much more efficiently than the actual model, the model now somewhat overuses the resources.
- Q Are there any work backups in the process?**
- A** After about a day of elapsed time, you will notice the input paths to the Create Account task and the Create Invoice task on the Order Processing detail are green more of the time.
- To determine how many orders are waiting to be processed, snapshot the queue on each input path. If you observe the process over time, you will continue to see work backups.
- Q What happens if you set the mean time between orders to a number that is half-way between maximum capacity and minimum capacity?**
- A** If you set the mean time between orders to a higher number and observe the process for another couple of days of elapsed time, the average resource utilization remains excessively high, and work continues to back up on the input paths of the blocks that require resources.
- Q Suppose you wanted to experiment with adding a resource to the model?**
- A** ReThink facilitates “what-if” analysis by allowing you to specify different resource constraints, model configurations, and input parameters.
- Alternatively, you can copy the current model, experiment with a different scenario in the copied model, and compare the alternative scenario with the current scenario.
- You can also simply reset the current model and observe the performance under different conditions.
- To add a resource to the model:**
- 1** Reset the simulation.
  - 2** Choose Create Detail on the resource to create a resource pool for the existing resource.
  - 3** Display the Resources palette of the ReThink toolbox.
  - 4** Create two people resources and place them on the detail.
  - 5** Run the simulation again under a capacity that is half-way between high and low capacity (0.5, 1, 1.5).

If you observe the process over time, you will see that the simulation now uses resources much more efficiently, at approximately 70% total. Also, the average cycle time levels out at approximately 1 day.

These charts show the simulation after approximately four days of elapsed time. Notice that under this scenario, the cycle time of orders and the average utilization of resources both level out at acceptable values.



By automating the payment process and fine-tuning the number of resources, you have succeeded in reengineering your business process to make it more efficient.

Congratulations!



# Building Process Models

---

*Describes the high-level tasks involved in organizing and building a process model, using ReThink.*

Introduction **61**

Organizing Models **62**

Determining the Process and Building the Model **65**

Computing Metrics and Obtaining Outputs **73**



## Introduction

When you build a process model, using ReThink, it is important that you use standard modeling techniques. Standards are critical for providing consistency when more than one individual is building a model. However, standards are also important for allowing someone other than the model builder to describe the purpose of the model and explain how it works.

This chapter uses the model presented in [Turbine Blade Model](#) to describe general guidelines for:

- Organizing a model.
- Determining the process and building the model.
- Computing statistics and obtaining outputs.

# Organizing Models

A ReThink model can become quite complex rather quickly; thus, it is a good idea to structure your model in a clear, concise, and consistent manner, which eliminates clutter and makes it easier to understand.

The following sections show the steps you should follow to:

- Organize your model when you begin building it.
- View the Navigator to show all objects in the model.
- Save your model.

## Building Your Model

This section describes the steps you should follow to organize your model when you first begin building it.

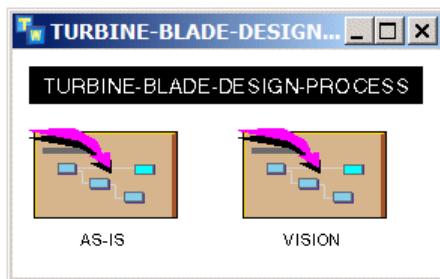
### Project

The first step in creating a model is to create a new project. ReThink creates and loads a new module with the project name you create.

### Models

The next step in organizing your model is to place one or more Model tools on the top-level workspace, then name the models. Model tools contain the individual models that you build. We recommend that you place all Model tools pertaining to a given process on one top-level workspace.

The following figure shows the Aero Design and Manufacturing Process model, which defines two Model tools, As-Is and Vision:

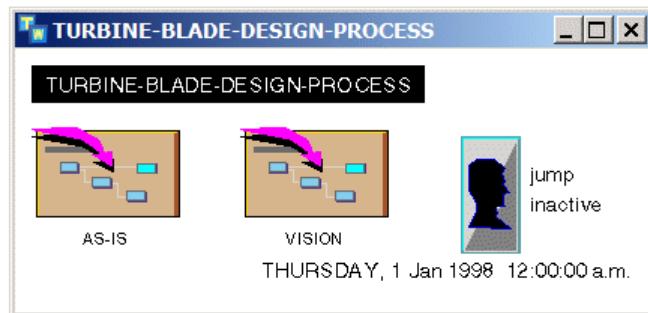


You build the actual models on the detail of the Model tools.

## Scenario Tool

In general, you place a Scenario tool on each model detail to ensure that the scenario runs only one model. Alternatively, you could place the scenario on the top-level workspace to run both simulations at the same time.

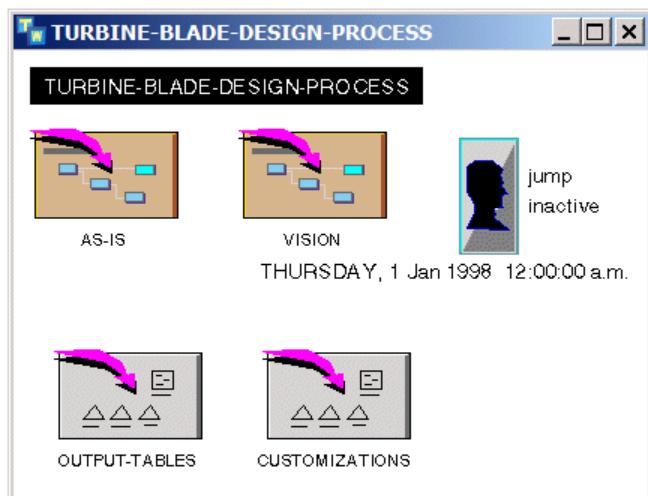
Here is the top-level workspace with a Scenario tool that controls both models:



## Organizer Tools

The last step in organizing your model is to place one or more Organizer tools on the top-level workspace and/or on the model details, as needed. **Organizers** contain work object class definitions, resources, and other information that the model requires but that is not directly connected to the model. You typically place on the top-level workspace organizers that contain objects that are common to all models, such as work object class definitions. You place on model details organizers that contain model-specific objects, such as resources and charts.

This figure shows the top-level workspace with the Model tools and Organizer tools:



## **Model Detail**

A model detail typically consists of:

- Connected blocks that describe the process.
- Unique class definitions that you place on the detail of an organizer.
- Unique resources that you place on the details of other organizers.

## **Model Definitions Organizer**

The Model Definitions organizer contains class definitions that are specific to the As-Is model. To organize these definitions, create an Organizer tool for each category of definition, for example:

- Work object class definitions.
- Block definitions that customize the block's icon.
- Custom block definitions that customize the block's behavior.

## **Resources Organizer**

The Resources organizer contains all of the resources, such as human resources, computers, and equipment that the model requires, as well as the associated **resource pools**, which define the organizational structure of those resources. For instance, a pool of engineering analysts might contain several human resources, each with an associated cost and/or priority.

Typically, you create resources as part of modeling the details of a process, as described in [Modeling Resources](#).

## **Storage Pools Organizer**

The Storage Pools organizer contains resource pools that represent some kind of storage of work objects in the model. For instance, the model might represent engineering files as work objects, which flow from task to task. At some point in the process, the model might store those files to a pool. At some later point in the process, the model might pull those files from the pool for further processing. Other examples of storage pools are database storage or hardcopy filing systems.

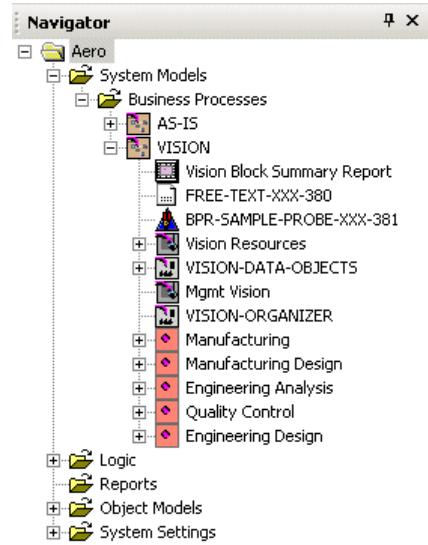
---

**Note** Although you use resource pools to represent both the hierarchical organization of human and other resources, and the storage of various type of work that the model processes, do not confuse the two types of pools. In particular, we recommend that you always keep the two types of pools in separate organizers.

---

## Viewing the Navigator

You can show the Navigator, which includes all objects in the project and in the model. Here is the navigator for the *aero.kb* model:



## Saving the Model

When you save your model, specify the:

- Module to save, which is the project name, for example, *aero*.
- Filename, which should match the name of the top-level module, for example, *aero.kb*.

## Determining the Process and Building the Model

Once you have created an organized structure for your model, you can begin to determine the process and build the model. You should begin modeling by mapping the process at a very high level.

The Aero Design and Manufacturing Process consists of five main steps:

- Engineering Design
- Engineering Analysis
- Manufacturing Design
- Manufacturing

- Quality Control

Obviously, a great deal of detail is involved in completing each high-level step. In fact, you can think of each step as its own process, which feeds each subsequent process in the model.

As you map out these initial steps, do not be concerned with modeling the resources required for each task or the times needed to complete each task. You will complete these modeling details as you flesh out each the details of each step.

## Identifying Metrics

Before you begin building your model, you should always determine what metrics the model should measure. You can capture metrics such as:

- Process **cycle time**, which represents the difference in simulation time between two events in the model.
- Process **cost**, which represents the sum of all fixed and variable costs applied to a work object.
- Individual task **cost**, which represents the sum of all fixed and variable costs associated with the individual tasks and its associated resources.
- Resource **cost**, which represents the sum of all fixed and variable costs associated with a resource multiplied by the amount of time the resource is allocated.
- Resource **utilization**, which represents the percentage of simulation time that a resource is allocated to a task, as opposed to idle.

Knowing the desired metrics at the beginning helps you build more efficient models by structuring them according to the metrics and providing the data required to meet the goal. If you do not know what you are measuring ahead of time, you will often need to redesign parts of your model as you determine the goals.

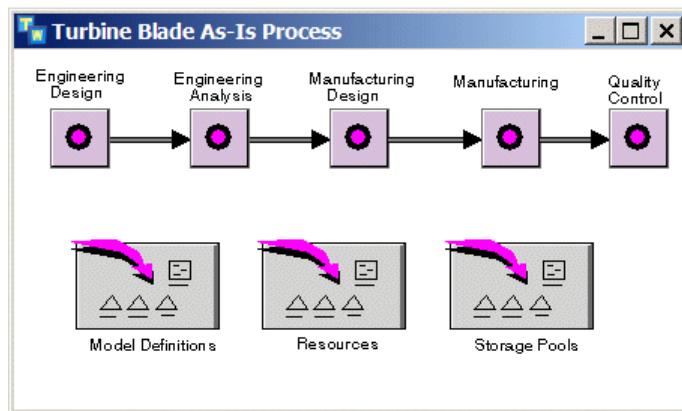
## Building the Top-Level Model

When you build the top-level model, we recommend that you follow these conventions to ensure that anyone viewing the model can easily understand it:

- Always build models so the direction of flow of the work objects is consistent. Traditionally, process models have a left-to-right flow, where the model creates work objects on the left side of the workspace, and either deletes or stores work objects on the right side.
- Always make the top-level model simple and easy to understand; do not clutter it with too many blocks. You can add detail to each block as you build the model.

- Begin the model with a Source block, end the model with a Sink block or a Store block, and use Task blocks to represent the high-level tasks. Alternatively, you can place the Source block and Sink block on the detail of the first and last top-level Task blocks, respectively.
- Label each block. The label should describe the task the block represents. For high-level tasks with detail, the label is typically a noun that identifies the high-level step in the overall process, for example, "Manufacturing."

Here is the top-level model for the As-Is process that represents the high-level process:



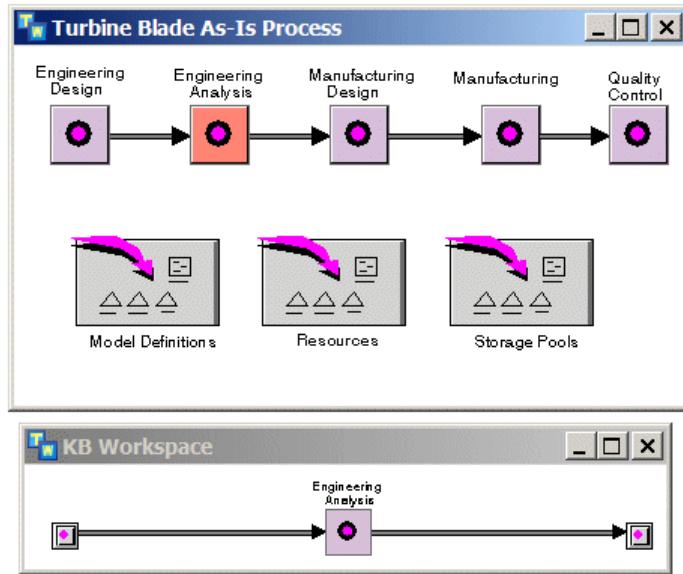
## Modeling Task Details

After you map out the high-level process, you can begin to break down each step into greater levels of detail to build a hierarchical model. To model task details:

- Create detail for the block. Creating detail changes the color of the top-level task to indicate that it has detail.
- Label the input and output connection posts with a description of where the input is coming from and where the output is going so you do not need to switch back and forth between the detail and the high-level model. This practice is also helpful when you print model details.
- Add connection posts to the detail by connecting the top-level task to other blocks.
- Iteratively add and connect blocks to the detail to describe the process, including adding nested levels of detail.
- Label each task that actually performs some action by using a verb and a noun, for example, "Process Order."

## Initial Task Detail

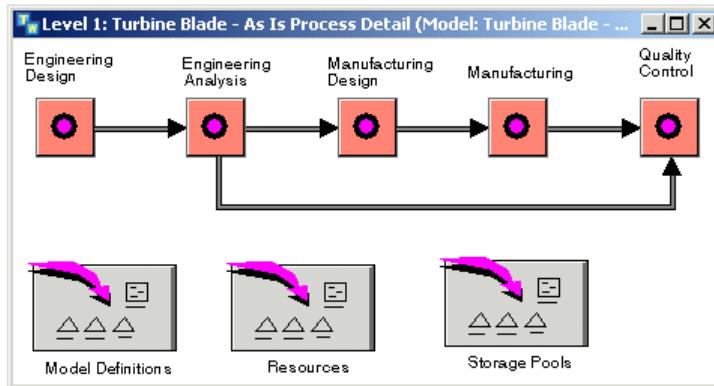
The following figure shows the top-level As-Is Model and the initial Engineering Analysis detail:



## Adding Connection Posts to the Detail

After further investigation, the process reveals that the Manufacturing Design task and the Quality Control task process the engineering design in parallel.

To accomplish this, the Engineering Analysis detail requires two connection posts, one which goes to Manufacturing Design and another which goes to Quality Control. To create the second connection post on the detail, create an output path on the high-level Engineering Analysis task and connect it to the Quality Control task on the high-level model, as follows:

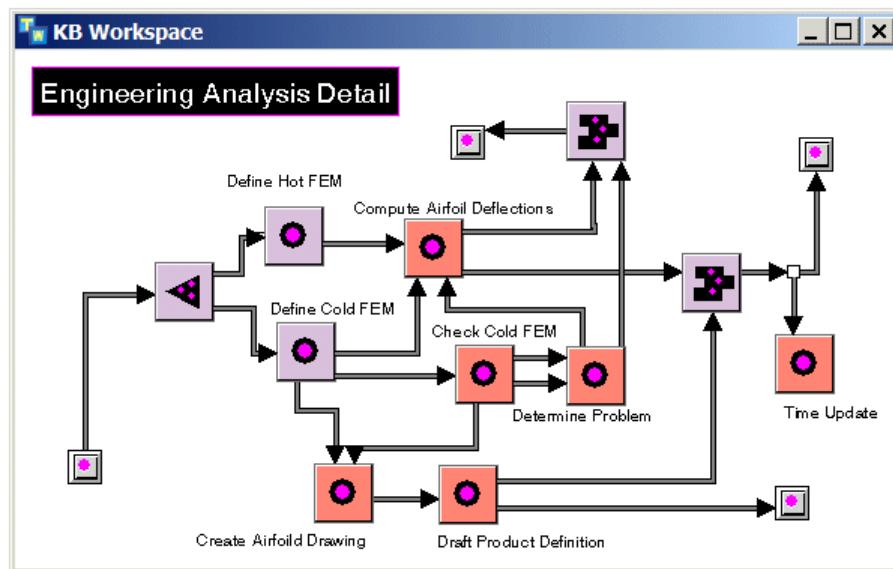


## Configuring the Detail

To add blocks to the detail, disconnect the existing paths, and insert and connect additional blocks into the process, as needed. You can use any type of block that is appropriate to model the process. You can also add nested levels of detail by creating detail for Task blocks on the detail of another task.

For general information about connecting blocks, see [Using Blocks](#) in the *ReThink User's Guide*.

The second iteration of the Engineering Analysis detail uses a variety of blocks with nested levels of detail, as this figure shows:



## Modeling Resources

As you continue modeling the details of the high-level task, you can begin to model the resources that each lower-level task requires. For example, several of the engineering tasks might require analysts and computers.

If these resources are unique to the As-Is model, you create them on the detail of an organizer located on the As-Is model detail. Otherwise, if both the As-Is and Vision model share those resources, you create them on an organizer on the top-level workspace.

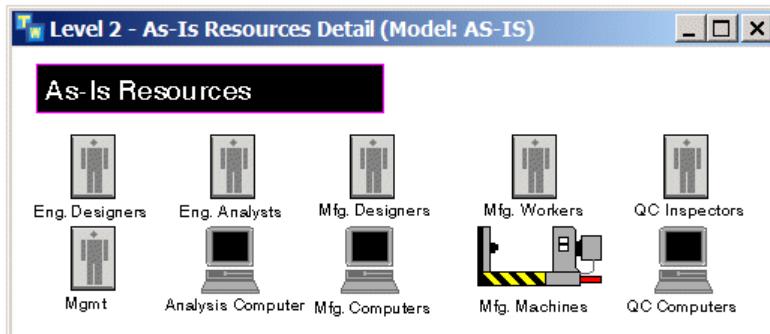
To model resources, you configure:

- The quantity of available resources of each type by creating detail for each type of resource and adding individual resources to the detail.
- The general category of each individual resource in the pool.
- The fixed and variable costs of each individual resource in the pool.

- The priority of each individual resource in the pool, if you are allocating resources based on priority.
- Whether the same resource is shared in multiple pools by using a **surrogate**.

## Creating Resource Pools

For example, here is the As-Is resource pool detail with human, computer, and machine resources:



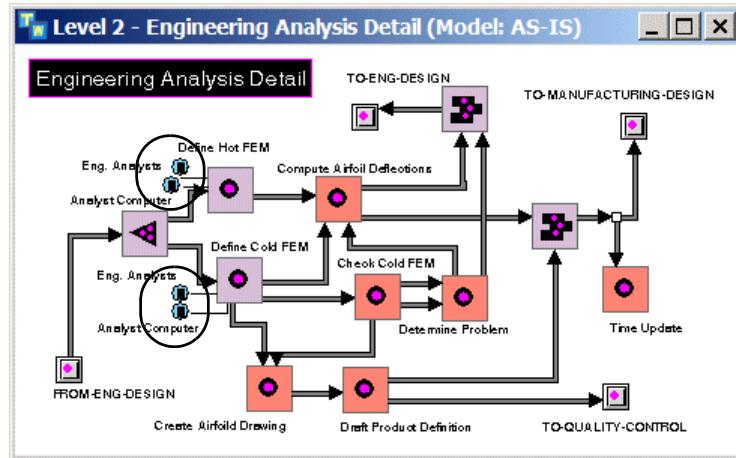
## Creating Resource Managers

To use resources in a model, create a **resource manager** from each resource pool and attach it to each Task block that requires a resource. The resource manager **allocates** the specified number of resources to the task when it activates and **deallocates** them when the task is complete. Resource manager labels match the label of the resource pool from which it allocates resources.

You can configure the following aspects of how the Resource Managers allocate and deallocate resources:

- Whether it allocates resources from a pool based on cost, utilization, priority, or randomly.
- How much of each type of resource the task utilizes, for example, a single resource, multiple resources, or a partial resource.
- Whether the resource manager allocates the same resource for several sequential tasks.

The final iteration of the Engineering Analysis detail might look like this with the resource managers circled:



## Configuring Duration

Once you have described the process by creating, connecting, and configuring blocks, and creating and configuring resources and their associated managers, you can describe the timing of each lower-level task in the process. To do this:

- Determine the mathematical distribution that the block will use to compute its duration, such as, random normal, random exponential, random uniform, or random triangular.
- Configure the parameters that the chosen distribution requires to compute the block's duration.

By default, most blocks use a random normal distribution, which requires a mean and a standard deviation.

## Configuring Costs

Generally, you model process costs by configuring the fixed and variable costs of individual resources allocated to particular tasks. ReThink computes the cost of the activity associated with each task by multiplying the variable resource cost by the duration of the activity and adding the fixed costs.

If a task does not require resources, you can also configure the fixed and variable cost associated with individual blocks. The total cost of an activity is thus the sum of the fixed and variable resource costs, and the fixed and variable block costs.

## Annotating Your Model

As already described, you annotate your model by editing:

- The names of top-level workspaces.
- The labels of Model tools, Organizer tools, blocks, and resources.

In addition, you can use the following **annotations**, available on the Displays palette of the ReThink toolbox, to explain the process further and make it easier for the casual observer to understand:

- The Free Text or Borderless Free Text tool, which you use for small amounts of **free text**, which is text that you can place anywhere on a workspace to annotate the model.
- The Annotation tool, which has detail that you use for larger amounts of text that would clutter the workspace.
- The Readout tool, which allows you to display the current value of any attribute in the model in a **readout table**.

## Creating Work Objects

The basic mechanics of a ReThink model involves creating, processing, and deleting work in a business process. The three categories of objects that you use to represent work in a model are:

- **bpr-object**, which is the default work object type.
- **bpr-container**, which is a subclass of work object called a **container** that defines an attribute into which the Insert block and the Batch block insert work objects and from which the Remove block removes work objects, such as a box or a load.
- User-defined work objects, which are user-defined objects that inherit their definitions from either **bpr-object** or **bpr-container**.

You specify the type of objects a model processes by configuring the path types of the various blocks.

In most cases, the model uses user-defined work objects. To create user-defined work objects, either:

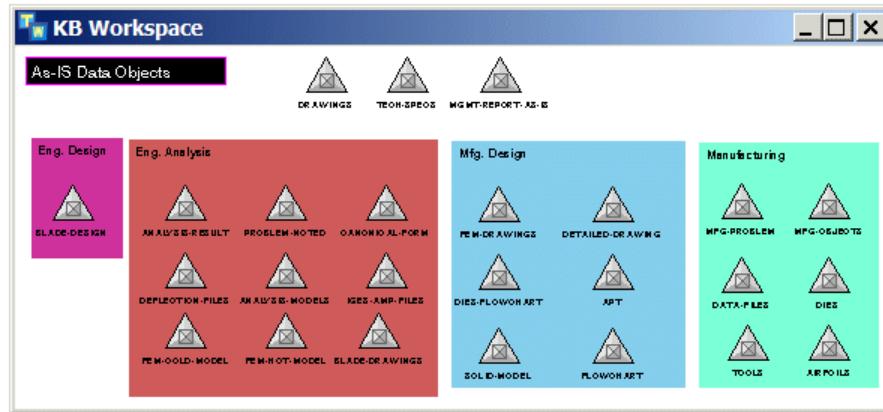
- Configure the path type to be a work object whose definition does not yet exist, then run the model to create the class definition automatically.
- Manually create a class definition from the Tools palette, configure its name, direct superior classes, and class-specific attributes, then configure this class as the output path type of a block to create work objects of this type.

If the work object class definitions are unique to the As-Is model, you create them on the detail of the organizer located on the As-Is model detail. Otherwise, if both

the As-Is and Vision model share the definitions, you create them on an organizer on the Model Definitions detail on the top-level workspace.

You create a class hierarchy of work objects so that you can configure output paths to carry different categories of work objects. For information on configuring path types, see [Using Blocks](#) in the *ReThink User's Guide*.

This figure shows the class hierarchy for work objects that the model processes:



## Computing Metrics and Obtaining Outputs

One of the goals of building a process model is to obtain statistics about how the model performs under various scenarios. For example, if you are building an As Is and a Vision model for comparison, the goal is to measure the same metric in both models and provide a way of comparing the two measurements.

ReThink provides numerous tools for computing metrics and obtaining outputs, including:

- **Instruments** that allow you to supply input parameters to the model and obtain metrics from the model.
- **Sliders** and **type-in boxes** for providing parameters to the model.
- **Charts** that plot metrics over time.
- **Dialogs** that show summary metrics for individual blocks, resources, instruments, and work objects.
- **Reports** that provide summary metrics on all types of objects in the model, from which you can also create charts.

You create sliders and type-in boxes from feeds, and you create charts from probes. You can create Windows charts, using a **remote chart**.

You create output reports for blocks, paths, probes, resources, or work objects, which include summary metrics for all system-defined attributes. You can write the summary report data to a CSV file, Microsoft Excel spreadsheet, or database.

# Turbine Blade Model

---

*Provides a step-by-step tutorial of a full-scale simulation, which compares a traditional engineering design process with a reengineered process.*

- Introduction **76**
- Opening the Model **76**
- Running the Simulation **78**
- Comparing Processes **78**
- Exploring the As Is Model **79**
- Exploring the Vision Model **87**
- Viewing the Results **92**
- Organizing Versions **93**
- Using Customizations **95**
- Viewing the Object Hierarchy **96**
- Typical Questions and Answers **96**



# Introduction

This model simulates the current and future processes of the design and initial manufacturing of aircraft turbine blades. The models in this application illustrate the difference in time and cost between the classic “throw the design over the wall” process and the more effective “design for manufacturability” process.

This tutorial assumes knowledge of basic ReThink behavior and techniques.

For information on how to perform the operations described in this tutorial, see the *ReThink User's Guide*.

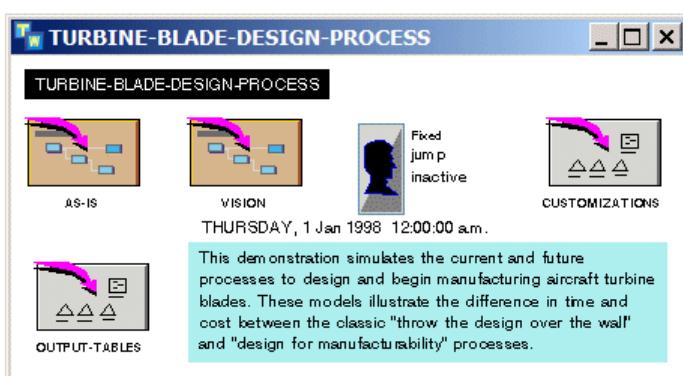
## Opening the Model

The Turbine Blade model is available as model that you can open.

### To open the turbine blade model:

- ➔ Open the file `aero.kb` from the `rethink\examples` directory of your ReThink installation directory.  
**or**
- ➔ Choose Start > Programs > Gensym G2 2011 > Examples > G2 ReThink > Aero Model.

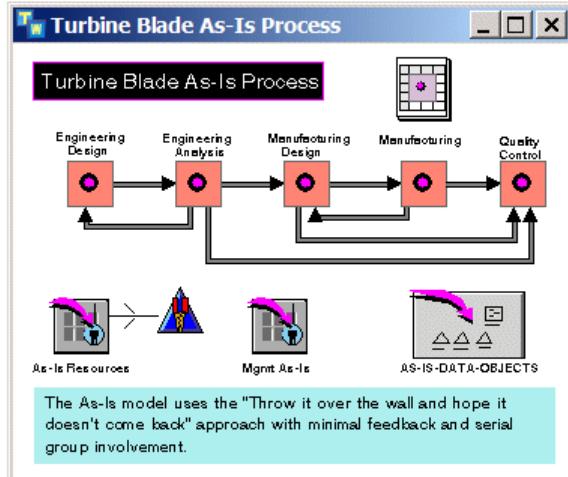
You will see the top-level workspace, Turbine Blade Design Process:



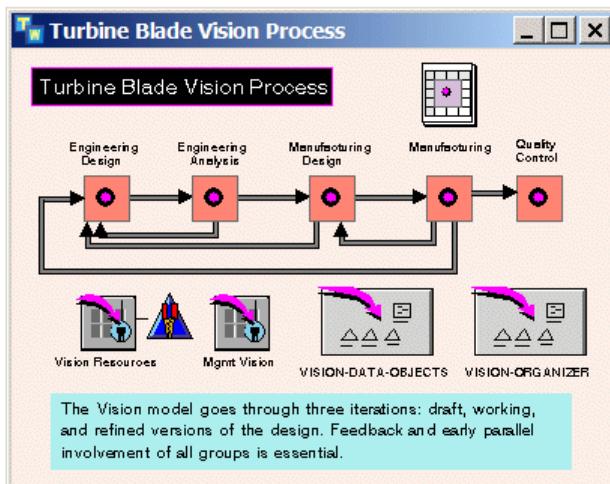
The workspace has two models, one named As Is and one named Vision, which you control with one scenario.

**To view the models:**

- 1 Show the detail of the As Is model:



- 2 Show the detail of the Vision model:



From this high-level view, both models appear similar. As the Task blocks illustrate, the production of turbine blades consists of five fundamental steps:

- Engineering Design
- Engineering Analysis
- Manufacturing Design
- Manufacturing
- Quality Control

Each of these blocks contains several layers of detail, as you will see later.

The primary difference between the As Is and Vision models involves feedback loops, as shown by the red-colored paths. In the As Is model, there are feedback loops between Engineering Design and Engineering Analysis, and between Manufacturing Design and Manufacturing. However, in general, the As Is model takes a serial approach; once objects pass through the Engineering stages into Manufacturing Design, they do not return. In contrast, the Vision model makes use of concurrent engineering concepts. With the exception of Quality Control, feedback paths exist between each stage in the Vision model.

The Vision model is an example of the counter-intuitive notion that doing something several times can actually be faster than doing it once. The problem with trying to finish something satisfactorily the first time is that it rarely happens in practice. Therefore, preparing for several development cycles often saves time and reduces costs.

## Running the Simulation

The Scenario tool controls both simulations.

### To run the simulation:

- 1 Close the details of the two models.
- 2 Start the simulation running.

Starting the simulation starts both the As Is and Vision model simulations. Due to the scale and complexity of the model, you will notice a delay whenever you reset and start the model.

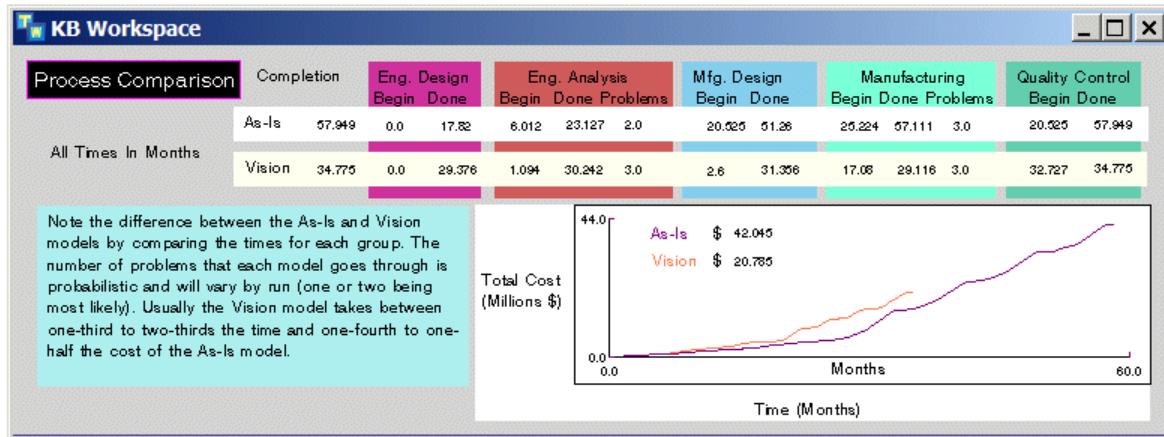
The simulation runs for a period of time, then it stops.

## Comparing Processes

The Output Tables organizer contains a display of several key simulation values important for evaluating and comparing the performance of both the As Is and Vision approaches. In each model's workspace hierarchy lies an assortment of instruments such as Timestamp feeds and Sample probes. These instruments tie data from the model to this display.

### To display the output of the process:

- 1 Show the detail of the Output Tables organizer, which is located on the Turbine Blade Design Process workspace:



In the simulation shown in the output table above, the As Is model took approximately 60 months to complete and cost approximately \$40 million dollars.

- 2 Close the Process Comparison workspace.

## Exploring the As Is Model

Now you will look at the components of the As Is model.

### Exploring the High-Level View

First, you will observe the high-level view of the Turbine Blade As Is model.

#### To explore the high-level view of the As Is model:

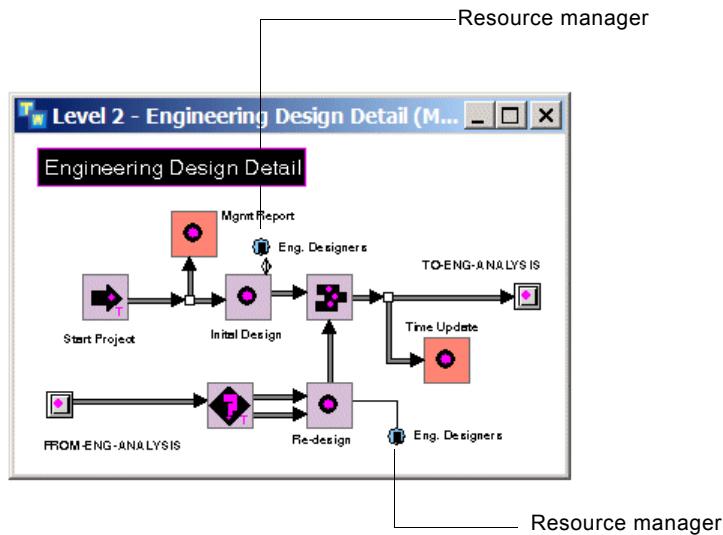
- 1 Show the detail of the As Is model on the Turbine Blade Design Process workspace
- 2 Show the properties dialog for the path between the Engineering Design and Engineering Analysis tasks on the As Is model detail.  
Engineering Design produces **blade-design** objects and passes them to Engineering Analysis.
- 3 Show the properties dialog for the red feedback path between these Engineering stages.  
Engineering Analysis produces **analysis-results** objects and passes them back to Engineering Design.

# Exploring the Engineering Design Detail

Now you will observe the detail of the Engineering Design task.

## To explore the engineering design task:

- 1 Show the detail of the Engineering Design task:



Note the use of the resource managers labeled Eng. Designers, which incorporate resources into the design process.

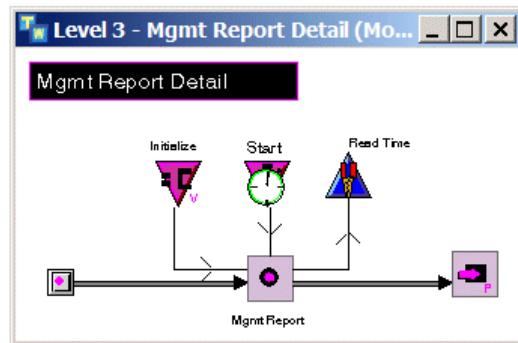
- 2 Show the properties dialog for the Start Project and Initial Design blocks.  
The Start Project block is a Source block that initiates the process by creating a **mgmt-report-as-is** object. The Initial Design task outputs a **blade-design**.  
Initial designs go to Engineering Analysis via the connection post on the right side of the workspace.
- 3 Show the properties dialog for the Initial Design task and click the Duration tab to note the time it takes to complete an initial design.  
The connection post at the left named From Eng. Analysis corresponds to the red feedback path we saw earlier. When problems occur in analysis, redesign work is necessary, as indicated by the Task block labeled Redesign. Designs that were redone are sent along the same path as initial designs with the Merge block.

## Creating Management Reports

For bookkeeping, it is important to record how much time each engineering stage takes. One way to accomplish this goal is by creating management reports.

**To observe the management reports:**

- 1 Display the Mgmt Report detail:



Here, reports are created, stamped with time information by the feeds, and stored in a resource pool by the Store block at the right.

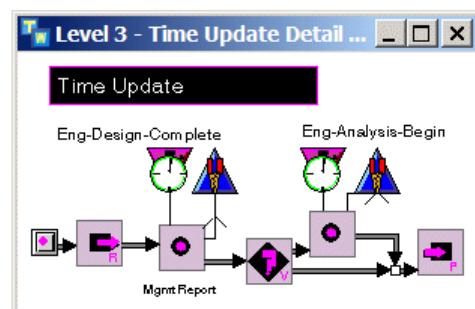
- 2 Close the Mgmt Report detail.

## Recording When Designs are Complete

The Time Update block on the Engineering Design detail accounts for the two types of blade designs, initial designs and redesigns.

**To inspect the Time Update block:**

- 1 Display the Time Update detail on the Engineering Design detail:



For both initial and redesigns, the model must note when Engineering Design is complete. After the Retrieve block on the left retrieves a management report from the resource pool, the Eng-Design-Complete timestamp takes care of this. The Branch block ensures that only initial design reports are updated with an Eng-Analysis-Begin value, since redesigns have already been through Engineering Analysis.

**2** Show the properties of the Branch block and click the Block tab to observe the Branch Mode, Branch Attribute, and Operation.

**3** Display the properties dialog for the two output paths of the Branch block and click the Branch tab to view its Branch Upper and Branch Lower attributes.

Reports that have already been through, that is, reports with eng-analysis-begin attributes of time greater than 1, are put back into the resource pool. Reports with eng-analysis-begin attributes of time 0 are timestamped appropriately before they are stored.

**4** Display the properties dialog for the two Sample probes.

These probes record begin and completion times; the values are linked to remotes, which update a display.

**5** Click either probe and choose Show Remotes to revisit the output tables.

An **indicator** arrow appears next to a remote, which is hidden under the free text. The values of the hidden remotes appear in the table above the indicator. These numbers show begin and completion times that correspond to the timestamps from the Time Update detail.

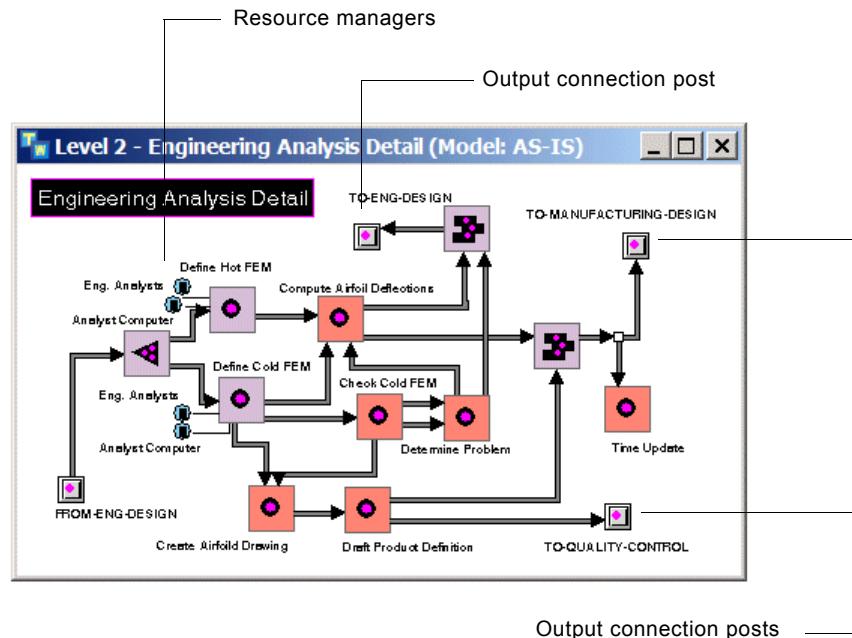
**6** Close the Time Update, Engineering Design detail, and Process Comparison details.

## Exploring the Engineering Analysis Detail

Now, you will explore the details of the Engineering Analysis task.

### To examine the Engineering Analysis task:

- On the Turbine Blade As Is Process workspace, show the detail of the Engineering Analysis task:



Blade designs enter from the connection post labeled From Eng Design. The As Is process takes about 26 weeks to finish these designs. Subsequently, Hot and Cold FEM's (Finite Element Models, which are wireframe representations of the blades) are defined by Eng. Analysts with Analyst Computer resources.

Three output nodes lead to the connection posts named To Manufacturing Design, To Quality Control, and To Eng Design. Remember, there is no feedback path from either Manufacturing Design or Quality Control; once objects leave through one of those connection posts, they are there for the duration of the simulation.

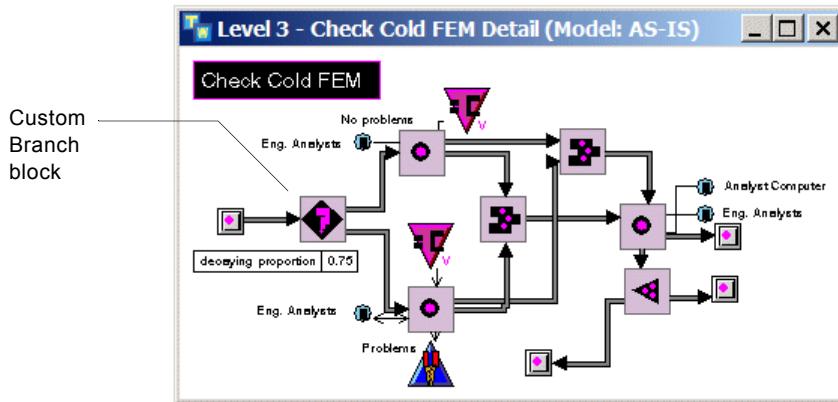
Manufacturing Design typically does not receive completed designs until 15 months into the project. Likewise, they have had no opportunity to make comments. Copies of drawings are also sent to Quality Control, but they are not used for about nine months. The odds that changes will occur before then are high.

## Determining Whether Problems Exist

A feedback path exists back to Engineering Design. This is required when FEM analysis reveals problems. Situations can occur with an FEM, which warrant requesting a new blade design, in other words, a redesign. These problems surface in Check Cold FEM.

### To examine the Check Cold FEM task:

- 1 On the Engineering Analysis Detail, show the Check Cold FEM task's detail:



The heart of this detail lies in the customized Branch block on the left. Its job is to examine cold FEMs and filter out those that are problematic.

In fixed mode, ReThink presets the number of problems to the average case of two, which is determined experimentally from the real-life process. Therefore, the first two **fem-cold-model** objects follow the lower path marked **Problems**. The third will follow the upper path labeled **No Problems**.

In probabilistic mode, the Branch block filters FEMs randomly according to a proportion.

- 2 Show the properties dialog on the Branch block's lower path and observe its Branch Proportion attribute.
- 3 Verify that the value agrees with the readout table marked Decaying Proportion.

The decaying proportion is the proportion of **cold-fem-model** objects that will follow the lower path. Custom procedures divide the proportion in half after each FEM passes through. For example, when the model starts, the percent chance of an FEM having a problem may be 75%. If chance permits and a problem occurs, the next FEM that results from a redesign will have only a 37.5% chance of having a problem. Eventually, a **fem-cold-model** object will be produced without problems, which will allow Manufacturing Design to begin.

**4 Close Check Cold FEM.**

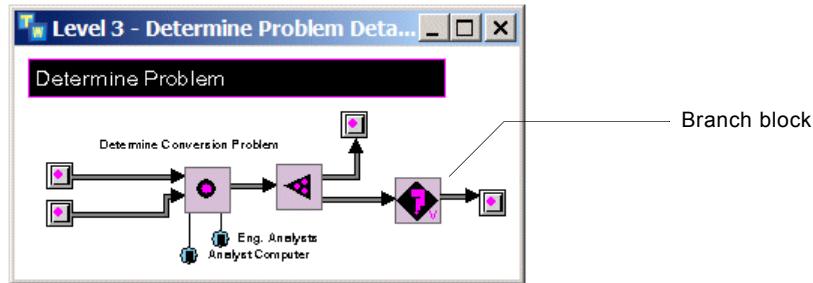
Later, you will view the Customizations organizer to see how the custom Branch block is implemented.

## Recording the Problem

Next, you will look at how the model records the problem in one more part of Engineering Analysis.

**To examine the Determine Problem task:**

- 1 On the Engineering Analysis detail, display the detail for the Determine Problem task:



Notice that the Branch block has only one output path. Now you will see how this Branch block works.

- 2 Examine the Type attribute of the input path of the Branch block.
- 3 Examine the Branch Attribute of the Branch block.
- 4 Examine the Branch Value attribute of the output path of the Branch block.

The effect of the single output path is as follows: problem-noted objects flow in; if their problem attributes are of type not-ok, the objects pass through, enabling the redesign process in Engineering Design; otherwise, nothing happens.

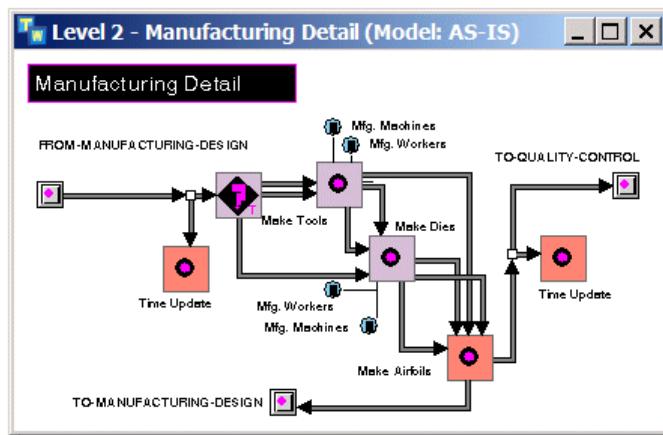
- 5 Close the Determine Problem detail and the Engineering Analysis detail.

# Exploring the Manufacturing Detail

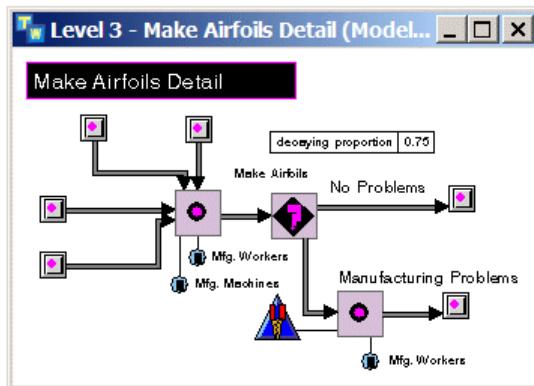
The feedback relationship between Manufacturing Design and Manufacturing is similar to that of Engineering Design and Engineering Analysis. Manufacturing Design creates in-process-shape objects. When problems occur while making airfoils in Manufacturing, in-process-shape objects need to be redesigned, and Manufacturing Design becomes active again.

## To explore the Manufacturing task:

- 1 On the Turbine Blade As Is Model detail, display the Manufacturing detail:



- 2 Display the Make Airfoils detail:



Observe the similarities of this detail and that of Check Cold FEM. The same type of custom Branch block controls when problems occur.

- 3 Close the Make Airfoils, Manufacturing, and Turbine Blade As Is Process details.

## Summary

To summarize the process in the As Is model:

- During long portions of the process only one group of blocks can work while the rest wait.
- Steps repeat themselves in loops, sequentially.
- There is no interaction between Manufacturing and Engineering.

The Vision Model takes a different approach. The process will repeat at least three times, refining the design and getting feedback along the way.

# Exploring the Vision Model

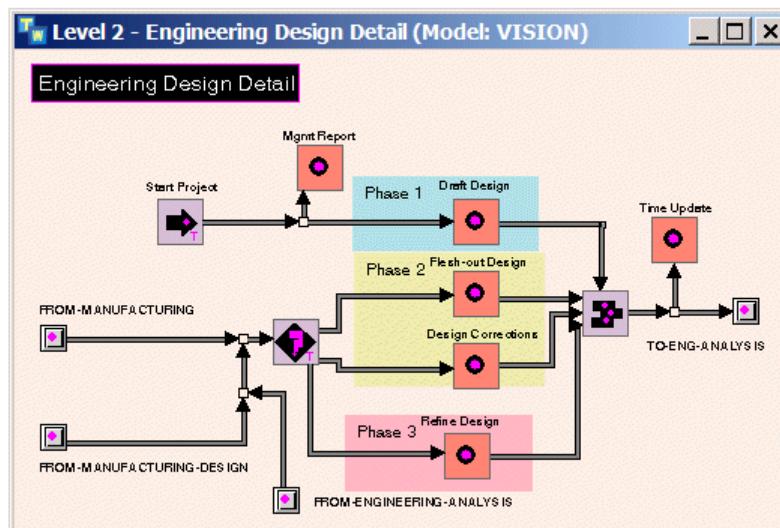
Now, you will look at the components of the Vision model.

## Exploring the Engineering Design Detail

You will compare the Engineering Design detail of the Vision model with that of the As Is model.

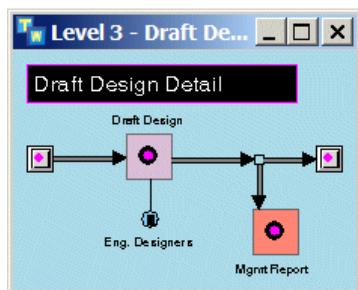
**To explore the Engineering Design task:**

- 1 On the Turbine Blade Design Process workspace, display the detail of the Vision model.
- 2 Display the Engineering Design detail of the Vision Model:

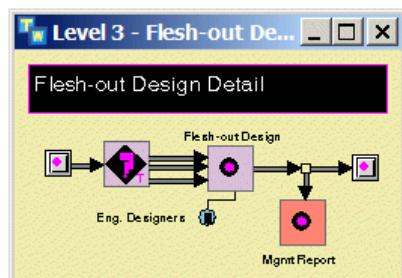


The three colored boxes highlight the three phases of the Vision approach. The first phase develops high-level designs for the purpose of getting quick feedback.

- 3 Show the detail of the Phase 1 Draft Design, which is on a blue background:



- 4 Observe the duration of the Draft Design task, which is 3 weeks.  
Compare this with the corresponding Initial Design task from Engineering Design in the As Is model, which is 26 weeks. Also note that redesign does not occur until Phase 2.
- 5 Close the Draft Design Detail and display the Flesh Out Design detail:



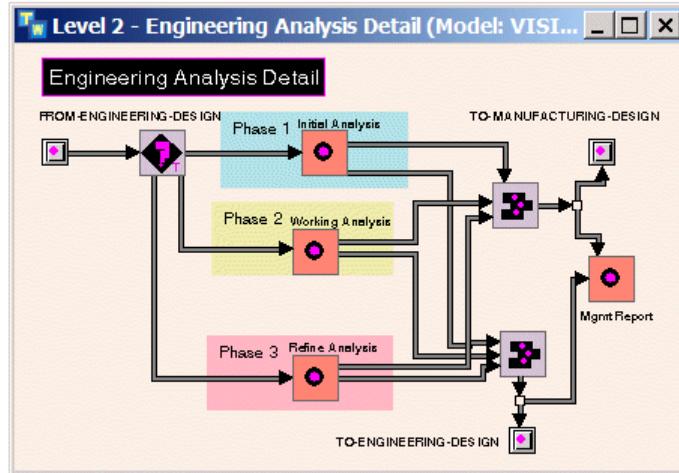
Here you can see three feedback paths, which carry objects from different parts of the model. The Branch block separates objects based on their type.

- 6 Examine the Branch block's Mode attribute, which is Type.
- 7 Examine the Type attribute of each of the Branch block's output paths.  
Initial-tools-report, initial-conversion-problems, and initial-deflection objects come here to be “fleshed out” from Manufacturing and Engineering Analysis stages.
- 8 Close Flesh Out Design and Engineering Design detail.

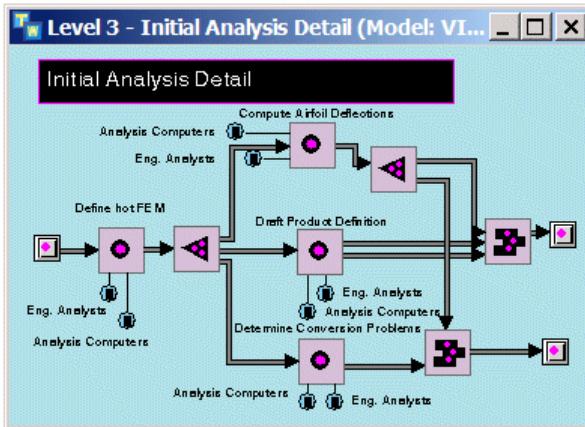
# Exploring the Engineering Analysis Detail

To explore the Engineering Analysis task:

- 1 On the Turbine Blade Vision Process workspace, display the detail of Engineering Analysis:



- 2 Display the detail of the Phase 1 Initial Analysis detail:



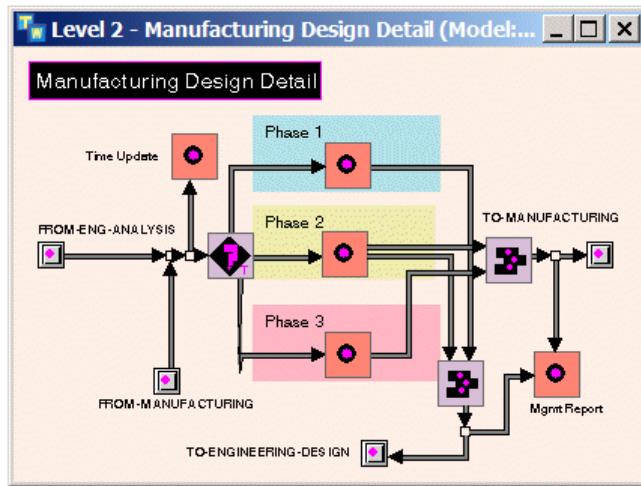
Many steps are not included in Phase 1, such as cold FEM analysis. Phase 1 is only concerned with studying feasibility, to catch problems in advance.

- 3 Close Initial Analysis Detail and Engineering Analysis detail.

# Exploring the Manufacturing Design Detail

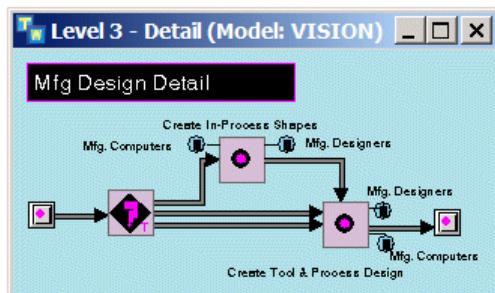
To explore the Manufacturing Design task:

- 1 On the Turbine Blade Vision Process workspace, display the Manufacturing Design task detail:



The three phases are here as well. Each consecutive phase assumes a higher level of assurance and completeness of the objects it works with.

- 2 On the Manufacturing Design detail, show the Phase 1 Task block's detail:

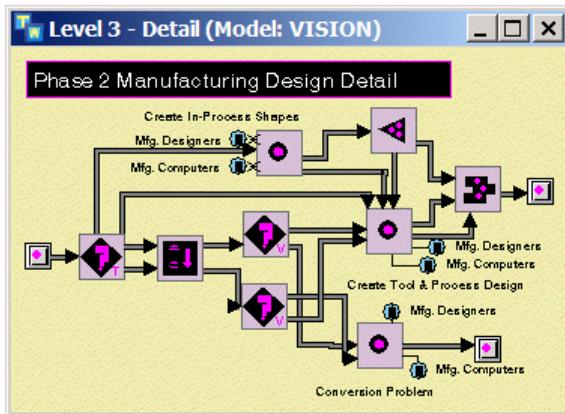


- 3 Examine the Type attribute of each output path of the Branch block on the left.

By now, you should be able to see a pattern: each Phase 1 path carries a work object whose type has the “initial” prefix, such as initial-deflections.

- 4 Close the Mfg Design Detail for Phase 1.

- 5 View the detail of the Phase 2 Task block on the Manufacturing Design detail:



- 6 Examine the Type attributes of some of the output paths from the Branch blocks on the Phase 2 Manufacturing Design Detail.

These paths are carrying the same types of objects as in Phase 1, except that the type of these work objects have the “working” prefix, such as working-deflections. These objects have already been approved to a certain degree by Phase 1 from other stages. Phase 3’s paths carry objects whose type has the “refine” prefix, which indicates an even higher level of confidence.

- 7 Close the Phase 2 Manufacturing Design Detail and return to the higher level Manufacturing Design Detail.

Observe the additional input and output paths, which are indicative of the increased feedback from different stages. The work is distributed across many tasks and resources at any given time, in contrast to the As Is model.

- 8 Close the Manufacturing Design detail.

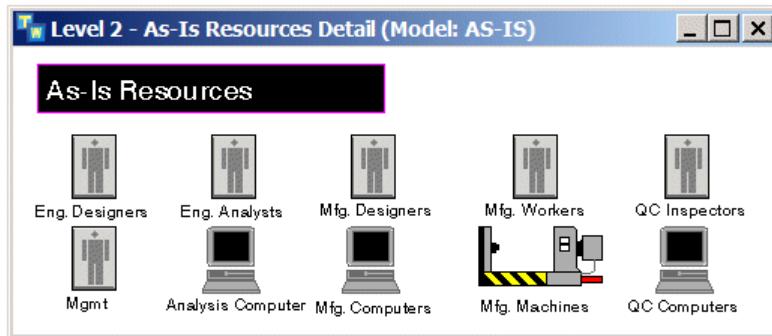
## Observing the Resources

If the simulation is running and neither the Vision nor As Is model has finished, you should observe that the Vision model often has several high-level Task blocks activated simultaneously. You can view the resources in action while the model is running.

### To observe the resources:

- 1 Display the Turbine Blade As Is Process and Turbine Blade Vision Process details.
- 2 Show the details of the resource pools for both versions of the model, labeled As Is Resources and Vision Resources, respectively.

Here are the resources for the As Is model:



The icons change color when they are allocated.

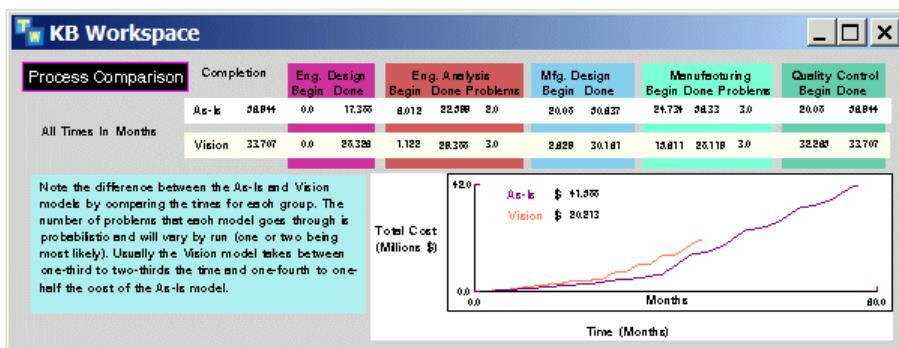
For more information about the utilization, cost, and other attributes of the individual resources, show the properties dialogs for the various resources.

- 3 Close the resource pool workspaces.

## Viewing the Results

### To view the results of the two models:

- 1 Show the Output Tables organizer detail from the top-level workspace:



As the model progresses, you will see values for the Begin and Done columns of each stage. Observe how quickly the Vision model started the Analysis process compared to the As Is model. Also, notice that the Vision process

begins plotting before the As Is model and initially costs more than the As Is model. However, as the simulation progresses, notice that the Vision model is complete significantly before the As Is model, which means the Vision model costs significantly less than the As Is model. In fact, the Vision model usually costs a fourth to a half the amount of the As Is process, and it finishes in about half the time.

In the simulation shown in the output table above, the As Is model took approximately 55 months to complete, whereas the Vision model took only 31 months. Also, the As Is model cost approximately \$41 million, compared with \$19 million for the Vision model.

- 2** Close the Output Tables organizer detail.

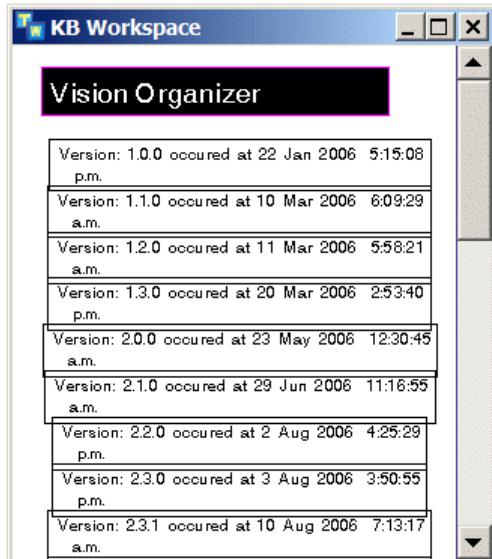
## Organizing Versions

One complication imposed by the three phases of the Vision model is managing the many revisions during each stage. The Vision Organizer helps keep track of versions of the blade's components.

**To see how the Vision model is organized:**

- 1** On the Turbine Blade Vision Process workspace, show the detail of the Vision organizer.

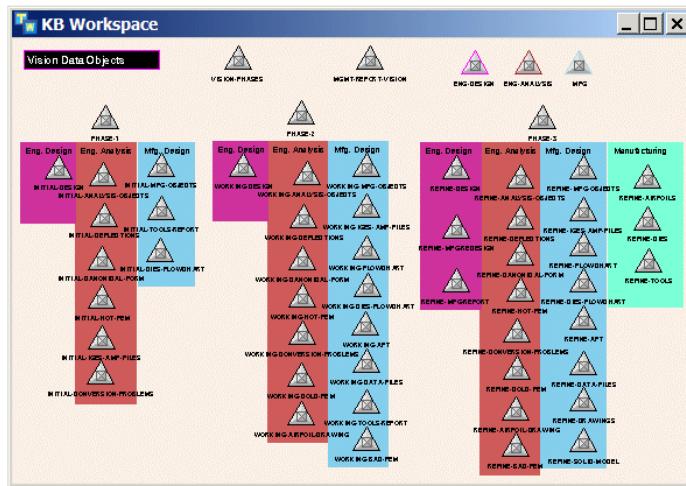
Here is the Vision Organizer for a sample simulation:



Here you see messages indicating the time each version was produced. The model stamped versions at the beginning of each phase and when they passed between tasks. To see just where this happens, you can explore the Mgmt

Report tasks throughout the model, and you can look at the instruments that do the stamping. In the Vision model, special attributes of management reports keep track of how many revisions take place in each revision category. The categories are Super, Major, or Minor revisions.

- 2** Close the Vision Organizer's detail.
- 3** Display the Vision Data Objects organizer detail:



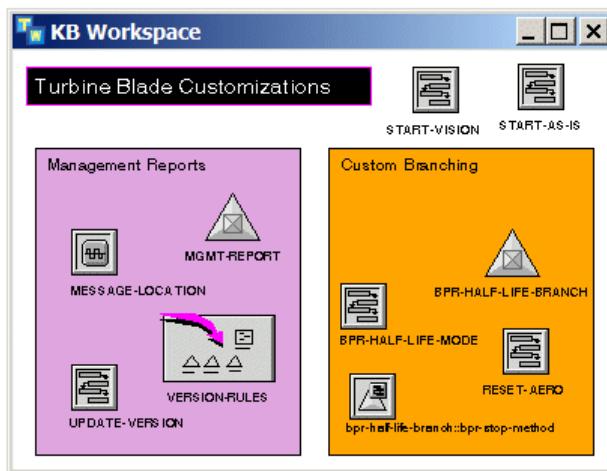
- 4** Look at the class-specific attributes of the mgmt-report-vision class.  
An attribute exists for each type of revision.

# Using Customizations

This model uses several customizations for branching and creating management report versions.

## To view the customizations:

- 1 On the top-level Turbine Blade Design Process workspace, display the detail of the Customizations organizer:



- 2 Display the properties dialog for the bpr-half-life-branch class definition in the Custom Branching area.

Notice the value of the Direct Superior Classes attribute at the top of the table, which shows that the class definition is a subclass of the bpr-branch class.

Customizations control branching and versions in management reports. In the case of branching, a new type of Branch block, bpr-half-life-branch, models complicated branching situations, such as the decaying branch probabilities we encountered earlier. For a closer look, click the Branching Notes memo in the orange Custom Branching box.

Management report versions are more subtle. The **version** attribute of mgmt-report-vision objects is a string of the "X.Y.Z" format where X, Y, and Z represent super, major, and minor revisions, respectively.

The rules located in the Version Rules organizer trigger the update-version procedure when any kind of revision occurs. This procedure manipulates the version string appropriately for each revision. For example, a minor revision after version "2.0.9" leads to version "2.0.10." A major revision, however, yields version "2.1.0," and a super revision produces "3.0.0."

- 3 Close the Customizations organizer detail.

# Viewing the Object Hierarchy

**To view the object hierarchy:**

- ➔ Study the contents of the data object organizers labeled As Is Data Objects and Vision Data Objects for both the As Is and Vision model.
- or
- ➔ Show the class hierarchy for the **bpr-object** class by using Tools > Class Hierarchy.

Note the object hierarchy. Look at the multiple inheritance of the work objects of the Vision model. Each is a subclass of both a development stage, such as Manufacturing, and a phase.

## Typical Questions and Answers

**Q How was this model built?**

- A** The serial model was constructed from a set of IDEF diagrams and verified with actual process measurements.  
The concurrent model was developed as a design for a new process. Since the tasks themselves did not change significantly, only the ordering, there is a high degree of confidence in the new results.

**Q How long did this model take to develop?**

- A** About a week of knowledge gathering and another week to implement and test. Cosmetic changes were necessary for demo purposes, which took an extra few days.

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

---

## A

**activity:** The amount of work associated with processing the inputs of a single block. Each time work objects flow to the input paths of a block, the block creates an activity object. Each activity adds value to the work object; activities have a duration and they can have an associated cost.

**allocate:** To assign a resource to an activity. When the activity is complete, the resource is typically deallocated and can be allocated to another activity. You can allocate a resource for multiple sequential activities. You can allocate particular resources to an activity based on various criteria, such as cost, utilization, or priority. You can also constrain the availability of resources, using temporal constraints.

**annotation:** A tool that contains detail for adding documentation to a model.

**attribute display:** Text displayed next to an icon that shows the value of a particular attribute of the object. For example, all blocks show the Label attribute as an attribute display, whose default value is an empty string.

## B

**block:** A graphical object that represents an operation within your model. Blocks operate on work objects. Blocks can have a duration and a cost. Blocks can also require resources. ReThink represents each block execution as an activity.

## C

**chart:** A display of historical values that a probe in the model collects. *See also probe.*

**class definition:** The definition of an object that defines its characteristics and behaviors. *Contrast with instance.*

**connection posts:** Objects that connect the blocks on one workspace to the blocks on the detail of a task. Work objects flow from the top-level block, through the connection post, to the blocks on the detail, and back up through the connection post to the top-level blocks.

**container:** A subclass of work object that defines an attribute into which the Insert block and the Batch block insert work objects and from which the Remove block removes work objects. A container is an instance of the bpr-container class.

**cost:** The sum of all fixed and variable costs associated with tasks and resources, which are applied to different objects in the model. For example, process costs measure costs as they apply to work objects, task costs measure costs as they apply to individual blocks, and resource costs measure costs as they apply to resources or resource pools that are allocated to a task.

**creation time:** The simulation time at which a ReThink object was created. The creation time of some ReThink objects, like blocks, is typically the start time of the simulation, while the creation time of other ReThink objects, like work objects, can be any time during the simulation when the work object is created.

**cycle time:** The amount of simulation time from one operation in a model to another. Cycle time is one way of measuring the performance of a process or subprocess. Cycle time is also called delta time.

## D

**deallocate:** To finish using a resource for an activity. When a Resource Manager deallocates a resource, the manager can allocate it to some other activity.

**detail:** The subworkspace or associated workspace of any ReThink object. For example, a Task block can have a detail to define its subprocesses, and a Model tool has a detail to display the model. Task block details allow work objects to flow from the top-level task, through the blocks on the detail, and then back up to the top-level blocks. A detail can contain multiple levels deep of subworkspaces.

**developer:** A ReThink user who uses G2, Gensym's core technology upon which ReThink is built, to create custom ReThink objects. Developers create new ReThink objects based on existing ReThink objects, which are specific to a particular application. Developers work in Developer mode. *Contrast with modeler and end user.*

**discrete event simulation:** A paradigm for process modeling where a model consists of a set of sequential tasks, which create and process work objects as discrete events. The discrete simulation clock advances with each new discrete event.

**duration:** The amount of simulation time applied to a particular event. ReThink computes duration differently for different types of objects. The duration of most blocks is computed based on a random normal function and represents the amount of simulation time the block has been processing work objects. The duration of a resource is the amount of simulation time the resource has been allocated to activities in the model. The duration of a work object is the amount of simulation time the blocks in the model have spent processing the work object.

## E

**elapsed time:** The amount of simulation time that has passed since a particular ReThink object was created. *Contrast with work time.*

**end user:** A ReThink user that applies an existing model to understand a given business process. End users use “what-if” analysis to do business process reengineering. End users work in user mode. *Contrast with* modeler and developer.

## F

**feed:** A type of instrument that you use to assign values to objects in a model while the model is running. You can feed values into attributes of blocks, work objects, and resources. *See also* sliders and type-in boxes.

**free text:** Text that you can place anywhere on a workspace to label the model.

## H

**hierarchical view:** A way of viewing a model such that only the relevant information is visible. You create hierarchical views in a model by adding detail to Task blocks. You can create nested levels of detail in a hierarchical view, as needed.

## I

**indicator:** A large magenta arrow that a Scenario tool displays next to an object when the user chooses certain menu choices, such as Show Scenario on any object or Choose Resource on a Resource Manager. The Scenario tool controls whether clicking the indicator removes it from the workspace (the default) or whether the indicator disappears automatically.

**instance:** A specific occurrence of a class of objects. For example, the work objects that a block processes are instances of the **bpr-object** class, or a subclass of this class. *Contrast with* class.

**instrument:** A type of ReThink object that either supplies values to a model or obtains values from a model. Feeds and probes are the two classes of instruments within ReThink. From ReThink instruments, you create user interface objects to observe and control your model. You create charts from probes and sliders, and you create type-in boxes from feeds.

## J

**jump mode:** The normal discrete event simulation mode. Events occur while a model is running. After each event, ReThink advances the simulation clock to the starting time of the next event. Work objects flow through the model continuously. *Contrast with* step mode, synch mode, and online mode.

## K

**knowledge base (KB):** Any G2 application, such as a ReThink model that you create or any of the files that are required to run a ReThink model. A knowledge base is stored in a file with a `.kb` extension. A knowledge base can consist of a single `.kb` file or an entire set of files that make up the application.

## M

**model:** A representation of some part of a business process that you create by using ReThink blocks, instruments, and resources. You place ReThink models on the detail of a Model tool.

**modeler:** A ReThink user who creates models by cloning, connecting, and configuring blocks, instruments, and resources. Modelers work in modeler mode. *Contrast with end user and developer.*

**module:** A set of related information in an application. Typically, you store a single module in a single `.kb` file that has the same name as the module.

## O

**object-oriented representation:** A way of representing information in a computer model, where each distinct piece of information is an “object” that has properties, behaviors, and relationships to other objects. The properties take the form of attributes, the behaviors take the form of methods and procedures, and the relationships take the form of connections and relations. You define an object by creating a class, and you specify the particular characteristics of the object by creating an instance.

**online mode:** The mode you use for real time transaction processing. *Contrast with jump mode, step mode, and synch mode.*

**organizer:** A type of tool with detail on which you can place various types of objects, for example, resources and work object definitions.

## P

**parameter:** Inputs to a model that determine how the model behaves. For example, the mean time between orders, the hourly wage of a clerk, and the number of boxes in a truckload are all parameters of the model. You use ReThink feeds to create sliders and type-in boxes to supply parameters to the model.

**path:** A connection between two blocks. Paths carry work objects from block to block within a model. Paths have a direction of flow, typically from left to right. Blocks can have input paths and output paths. You determine the type of work object that a block processes by specifying the path type. Paths automatically reconfigure when you move one of the connected blocks. *See also stub.*

**path queue:** A list that contains pending work objects for a particular task. When a block has constraints or when a block synchronizes its inputs, input paths can develop a backlog, called a work backup. Paths keep track of queuing statistics, such as wait time, making it easy to identify and diagnose bottlenecks in a process.

**path type:** The type of work object that a block receives on its input path or passes on its output path. The path type can be `bpr-object` or `bpr-container`, or any subclass of these classes.

**pool:** *See* resource pool.

**probe:** A type of instrument that obtains values of objects while a model is running. You can obtain performance information about blocks, work objects, resources, and activities, using probes. You create charts directly from probes to plot the history of probed values.

## R

**readout table:** A display that shows the current value of an attribute of an object, which updates at a periodic interval. You typically use readout tables to display attributes of paths.

**remote:** An intermediate object that ReThink creates when you create a chart from a probe. You configure the remote to specify how the chart looks and behaves.

**report:** Summary statistics for blocks, paths, probes, resources, and work objects, presented as a spreadsheet. Reports contain the current values of all the system-generated attributes for the particular type of object as of the moment the report is created or refreshed.

**resource:** An object that a particular block requires to process its work objects. Resources constrain the model based on availability. You can also apply costs to a model through resources. You use Resource Managers to allocate and deallocate resources for particular tasks.

**resource manager:** Determines which resources a block uses to perform a particular task and how the block allocates and deallocates them. You create Resource Managers directly from resources and attach them to blocks that require the particular resource. By default, the Resource Manager chooses resources from a pool at random and schedules the blocks that are waiting for the resource when the resource is unavailable. The Resource Manager also determines the utilization of the resources, which is the amount of resources that are required.

**resource pool:** A collection of resources available for a particular task. To create a pool for any resource, create detail for the resource and add resources. Modelers and end users can add, change, or remove resources while the model is running to explore the performance implications of different resource constraints.

## S

**scenario:** The control center of a model. You control a model by creating an associated scenario, which keeps track of the simulation time of the model. The scenario advances the simulation clock for each event and either continues running the model or pauses, depending on the mode in which you are running the model. You can use one scenario to control multiple models, or you can use different scenarios to control the same model.

**simulation time:** The current time at which the model is running. Scenarios control when the simulation time advances, and the duration of activities determines how much it advances for each event.

**slider:** A user interface display that allows end users to provide parameters to the model by sliding a value along a number line. Sliders enable end users to perform “what-if” analysis. You create sliders from feeds.

**stand-alone model:** A model that runs independently of other models in a knowledge base. You create a stand-alone model by placing a Scenario tool on the detail of a Model tool with the blocks, resources, and instruments that make up the model.

**step mode:** The mode used for diagnosing and debugging the model. ReThink pauses after each event so that you can walk through the simulation one step at a time. When you continue running, the simulation clock immediately advances to the starting time of the next event. *Contrast with* jump mode, synch mode, and online mode.

**stub:** A connection coming into or out of a block that is not yet connected to another block. You can connect stubs to other stubs, directly to a block, or to a junction. *See also* path.

**subclass:** A class of objects that inherits its definition from another class of object. When processing work objects in a ReThink model, you automatically create subclasses of the built-in bpr-object class by specifying output path types.

**surrogate:** A different manifestation of an existing resource, usually placed in a different pool, that the model uses in another location. You use surrogates to share an individual resource among more than one task. For example, in a model of a delivery process, a truck loader might also act as a truck driver.

**symbol:** A string of alpha-numeric characters without spaces. Use hyphens in place of spaces in any symbol. All user-defined names and attributes must be symbols in ReThink.

**synch mode:** The mode used to help visualize the delays in the process. ReThink scales the time of the simulation to real time. For example, you can use this mode to run the simulation at one hour simulation time per second of real time. Most of the time when you are building models, however, you focus on diagnosing the work flows and let the clock keep track of the simulation time. *Contrast with* jump mode, step mode, and online mode.

## T

**task:** A distinct operation within a process that adds value and applies costs to work objects. ReThink represents tasks dynamically as activities.

**temporal constraints:** Objects that allow you to constrain the date and time availability of resources. To specify the availability of resources, you connect a temporal scheduler to any resource and configure the constraints on the detail of the scheduler. You can configure the availability of resources by the date, month, week, and day.

**top-level workspace:** A named workspace that you create and which you can display by using the Workspace menu. You can assign a top-level workspace to its own module thereby allowing you to split a model across multiple modules.

**type-in box:** A user interface display that allows end users to provide parameters to the model by typing a value into a box. Type-in boxes allow end users to perform “what-if” analysis. You create type-in boxes from feeds.

## U

**utilization:** The amount of a resource allocated to an activity. You specify utilization in the Resource Manager that allocates resources to a task. You specify the amount of the resource that is available for a task in the resource itself. Resources and work objects compute various utilization statistics to analyze the efficiency of the process.

## W

**“what-if” analysis:** A technique of business process reengineering in which you experiment with different parameters, resource constraints, costs, and work flows, to come up with a more efficient process. ReThink enables “what-if” analysis by providing charts, sliders, and type-in boxes for supplying new values to the model, as well as by providing the ability to add and delete resources interactively while the model is running.

**work backups:** Work objects that ReThink places in the path queue. Work backs up in a process when a block is too busy processing its current inputs to process the work in the queue. This happens when resource constraints exist, when the maximum number of activities for a block is specified, or when the block synchronizes its inputs.

**work flow analysis:** The process of analyzing how work objects move from task to task in a process. Depending on the type of blocks that the model uses, bottlenecks can occur that cause inefficiencies. By experimenting with different types of blocks, you can reengineer how work flows through the process to make it more efficient. For example, you can experiment with sequential processing rather than synchronized processing.

**work object:** An object in a model on which blocks operate. Work objects represent the inputs and outputs of a process, for example, orders and invoices. Work objects compute summary duration and cost statistics that indicate the overall performance of the process.

**work time:** The amount of simulation time that an object has actually been active. ReThink computes work time differently depending on the type of object. The work time of a work object is the amount of time the object has been actively operated on by blocks in the process. The work time of a resource is the amount of time the resource has been allocated. The work time of a block is the amount of time the block has been actively processing work objects. *Contrast with* elapsed time.

**workspace:** *See* top-level workspace.

@	A	B	C	D	E	F	G	H	I	J	K	L	M
#	N	O	P	Q	R	S	T	U	V	W	X	Y	Z

---

## A

activities  
*aero.kb* model  
allocating resources  
annotating models  
  using labels  
  using tools  
annotations

## costs

configuring  
measuring  
creating  
  resource managers  
  resource pools  
  work objects  
creation time  
customer support services  
cycle time

## B

Basic Activities palette  
Basic Activities tab  
blocks  
  adding to details  
  configuring  
    costs  
    duration  
    getting started  
    work objects to process  
  connecting  
  creating  
  getting started with  
  labeling  
  process modeling paradigm  
building  
  models  
  top-level models

## D

deallocating resources  
details  
  adding  
    blocks to  
    connection posts to  
    nested levels to  
  example of  
    order processing task  
    payment task  
model  
modeling  
  from the top down, using  
  initial  
  modeling task  
dialogs  
  showing summary statistics, using  
discrete  
  event simulation  
  simulation clock  
Distribution Mode attribute  
duration  
  configuring  
    building process models  
    getting started

## C

charts  
configuring  
  blocks  
  costs  
  duration  
connection posts  
  adding to details  
  on Task block details  
  order processing task detail, example of  
containers  
Continue button

## E

elapsed time

## F

feeds  
free text

## G

green paths

## H

hierarchical views

## I

indicator arrows  
inputs  
instruments  
computing statistics, using  
process modeling paradigm

## J

jump mode  
Jump Mode menu choice  
running the simulation, using

## K

knowledge base

## L

Label attribute  
configuring

## M

managers, resources  
Mean attribute  
metrics  
metrics, computing  
mode, scenario  
Model tools  
organizing applications, using  
using  
modeling  
resources  
task details

models

- annotating, using tools
- building
  - determining the process
  - organizing the model
  - top-level
- cloning
- details of
- hierarchical
  - building
  - example
- opening
  - existing
  - order fulfillment model
  - turbine blade model
- organizing
  - building process models
  - getting started with
- saving
  - building process models
  - getting started with
- modules
  - ReThink hierarchy of

## N

navigator

## O

object-oriented representation  
objects  
creating work  
Open menu choice  
loading an existing model, using  
order fulfillment model  
automating the process  
constraining, using resources  
details

- order processing task
- payment task
- modeling the details of
- opening
- running
  - under high capacity
  - under normal conditions
- top-level workspace of
- viewing definitions for

Organizer tools  
order fulfillment model, example of

using  
organizers  
  for model definitions  
  for resources  
  for storage pools  
  organizing models, using  
organizing models  
outputs  
  obtaining  
process modeling paradigm

**P**

path queue  
paths  
  configuring work objects to process on  
  process modeling paradigm  
pools, resource  
posts, connection  
probes

**Q**

queue, path

**R**

readout tables  
reports  
  obtaining outputs, using  
  process modeling paradigm  
resource managers  
resources  
  allocating  
  configuring while running  
  creating pools  
  deallocating  
  modeling  
    building process models  
    order fulfillment model  
  organizers for  
  pools  
  process modeling paradigm  
Resources palette  
ReThink  
  applications, organizing  
  benefits of  
  business process modeling, using  
  getting started tutorial  
  introduction to

modeling a complete process, using  
modules of  
paradigm of  
required knowledge for  
  end users  
  experts  
  general  
  modelers  
saving models in  
users of  
  developers  
  end users  
  modelers  
ReThink toolbox  
*rethink-online.kb* file

**S**

Save menu choice  
  saving to the default filename, using  
saving models  
Scenario tools  
  creating  
  using  
scenarios  
  activating  
  creating  
  getting started with  
Show Detail menu choice  
  viewing order processing detail, using  
  viewing payment detail, using  
Shut Down G2 menu choice  
simulation  
  clock  
  discrete event  
  time  
simulations  
  resetting  
  running  
    in jump mode  
    in step mode  
    starting  
sliders  
Snapshot Activities menu choice  
Snapshot Queue menu choice  
stand-alone model  
Standard Deviation attribute  
status, scenario  
Step Mode menu choice  
  running simulations, using

storage pools  
subclasses  
surrogates

## T

Task block  
    with details  
        initial  
        modeling  
temporal constraints  
Tools  
    tab  
tools  
    model  
    organizer  
    scenario  
Tools palette  
top-level  
    models  
    workspaces  
tutorials  
    getting started  
    order fulfillment model  
    turbine blade model  
Type attribute  
type-in boxes

## U

utilization

## W

work backups  
    process modeling paradigm  
    viewing  
work flow analysis  
work objects  
    automatically creating definitions for  
    creating  
    getting started with  
    process modeling paradigm  
    types of  
work time  
workspaces  
    top-level