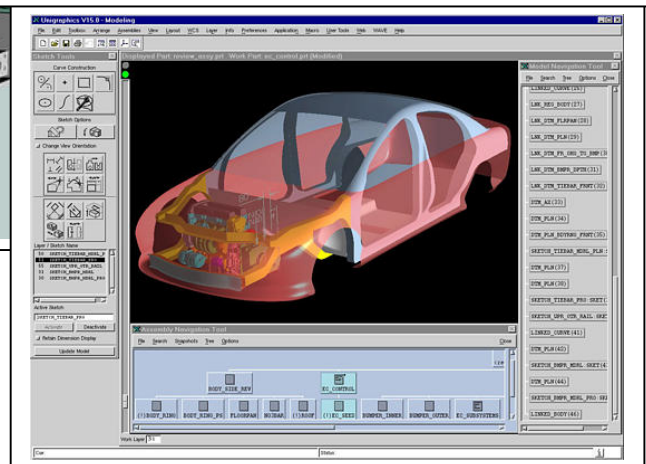
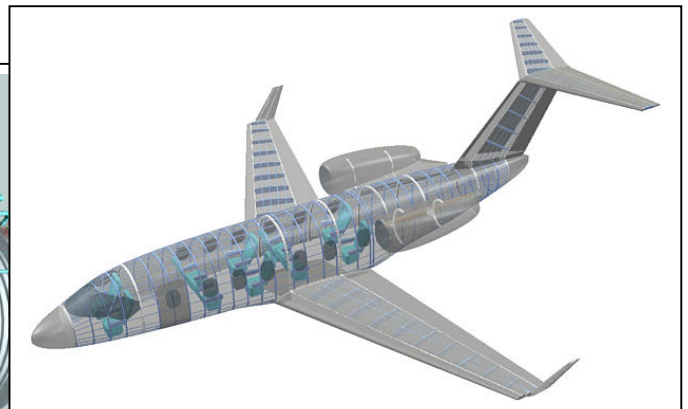
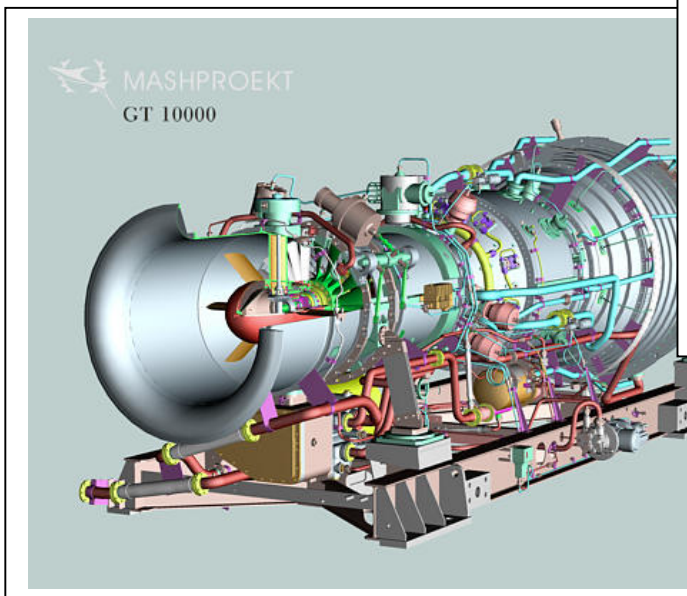




## ***UG/ Knowledge Fusion for Designers V18***



***KNOWLEDGE FUSION FOR DESIGNERS***  
***STUDENT GUIDE***  
***November 2001***  
***MT15130 – Version 18.0***

---

## *Proprietary & Restricted Rights Notices*

---

### **Copyright**

Proprietary right of Unigraphics Solutions Inc., its subcontractors, or its suppliers are included in this software, in the data, documentation, or firmware related thereto, and in information disclosed therein. Neither this software, regardless of the form in which it exists, nor such data, information, or firmware may be used or disclosed to others for any purpose except as specifically authorized in writing by Unigraphics Solutions Inc. Recipient by accepting this document or utilizing this software agrees that neither this document nor the information disclosed herein nor any part thereof shall be reproduced or transferred to other documents or used or disclosed to others for manufacturing or any other purpose except as specifically authorized in writing by Unigraphics Solutions Inc.

©2002 Unigraphics Solutions Inc. All rights reserved.

### **Restricted Rights Legend**

The commercial computer software and related documentation are provided with restricted rights. Use, duplication or disclosure by the U.S. Government is subject to the protections and restrictions as set forth in the Unigraphics Solutions Inc. commercial license for the software and/or documentation as prescribed in DOD FAR 227–7202–3(a), or for Civilian Agencies, in FAR 27.404(b)(2)(i), and any successor or similar regulation, as applicable. Unigraphics Solutions Inc., 10824 Hope Street, Cypress, CA 90630.

### **Warranties and Liabilities**

All warranties and limitations thereof given by Unigraphics Solutions Inc. are set forth in the license agreement under which the software and/or documentation were provided. Nothing contained within or implied by the language of this document shall be considered to be a modification of such warranties.

The information and the software that are the subject of this document are subject to change without notice and should not be considered commitments by Unigraphics Solutions Inc.. Unigraphics Solutions Inc. assumes no responsibility for any errors that may be contained within this document.

The software discussed within this document is furnished under separate license agreement and is subject to use only in accordance with the licensing terms and conditions contained therein.

### **Trademarks**

**UNIGRAPHICS SOLUTIONS®**, **UNIGRAPHICS®**, **GRIP®**, **PARASOLID®**, **UG®**, **UG/...®**, **UG SOLUTIONS®**, **iMAN®** are trademarks or registered trademarks of Unigraphics Solutions Inc.

*Knowledge Fusion for Designers Student Guide* Publication History:

Version 18.0 ..... November 2001

# Table of Contents

---

## Course Overview

.....	–1
Course Description .....	–1
Intended Audience .....	–1
Prerequisites .....	–1
Objectives .....	–2
How to Use This Manual .....	–3
Classroom System Information .....	–4
Class Part File Naming .....	–5

## Introduction to Knowledge Fusion ..... 1–1

About This Class .....	1–2
KBE Overview .....	1–2
What is Knowledge Based Engineering (KBE)? .....	1–2
Why use KBE? .....	1–3
Difference between KBE and Parametrics .....	1–3
KBE Example .....	1–4
Knowledge Fusion Overview .....	1–6
What is Knowledge Fusion? .....	1–6
Why Use Knowledge Fusion? .....	1–7
Who will use Knowledge Fusion? .....	1–7
Basic Concepts of Knowledge Fusion .....	1–8
Knowledge Fusion is Declarative and Demand Driven ...	1–8
Knowledge Fusion is Object Oriented .....	1–9
Knowledge Fusion is Hierarchical .....	1–9
Geometry .....	1–9
Some Key Definitions .....	1–10
What is a Knowledge Fusion Class? .....	1–10
What is a Knowledge Fusion Object? .....	1–10
What is a Knowledge Fusion Attribute? .....	1–11
What is a Knowledge Fusion Rule? .....	1–11
Knowledge Fusion Toolbar .....	1–12
Knowledge Fusion, Parametrics and WAVE .....	1–13
Activity 1–1: What Do You Think .....	1–14
Activity 1–2: The Bike Frame Project Tour .....	1–15

## Knowledge Fusion Navigator ..... 2–1

KF Navigator Overview .....	2–2
-----------------------------	-----

MB3 Pop-up Menus .....	2-5
<b>Activity 2-1: Knowledge Fusion Navigator Tour .....</b>	<b>2-7</b>
Add Attribute .....	2-15
<b>Activity 2-2: Add Attributes in the KF Navigator .....</b>	<b>2-18</b>
Add Child Rule .....	2-23
<b>Activity 2-3: Add Child Rule in the KF Navigator .....</b>	<b>2-27</b>
KF Classes .....	2-33
<b>Activity 2-4: Instantiating a User Defined Class .....</b>	<b>2-34</b>
<b>Activity 2-5: Instantiating a UG System Class .....</b>	<b>2-38</b>
<b>Design Control .....</b>	<b>3-1</b>
Controlling Topology .....	3-2
<b>Activity 3-1: Control the Door to Have a Dynamic Window .....</b>	<b>3-5</b>
Controlling Color, Layer, Suppression Status .....	3-7
Mass Properties .....	3-8
<b>Activity 3-2: Examine the Mass Properties of the Door ..</b>	<b>3-9</b>
Controlling Interactive Edit .....	3-11
<b>Activity 3-3: Control Interactive Edits of the Door .....</b>	<b>3-13</b>
KF and the Expression Tool .....	3-17
<b>Activity 3-4: Use UG Expression to Control a Block .....</b>	<b>3-18</b>
<b>Adoption .....</b>	<b>4-1</b>
The Adoption Concept .....	4-2
Adopt Existing Object .....	4-3
The Procedure of Adoption .....	4-5
<b>Activity 4-1: Adopting Existing Objects .....</b>	<b>4-6</b>
<b>Activity 4-2: Adoption with Blend Position Change .....</b>	<b>4-12</b>
<b>User Defined Features .....</b>	<b>5-1</b>
UDF Overview .....	5-2
Exporting User Defined Features .....	5-3
<b>Activity 5-1: Create a Knowledge Enabled Boss UDF .....</b>	<b>5-5</b>
UDF Positioning Strategies .....	5-8
Reference Frames .....	5-8
Reference Geometry .....	5-10
<b>Activity 5-2: Reference Frame Positioning of a UDF .....</b>	<b>5-11</b>
<b>Activity 5-3: Reference Geometry Positioning of a UDF ..</b>	<b>5-13</b>
Customizing the UDF Dialog .....	5-18
<b>Activity 5-4: Customizing UDF Dialog .....</b>	<b>5-20</b>
Swap User Defined Features .....	5-27
<b>Activity 5-5: Swapping User Defined Features .....</b>	<b>5-28</b>
<b>Assemblies .....</b>	<b>6-1</b>
UG/KF Assemblies Concepts .....	6-2

Assemblies and the Knowledge Fusion World .....	6-2
Evaluating a rule in another part .....	6-3
Reading a KF attribute in another part .....	6-3
<b>Activity 6-1: Create a Simple Chassis Assembly .....</b>	<b>6-4</b>
<b>Optimization .....</b>	<b>7-1</b>
Optimization Concepts .....	7-2
General Optimization Process .....	7-4
<b>Activity 7-1: Optimize the Volume of a Squeeze Bottle ...</b>	<b>7-5</b>
<b>Spreadsheet Access .....</b>	<b>A-1</b>
Spreadsheet Concepts .....	A-2
The ug_spreadsheet Class .....	A-4
<b>Activity A-1: Use Spreadsheet to Control Block Parameters .....</b>	<b>A-6</b>
<b>Database Access .....</b>	<b>B-1</b>
The ODBC Interface .....	B-2
ug_odbc_database class .....	B-3
ug_odbc_recordset class .....	B-5
ODBC Drivers .....	B-6
<b>Activity B-1: Use a Database to Control Door Properties</b>	<b>B-7</b>
<b>Glossary .....</b>	<b>GL-1</b>
<b>Index .....</b>	<b>IN-1</b>

**(This Page Intentionally Left Blank)**

## *Course Overview*

---

### ***Course Description***

Unigraphics Knowledge Fusion (KF) provides a powerful set of tools for the capture and manipulation of engineering rules and design intent. Knowledge Fusion enables the user to develop applications and control Unigraphics objects via engineering rules that extend beyond a purely geometric nature. With Knowledge Fusion, engineers and designers will be able to construct a knowledge base that is completely reusable.

### ***Intended Audience***

- Designers, Engineers, and end users of Unigraphics Knowledge Fusion applications.
- High–End Unigraphics Users.
- Current and future KBE core application developers.

### ***Prerequisites***

- Practical Applications of Unigraphics course or CAST equivalent.



## **Objectives**

After successfully completing this course, students can understand and use the following in the context of Knowledge Fusion:

- Knowledge Fusion Navigator and the MB3 pop-up menus
- Design control via engineering rules
- Adoption
- User Defined Features
- Assemblies
- Optimization

## ***How to Use This Manual***

It is important that you use the Student Guide in the sequence presented because later lessons assume you have learned concepts and techniques taught in an earlier lesson. If necessary, you can always refer to any previous activity where a method or technique was originally taught.

The format of the activities is consistent throughout this manual. Steps are labeled and specify what will be accomplished at any given point in the activity. Below each step are action boxes which emphasize the individual actions that must be taken to accomplish the step. As your knowledge of Unigraphics increases, the action boxes will seem redundant as the step text becomes all that is needed to accomplish a given task.

### **Step 1 Create a new part and enter the Modeling application.**

- ☐ Choose **File→New**. Enter **\*\*\*\_boss** in the File name box, where **\*\*\*** represents your initials.
- ☐ Choose **OK**.
- ☐ Click MB3 from the graphics window. Choose **Replace View→TFR–TRI**.
- ☐ Choose **View→Model Navigator**.
- ☐ Choose **Application→Modeling**.

## ***Classroom System Information***

Your instructor will provide you with the following items for working in the classroom:

**Student Login:**      **Username:** \_\_\_\_\_

**Password:** \_\_\_\_\_

**Work Directory:** \_\_\_\_\_

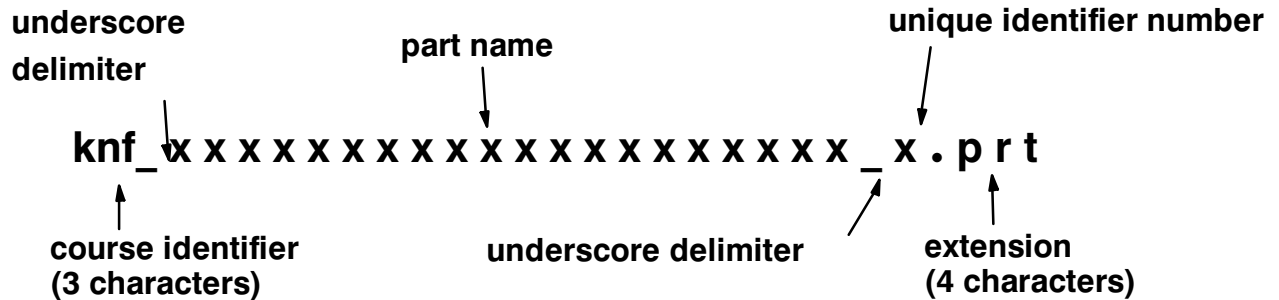
**Parts Directory:** \_\_\_\_\_

**Instructor:** \_\_\_\_\_

**Date:** \_\_\_\_\_

## Class Part File Naming

This course utilizes the following filenames standard:



Where the student is requested to save a part file for later use, the initials of the student's given name, middle name, and surname replace the course identifier "knf" in the new filename with the remainder of the filename matching the original.

**(This Page Intentionally Left Blank)**

# Introduction to Knowledge Fusion

## Lesson 1



### PURPOSE

This lesson will introduce the Knowledge Fusion concepts as well as KBE in general.

### OBJECTIVES

Upon completion of this lesson, you will be able to:

- Gain an initial understanding of Knowledge Based Engineering (KBE).
- Get an initial understanding of Knowledge Fusion.
- Open and instantiate an existing Knowledge Fusion design.

This lesson contains the following activities.

Activity	Page
1–1 What Do You Think . . . . .	1–14
1–2 The Bike Frame Project Tour . . . . .	1–15

## About This Class

1

This is an introductory course for designers and engineers to touch the next generation of CAD/CAM/CAE tool – Knowledge Based Engineering (KBE). Knowledge Fusion is a fully integrated KBE technology within Unigraphics that revolutionize the way an organization looks at solving and automating repetitive engineering problems.

## KBE Overview

### What is Knowledge Based Engineering (KBE)?

Knowledge Based Engineering is fundamentally about **re–use**. It concerns being able to take advantage of any experience, expertise and other information relevant to each phase of the engineering life cycle of an end user product.

These “knowledge bases” can exist in many forms, such as:

- spreadsheets
- handbooks
- engineering formulas
- proprietary software
- human judgement, such as rules of thumb

Being able to create and reference such knowledge bases and make them readily available as an aid to the engineering process constitutes Knowledge Based Engineering. KBE is the key to being able to answer questions that traditional CAD systems have not been capable of addressing, such as:

- “What was the rationale behind this design?”
- “Have any design constraints been violated?”
- “How much will this product cost?”
- “Can this part be manufactured?”
- “Will this part meet its performance goals?”
- “Is this design optimum or are there better alternatives?”

## **Why use KBE?**

Knowledge Based Engineering is intended for use in building engineering automation solutions. This implies that there is a repetitive problem which requires automation.

It is most useful when the solution requires a combination of *configuration* (the selection and assembling of components to make a coherent whole product), *engineering* (the decision –making process about the validity and applicability of the components), and *geometry* (the physical organization of the components).

## **Difference between KBE and Parametrics**

In KBE, knowledge is the driving force, the geometry is driven by configuration and engineering rules. While in Parametric methodology, geometry and dimension driven relationships are the driving force. Another significant difference is that KBE is non –procedural whereas Parametrics is procedural.

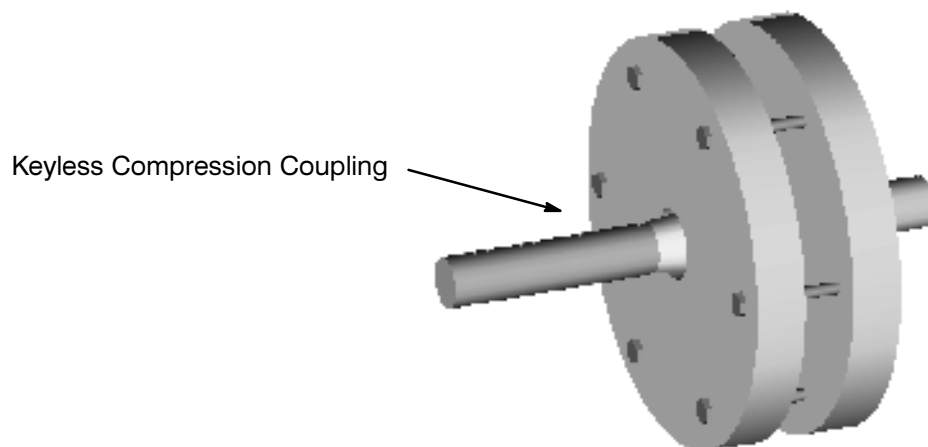
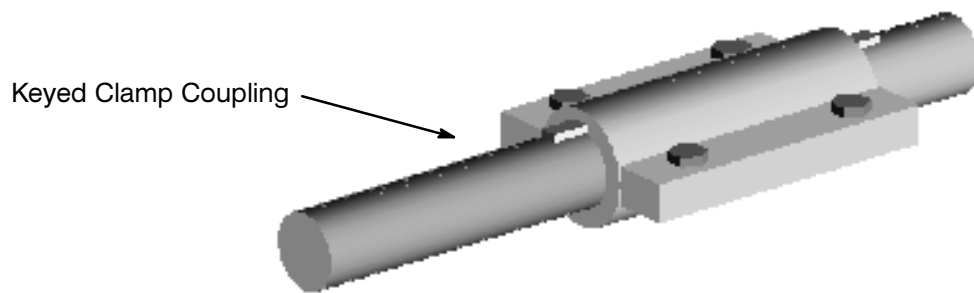
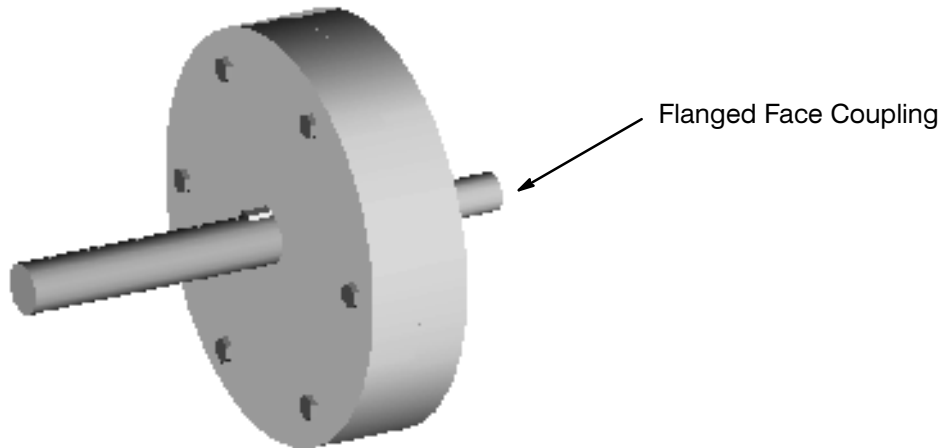




### **KBE Example**

Let's look at a simple KBE example. As shown in the following figure, there are three types of drive coupling available:

- a. Flanged Face Coupling that allows the input and output shafts to have different diameters;
- b. Keyed Clamp Coupling to support higher torque;
- c. Keyless Compression Coupling for low torque applications.



We can setup the automated layout of the coupling using standard engineering inputs. The following are the engineering and configuration rules to control the coupling type:

– torque

$$(33000 * \text{horsepower}) / (2\pi * \text{rpm})$$

– coupling type

If (input\_radius != output\_radius) Then Flanged Face Coupling  
Else If (torque < 100.0) Then Keyless Compression Coupling  
Else Keyed Clamp Coupling

The Knowledge Fusion implementation of this example will be provided as a student activity in the next lesson.



## **Knowledge Fusion Overview**

1

### **What is Knowledge Fusion?**

By incorporating a KBE technology within Unigraphics, Knowledge Fusion provides superior technology that permits UG to take advantage of engineering knowledge bases in conjunction with rules to deliver powerful applications. Through these applications and/or wizards, the end user need not be aware of the underlying KF technology.

The UG Knowledge Fusion language is an interpreted object oriented language that has been designed to permit an end user to easily add engineering knowledge to the task at hand by the creation of rules, which are the basic building blocks of the language. Additionally, the language has the capability to access external knowledge sources, such as databases or external spreadsheets, and to interface to other applications such as analysis or optimization packages.

Knowledge Fusion can also be described as:

- An engineering spreadsheet, that allows you to record and reuse engineering formulas. The formulas can refer to values of other formulas.
- A configuration system for selecting and assembling compatible parts.
- A very powerful computer–aided design (CAD) programming language.
- A system with which you can define and create “smart parts” that can be assembled and/or configured in ways that are not preprogrammed.

The Knowledge Fusion language, combined with the Knowledge Fusion Navigator user interface, can support the end user at a wide range of skill levels. It allows one to go from making simple adjustments to an existing KBE application (by changing parameters or adding dynamic rules to an existing UG part file), to the creation of complex assemblies by a design team, such as a parametric family of automotive transmissions.

## **Why Use Knowledge Fusion?**

With the integration of Knowledge Fusion, a single data model can be used to manage both parametric and knowledge based part information. It can also capture rules/requirements from the engineering designer and convey this information seamlessly into design definitions through the instantiation of class files into UG part files. With this capability, UG is raised to the status of a design automation, or as we say, a “Predictive Engineering” product. This is because Knowledge Fusion is a knowledge based language, and can therefore capture both geometry and non–geometric attributes of a given part or assembly, and write the rules which describe the process to create it. This in essence gives us the ability to capture intelligence and engineering know–how within the CAD/CAM part file.

1

## **Who will use Knowledge Fusion?**

Knowledge Fusion would normally be used by three groups of people:

- Designers / End users, using collections of rules defining objects, called “class” files, likely created by software developers. End users would be creating/updating UG part files with instances of class files, using the Knowledge Fusion Navigator or UI Styler interface.
- Design Engineers could also create or edit files using the Knowledge Fusion Navigator interface, and then use instances of these files in UG part files.
- Software developers (KBE experts), who write and debug class files in Knowledge Fusion, creating complex automated systems of various types. Developers would normally be programming by creating/updating what are called “data files, ASCII” (DFA) files. These files can be created in any text editor, and are identified with a .dfa file suffix.

This class is aimed at the first two groups of people and is a prerequisite for the successive class – “Knowledge Fusion for Programmers”, which is aimed at the third group of people. For those of you who want to learn more about the Knowledge Fusion Language (generating the DFA files), you can take the “Knowledge Fusion for Programmers” class.

## ***Basic Concepts of Knowledge Fusion***

1

Knowledge Fusion has certain basic features which, from the developer's standpoint, should be kept in mind in order to maximize the functionality possible. These features are described below. Knowledge Fusion is much like a spreadsheet, in that:

- You can write formulas which can refer to the results of other formulas
- It is not necessary to be a programmer to write formulas
- You can write simple or complex formulas
- The formulas have a syntax.

Solutions to complex problems generally cannot be developed with a simple CAD like point and click interface. They require statements, formulas, relationships and conditions.

### ***Knowledge Fusion is Declarative and Demand Driven***

The Knowledge Fusion language is declarative, rather than procedural, which means that the rules can be written without regard for order. Like a spreadsheet, formulas do not have to be written in any particular order. If another formula is referenced (demanded), the system automatically finds and executes it. This is not the case with procedural languages, like Basic, C, FORTRAN, etc.

## ***Knowledge Fusion is Object Oriented***

Like modern computing languages such as C++, Knowledge Fusion language is object oriented. There are classes and objects (instances), and there is multiple inheritance. A *class* is the abstraction of objects that have the same characteristics. For example, a block is a class of geometries that are commonly in cubic shape, and have length, width and height attributes. An *object* is a specific instance of a class. e.g., when you create a block object, you specify the values of its length, width and height to instantiate a specific instance of the block class. You can create as many block objects (instances) as you want, like block1, block2, block3, etc. They can have different dimensions but they all belong to one class – block. The process of creating object from a class is called *instantiation*.

1

## ***Knowledge Fusion is Hierarchical***

In much the same way engineers think about designing products, in Knowledge Fusion:

- Components are built into sub-assemblies and sub-assemblies are combined to become assemblies.
- There is consistent treatment at all levels.

## ***Geometry***

Knowledge Fusion is like a CAD system, but is not interactive in the same way:

- Some Knowledge Fusion Parts will be raw geometry.
- Geometric parts in the language create geometric entities in the CAD system.
- Knowledge Fusion controls relationships between its parts, and automatically propagates them to the CAD system.
- Knowledge Fusion Supports the geometry of Unigraphics.

## Some Key Definitions

1

Since several Knowledge Fusion–specific terms have been introduced up to this point, some key definitions are described in the following:

### *What is a Knowledge Fusion Class?*

- A class is the logic for creating an object
- A class is a collection of rules
- A class has parameters that receive values
- KF classes are similar to C++ and Java classes
- Classes are instantiated to create KF objects (instances)
- Classes can instantiate other classes forming an object tree

### *What is a Knowledge Fusion Object?*

- Different than a UG object
- An object is an instance of a class
- An object is sometimes called an instance
- An instance has unique values for the class's parameters
- A class can be instantiated multiple times yielding multiple unique instances

*What is a Knowledge Fusion Attribute?*

- Much different than a UG attribute (totally unrelated)
- An attribute is basically a named value
- Closest thing in programming world is a “Variable”
- Can be a floating point number, an integer, a string, a point, a vector, an instance, a list of other attributes, etc.

*What is a Knowledge Fusion Rule?*

- Rules are what describe the object or part that is being designed.
- A rule will state a dimension, a radius, a length, a start angle, end angle, start point, end point, etc.
- A rule might also be a mathematic expression or formula, or, it might address such items as design stresses, resultant volume, position, mating or orientation.
- There are class rules (found in a class file), and dynamic rules, which are rules that have been instantiated into a UG part file.



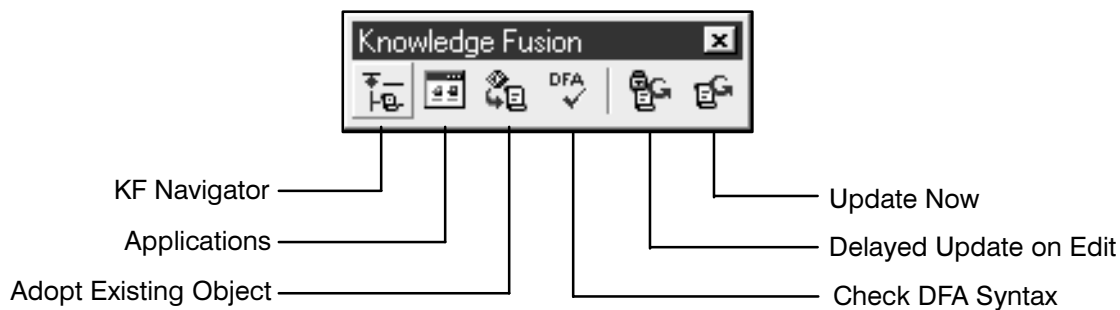
## Knowledge Fusion Toolbar

1

The Knowledge Fusion application is equipped with a toolbar to help you quickly navigate through desired Knowledge Fusion options. You can turn on the toolbars you want to be visible and customize their contents using the View→Toolbars→Customize→Commands or by moving the cursor over the docking container and clicking MB3.

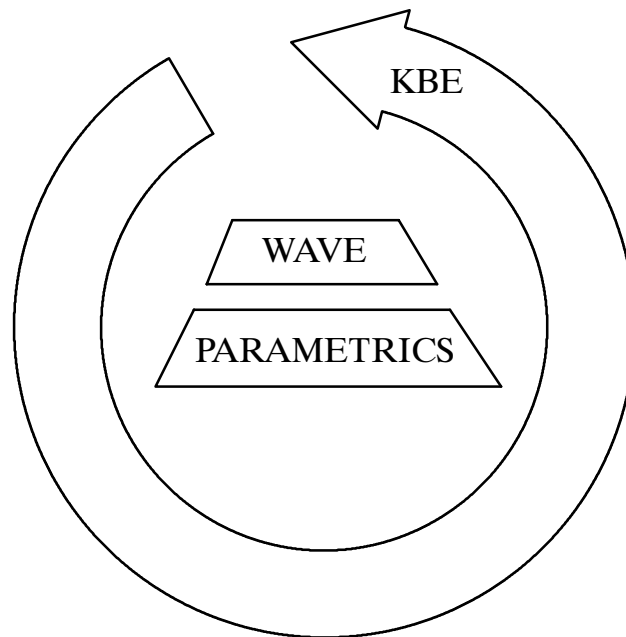
**NOTE**

If you add or remove options from the toolbar using the Customize commands, your toolbar will be different from the one shown here.



## Knowledge Fusion, Parametrics and WAVE

So how does the new product fit in with parametric modeling approaches and WAVE? Well, think of how WAVE has revolutionized parametric modeling by taking the concept of associativity to its next logical step. KBE does that for both parametric modeling and WAVE. With Knowledge Fusion, you will now have the ability to access outside applications such as databases and spreadsheets as well as model things parametrically. You can also use the new product to drive a control structure which then drives the product model. Perhaps a simple diagram may help with this concept.



## Activity 1 – 1: What Do You Think

1

In this activity, you will list some of your impressions and opinions about the Knowledge Fusion product and KBE in general.

**Step 1** Below, list three things you know about KBE.

☐☐☐

**Step 2** Below, list three things you know about Knowledge Fusion.

☐☐☐

**Step 3** List three applications for Knowledge Fusion at your company.

☐☐☐


This concludes the activity.

## Activity 1 – 2: The Bike Frame Project Tour

1

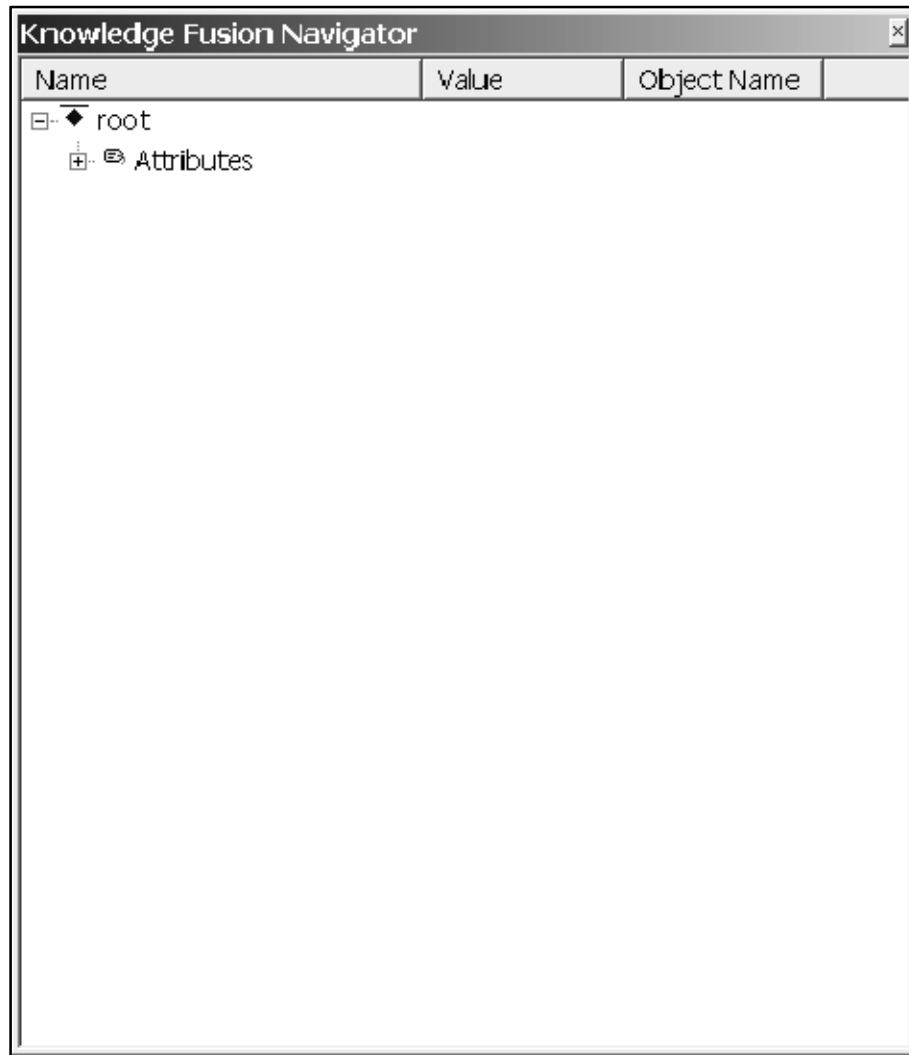
In this activity, you will experiment with a Bike Frame project to get an initial understanding of Knowledge Fusion.

### Step 1 Set up the Knowledge Fusion search directories.

- ☐ Start Unigraphics.
- ☐ Choose **Preferences**→**Knowledge Fusion**.
- ☐ Enter the path where the **bike\_frame** dfa files are located in the New Directory box.  
(e.g. C:\...\<your home directory>\dfa\_files\bike\_frame)
- ☐ Choose .
- ☐ Choose **OK**.

### Step 2 Create a new part and open the Knowledge Fusion Navigator.

- ☐ Choose **File**→**New**. Select **Millimeters** as the Units. Enter **\*\*\*\_fun** in the File name box, where **\*\*\*** are your initials.
- ☐ Choose **OK**.
- ☐ From the graphics window, click MB3. Choose **Replace View**→**TFR – TRI**.
- ☐ Choose **View**→**KnowledgeFusion Navigator**. The Knowledge Fusion Navigator appears. You may resize and/or dock the navigator according to your preferences.



**Step 3 Create an instance of the class bike\_generative.**

- ☐ Choose **Tools**→**Knowledge Fusion**→**Applications**
- ☐ Make sure the **Create** radio button is selected.
- ☐ Select **bike\_generative** from the list.
- ☐ Enter **fun** as the name.
- ☐ Choose **OK**. A UI/Styler dialog appears that provides a GUI interface to the user class.

**Billy Bob's Bike Frames**

Enter Name

Enter Address

Enter City

Enter State

Enter Zip Code

Enter Phone Number

---

Rider Torso Length (millimeters)

300  800

Rider Arm Length (millimeters)

600  1000

Rider Leg Length (millimeters)

600  1000

Rider Weight (kilograms)

30  120

---

Select Material

Racing Frame ☐

Write Contract ☐

Write Database ☐

Frame Weight (grams)

Total Frame Cost

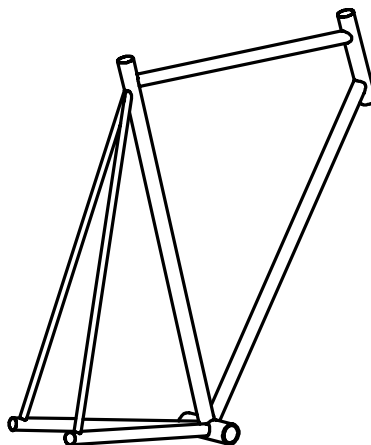
Contract Directory

1

- ☐ In the dialog window, modify the following values:
  - Enter Name: **“Your Name”** (Leave the quotes around your name).
  - You can change the address information if you wish, just be sure to leave the quotes around the text.
  - Drag (or click on) the sliders to adjust the following:
    - **Rider Torso Length: 780**
    - **Rider Arm Length: 840**
    - **Rider Leg Length: 878**
    - **Rider Weight: 82**
  - Select **Steel** as the material
- ☐ Choose **Apply**.

The system will generate a custom bike frame and then automatically switch to Update mode.

- ☐ Fit the view and change the Display Mode to **Shaded**.



The Update dialog will reflect the settings you chose as well as resultant output as shown in the following illustration.

Rider Torso Length (millimeters)

780

300 800

Rider Arm Length (millimeters)

840

600 1000

Rider Leg Length (millimeters)

878

600 1000

Rider Weight (kilograms)

82

30 120

Select Material Steel

Racing Frame ☐

Write Contract ☐

Write Database ☐

Frame Weight (grams) 3013.003

Total Frame Cost 271.0375

Contract Directory "p: \"

OK Apply Cancel

**NOTE**

What you did not see: Inside, the system evaluated the weight and sizes you entered and sized the frame accordingly. The system then calculated the weight of the frame and retrieved cost information from an Microsoft Access database.



**Step 4 Change the Material to Aluminum and recalculate the weight and cost.**

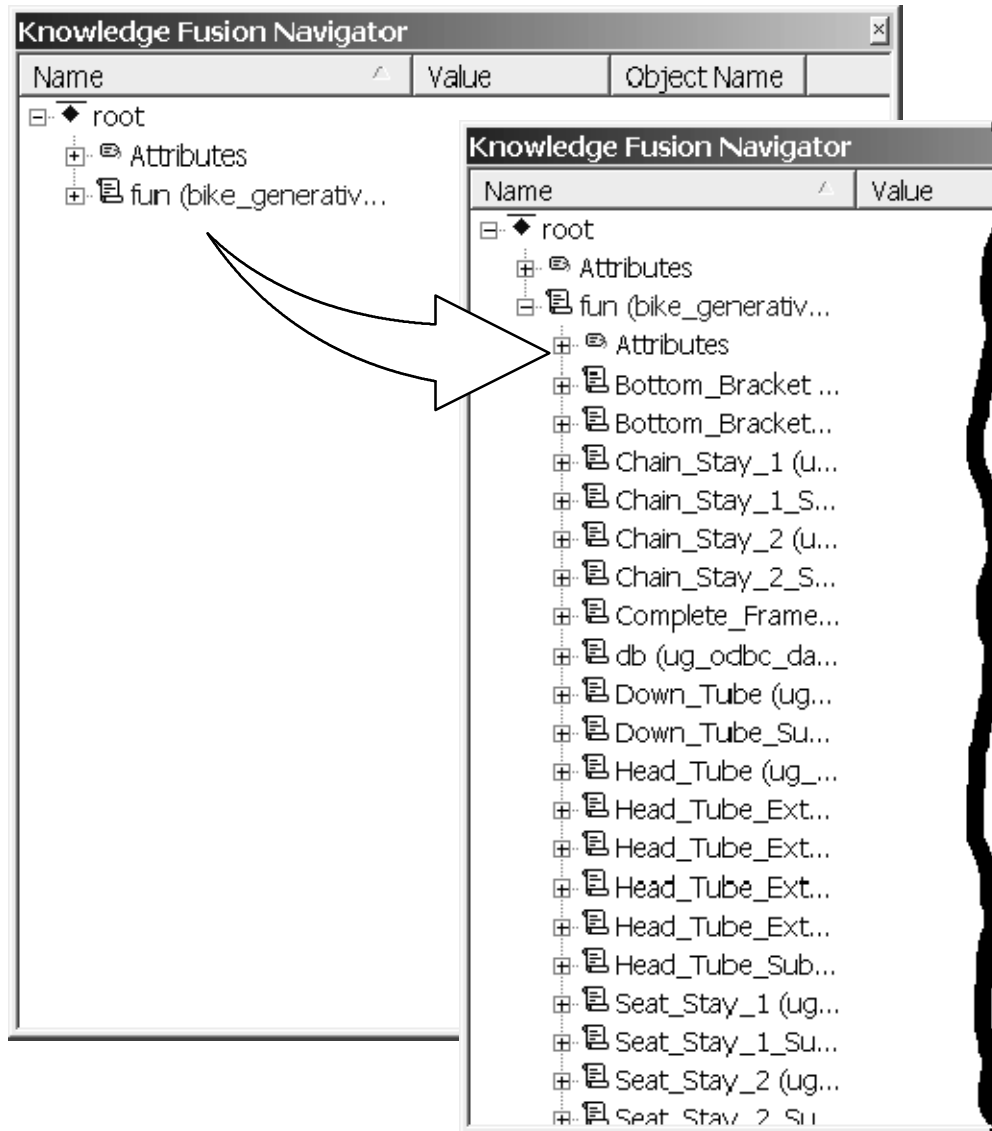
- ☐ Select **Aluminum** as the Material.
- ☐ Choose **Apply**. The system will regenerate the frame and reflect the new weight and cost.

The screenshot shows a software interface with four sliders and several input fields. The sliders are for Rider Torso Length (780 mm), Rider Arm Length (840 mm), Rider Leg Length (878 mm), and Rider Weight (82 kg). Below the sliders is a 'Select Material' dropdown menu set to 'Aluminum'. There are three checkboxes: 'Racing Frame', 'Write Contract', and 'Write Database', all of which are unchecked. To the right of these checkboxes are two text boxes: 'Frame Weight (grams)' with the value '1435.983' and 'Total Frame Cost' with the value '320.6695'. At the bottom, there is a 'Contract Directory' text box containing the path '"p: \'". At the very bottom are three buttons: 'OK', 'Apply', and 'Cancel'.

Parameter	Value
Rider Torso Length (millimeters)	780
Rider Arm Length (millimeters)	840
Rider Leg Length (millimeters)	878
Rider Weight (kilograms)	82
Select Material	Aluminum
Racing Frame	<input type="checkbox"/>
Write Contract	<input type="checkbox"/>
Write Database	<input type="checkbox"/>
Frame Weight (grams)	1435.983
Total Frame Cost	320.6695
Contract Directory	"p: \'

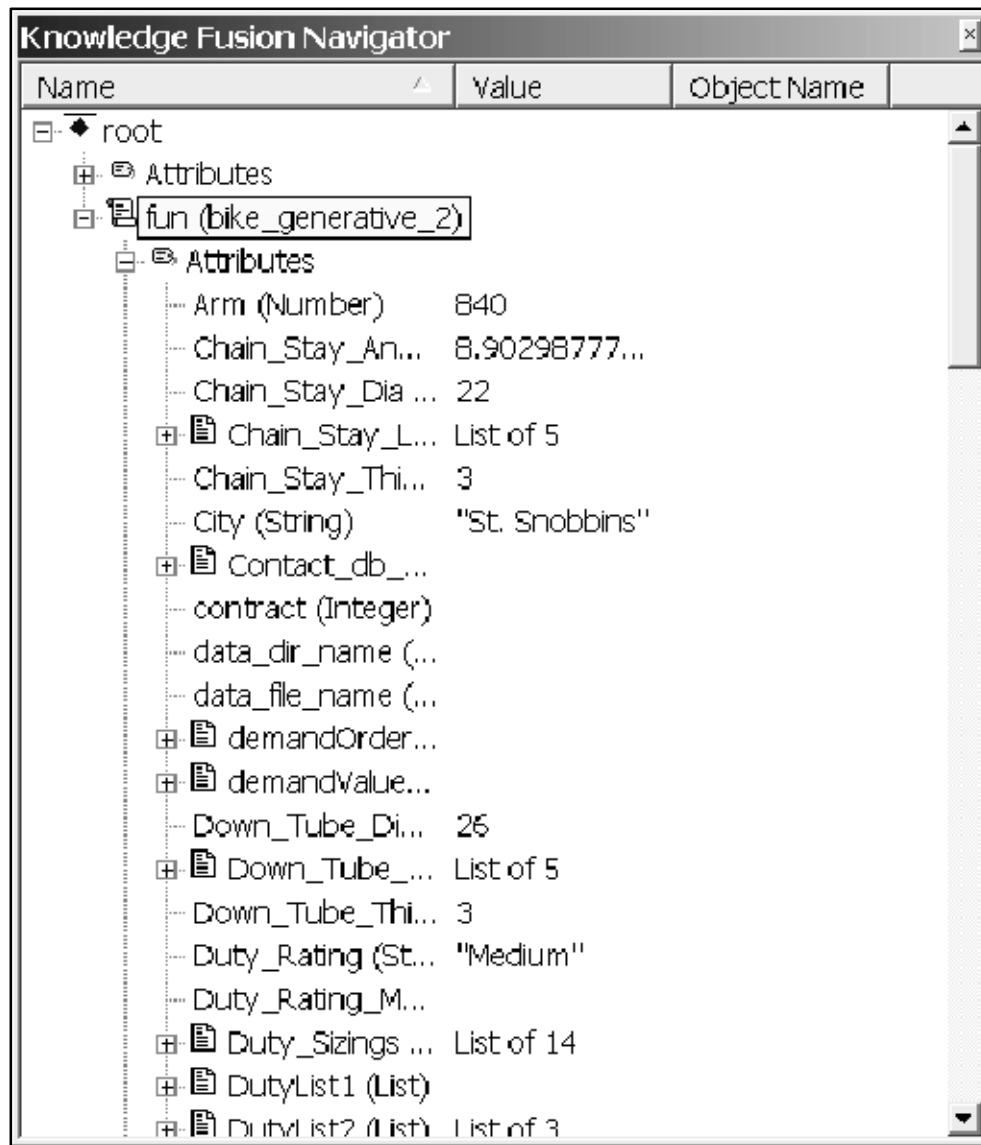
**Step 5 Review the rules.**

- ☐ In the Knowledge Fusion Navigator, select the plus sign next to “fun”.

**1**

- ❑ Next, expand “Attributes” by selecting the plus sign.

1



- ❑ Now that the tree has expanded, let's look at a few things.  
Place your cursor over each of the following rules, hold down on MB3, select **Show Value** and release.



- The first set of attributes is personal information about the rider that were keyed in:
  - **Name**
  - **Street\_Address**
  - **City**
  - **State**
  - **Zip**
  - **Phone**
- The next item is simply an attribute for the frame material.
  - **Material**
- The total frame cost is a function of material type, material welding costs and material length. This sizing information is queried against an Access Database Table and then totalled.
  - **Total\_Frame\_Cost**
- The Write\_Contract attribute is connected to the Write Contract checkbox on the dialog. Since we have not yet requested that a contract file be created, it is FALSE. If the box is checked, it would be set to TRUE and a text file for the contract would be saved. This could then be printed for the customer to sign.
  - **Write\_Contract**
- The Write\_Database attribute is connected to the Write Database checkbox on the dialog. Since we have not yet requested that the customer's information be saved to the database, it is FALSE. If the box is checked, it would be set to TRUE and the customer's personal information would be saved in an Access database.
  - **Write\_Database**

**Step 6 Create the contract and save the customer's information in the database.**

- ☐ Click in the **Write Contract** and the **Write Database** checkboxes in the UI/Styler Update dialog to select them.
- ☐ In the **Contract Directory** box, enter the path where you wish to write the contract text file.  
(e.g. "C:\<your home directory>\documents\**"**)  
The quotes are still needed around the pathname, also, be sure to put a \ at the end of the path, just before the ending quotes.
- ☐ Choose **OK**.

**NOTE**

What you did not see: Inside, the system wrote the rider's information to a database table and created a text file of the contract named "Your Name contract.txt" in the directory you specified.

**Step 7 View the contract.**

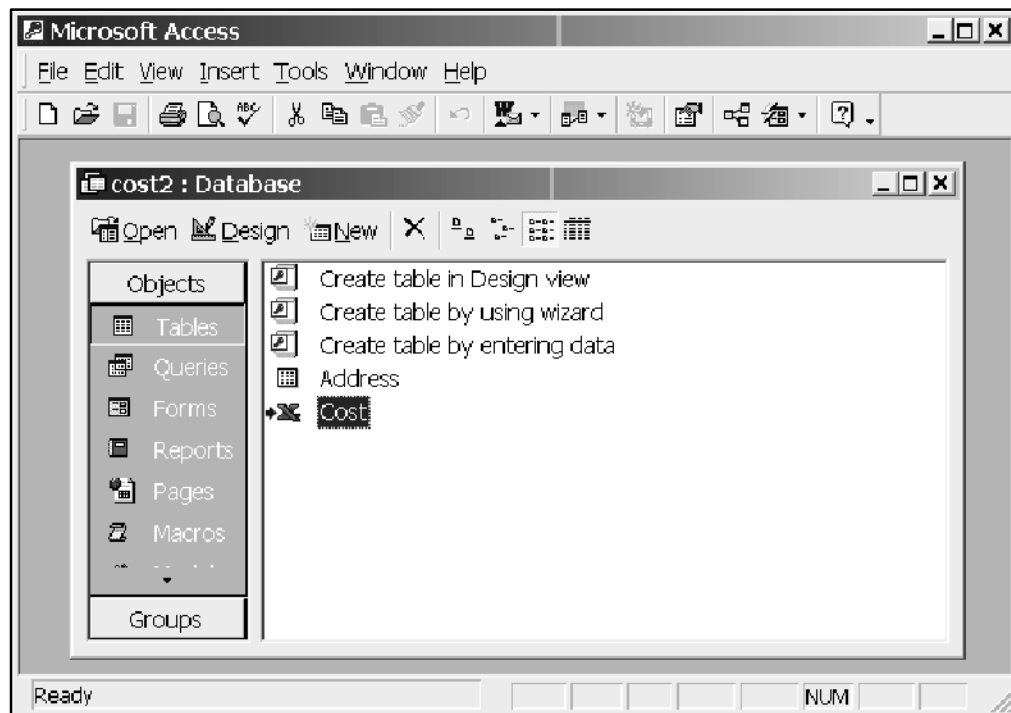
- ☐ Double Click on My Computer.
- ☐ Navigate to the **documents** directory.
- ☐ Click on the file "**Your Name contract.txt**" and then use MB3 to open it with WordPad.

Note the values that have been inserted. The customer's personal information, measurements and other choices are all there including the weight and cost of the frame itself.

- ☐ Close the contract file.

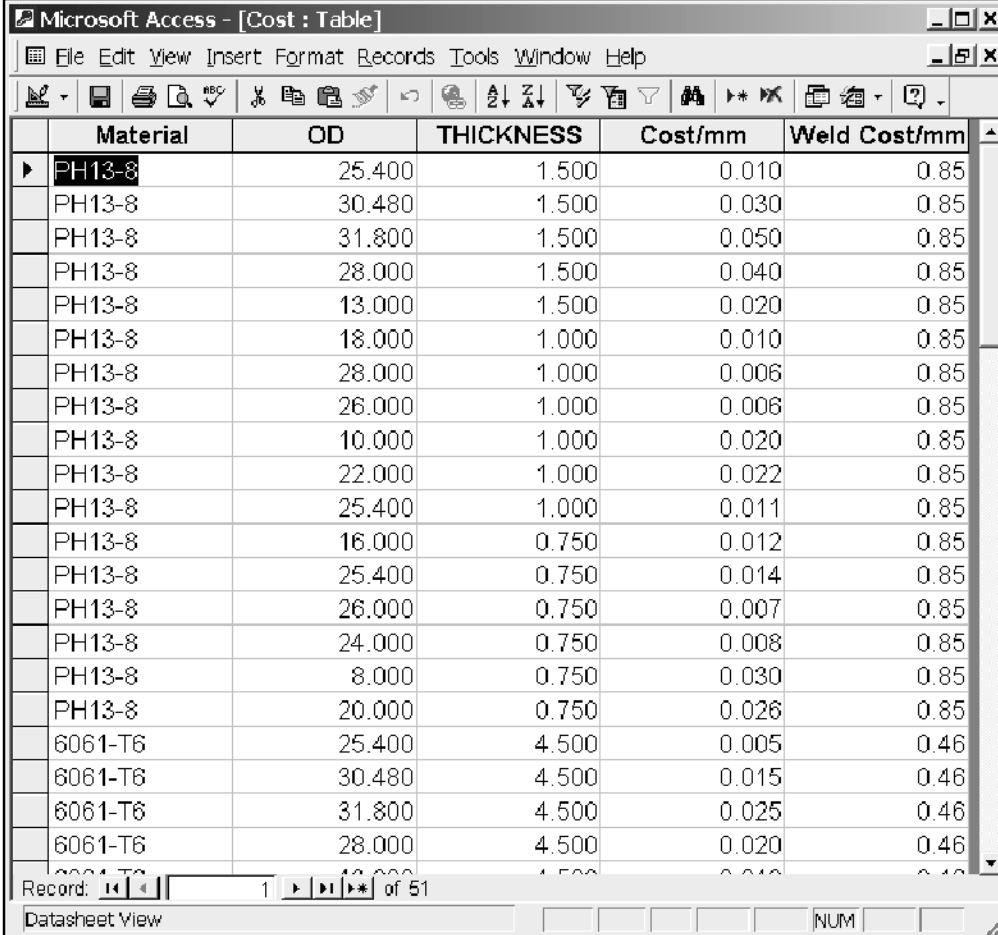
**Step 8 View the tables in the database.**

- ☐ Exit Unigraphics. This is required before we can view the database.
- ☐ In the **databases** directory there will be a file called *cost.mdb*. This is the database that houses both the cost and address tables. Double click the file to open the database.



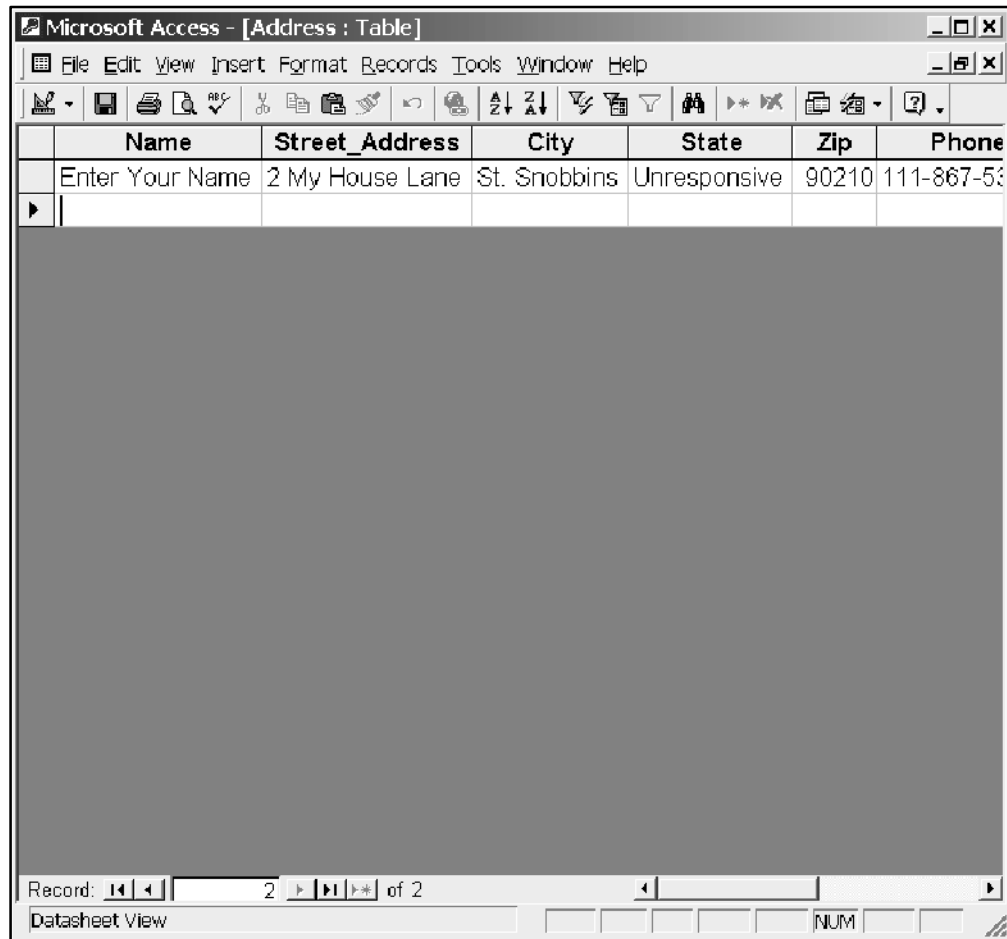
- ☐ Once in the database, select the Cost table and then select open. Notice the layout of the table. This table is little more than a linked Excel Spreadsheet.

1



Material	OD	THICKNESS	Cost/mm	Weld Cost/mm
PH13-8	25.400	1.500	0.010	0.85
PH13-8	30.480	1.500	0.030	0.85
PH13-8	31.800	1.500	0.050	0.85
PH13-8	28.000	1.500	0.040	0.85
PH13-8	13.000	1.500	0.020	0.85
PH13-8	18.000	1.000	0.010	0.85
PH13-8	28.000	1.000	0.006	0.85
PH13-8	26.000	1.000	0.006	0.85
PH13-8	10.000	1.000	0.020	0.85
PH13-8	22.000	1.000	0.022	0.85
PH13-8	25.400	1.000	0.011	0.85
PH13-8	16.000	0.750	0.012	0.85
PH13-8	25.400	0.750	0.014	0.85
PH13-8	26.000	0.750	0.007	0.85
PH13-8	24.000	0.750	0.008	0.85
PH13-8	8.000	0.750	0.030	0.85
PH13-8	20.000	0.750	0.026	0.85
6061-T6	25.400	4.500	0.005	0.46
6061-T6	30.480	4.500	0.015	0.46
6061-T6	31.800	4.500	0.025	0.46
6061-T6	28.000	4.500	0.020	0.46

- ☐ Close the Cost table.
- ☐ Select the Address Table and then select open. This is actually an integral Access table (not a linked spreadsheet).



- ☐ Close the Address Table.
- ☐ Exit the database application.

This concludes the activity.



## SUMMARY

How about that Knowledge Fusion!

In this lesson you:

- Learned some of the various definitions surrounding KBE and Knowledge Fusion.
- Instantiated a Knowledge Fusion design.
- Used the Knowledge Fusion Navigator.



# Knowledge Fusion Navigator

## Lesson 2

### PURPOSE

To learn the major functions of the Knowledge Fusion Navigator.



### OBJECTIVES

Upon completion of this lesson, you will be able to:

- Examine the Knowledge Fusion Navigator
- Examine the Add Attributes dialog
- Examine the Add Child Rule dialog
- Examine the Edit Child Rule dialog
- Instantiate an existing class

This lesson contains the following activities.

Activity	Page
2-1 Knowledge Fusion Navigator Tour . . . . .	2-7
2-2 Add Attributes in the KF Navigator . . . . .	2-18
2-3 Add Child Rule in the KF Navigator . . . . .	2-27
2-4 Instantiating a User Defined Class . . . . .	2-34
2-5 Instantiating a UG System Class . . . . .	2-38

## ***KF Navigator Overview***

The Knowledge Fusion product allows for not only a generative approach to KBE, but for an adoptive approach as well. The combination of these two approaches demands that there be some sort of intermediary to effectively manage them. Furthermore, due to the inherent graphical nature of Unigraphics, an interface is necessary to make the transition simpler.

In this lesson, we will dive into the functionality of the Knowledge Fusion Navigator and the Add and Edit Child Rule dialogs.

As mentioned above, the Knowledge Fusion Navigator serves a few basic purposes. These purposes are different depending on the user involved.

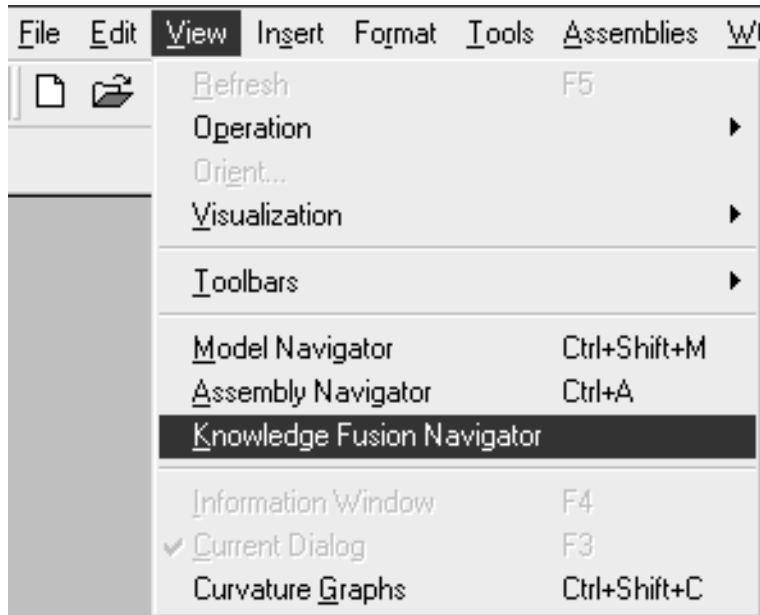
For the End User of a Knowledge Fusion application, the Knowledge Fusion Navigator allows

- graphical instantiation of a Knowledge Fusion design.
- simple modification of rule values.
- graphical representation of instantiated designs.

For the Power User/developer of a Knowledge Fusion application, the Knowledge Fusion Navigator allows the above plus

- an interactive location for rules creation.
- easy referencing ability.
- the ability to easily add rules to adopted objects.
- the ability to reload current designs to account for file changes.

There are two ways to open the Knowledge Fusion Navigator, either use **View→Knowledge Fusion Navigator** or use the Knowledge Fusion Navigator icon on the Knowledge Fusion toolbar.



Knowledge Fusion  
Navigator Control

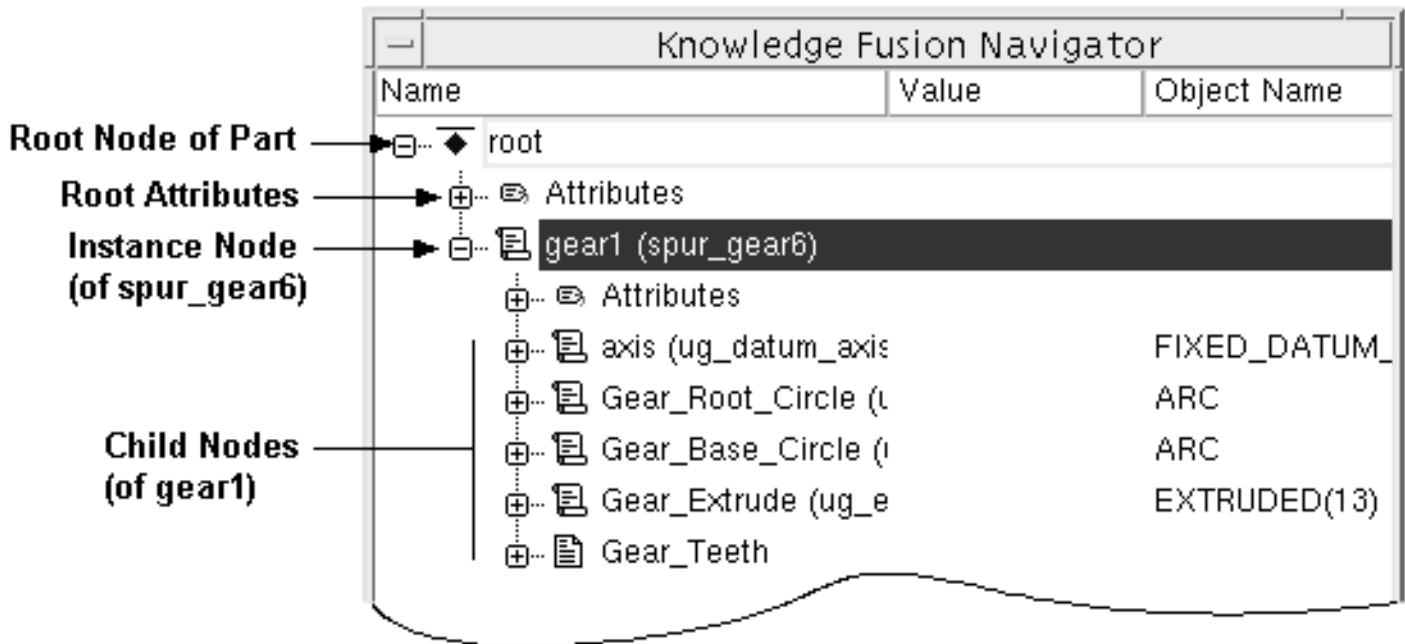


The Knowledge Fusion Navigator dialog displays the object tree which depicts the KF language view for a single instance of a parametric part or “assembly”. Each object node in the tree represents either an instance of a class or an attribute of a single instance of a class.

There are several types of nodes:

- **Root Node** – node at the base of the tree. The root has no parent.
- **Instance Node** – An object in the tree that represents an instance of a class.
- **Child Node** – A node that is contained within another object.
- **Attribute Node** – Each of the Root, Instance, and Child nodes have their own attribute nodes.

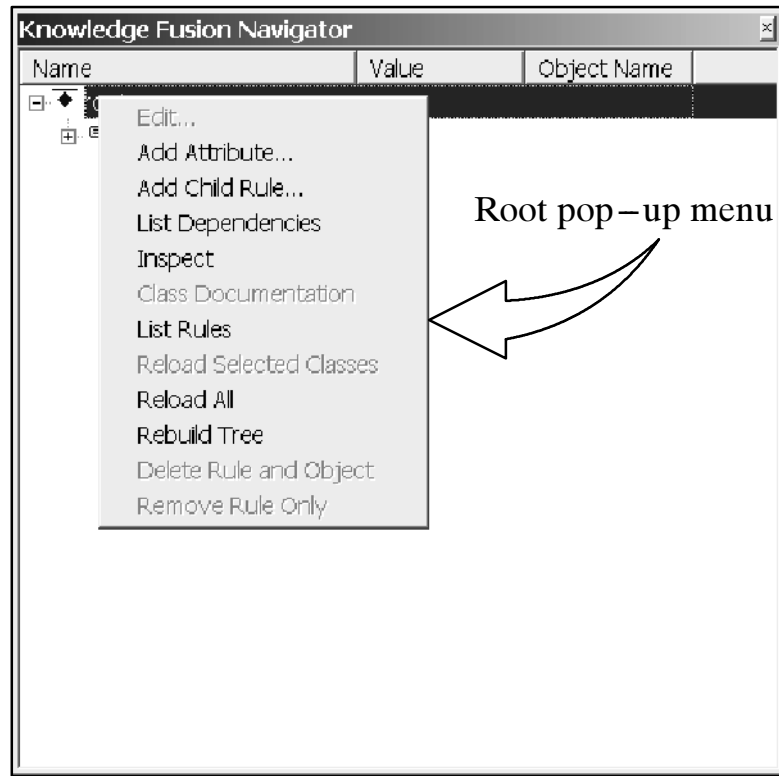
2



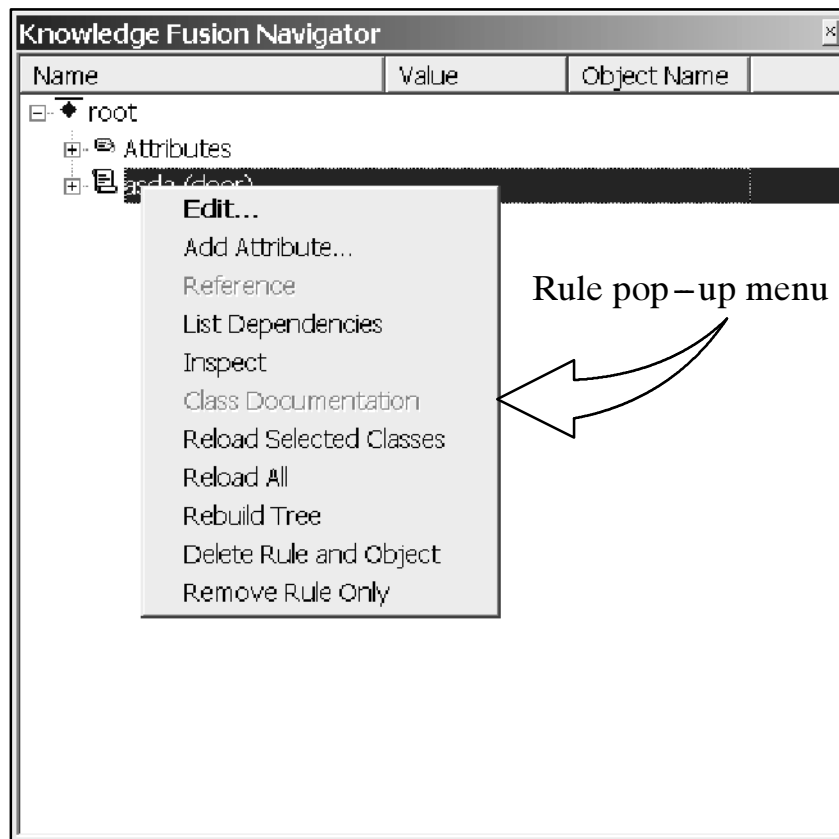
## MB3 Pop-up Menus

There are several different MB3 pop-up menus that can display when you choose nodes in KF Navigator or the KF background.

Pressing MB3 with the cursor over **root** will allow you to do such things as Add Child Rule, Add Attribute, List Rules, Reload All, etc.



Pressing MB3 with the cursor over an already instantiated **rule** will allow you to do such things as Edit, Add Attribute, Inspect Instance, Reload This Classes, Delete Rule and Object, etc.




2

## Activity 2 – 1: Knowledge Fusion Navigator Tour

In this activity, you will familiarize yourself with the Knowledge Fusion Navigator.

### Step 1 Set up the Knowledge Fusion search directories.

- ☐ Start Unigraphics.
- ☐ Choose **Preferences**→**Knowledge Fusion**.
- ☐ Enter the path where the **drive\_coupling** dfa files are located in the New Directory box.  
(e.g. C:\<your home directory>\dfa\_files\drive\_coupling)
- ☐ Choose .
- ☐ Choose **OK**.



### Step 2 Create a new part and open the KF Navigator.

- ☐ Choose **File**→**New**. Enter **\*\*\*\_coupling** in the File name box, where \*\*\* are your initials. Accept **Inches** as the Units. Choose **OK**.
- ☐ From the graphics window, click MB3. Choose **Replace View**→**TFR–TRI**.
- ☐ Choose **View**→**Knowledge Fusion Navigator**.

### Step 3 Add the drive coupling child rule.

- ☐ Place the cursor over **root** and click MB3. Choose **Add Child Rule**.



- ☐ In the Name box, enter **dc**.
- ☐ In the Class list select **drive\_coupling**. The dialog should appear similar to the one shown below.

**Add Child Rule**

Method: Choose Class from List

Name: dc

Class: ☐ User ☐ System ☒ Both

Filter: \*

Class List:

- bolt
- drive\_coupling**
- nut
- ug\_arc
- ug\_block
- un\_hndv

Input Parameters:

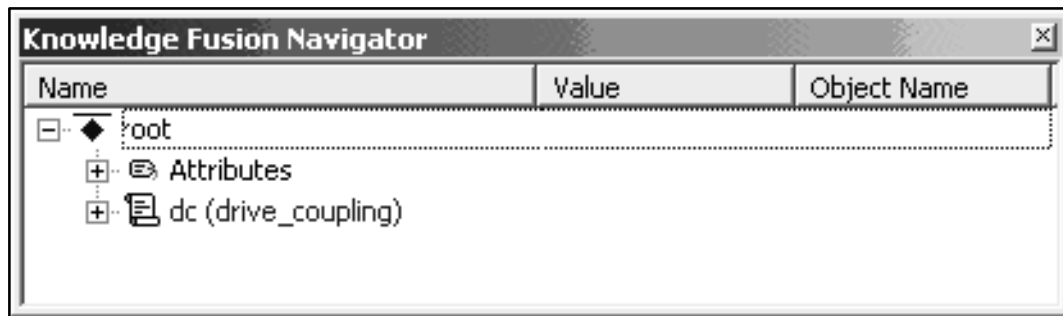
- horsepower : <4.0;>
- rpm : <200;>
- input\_shaft\_radius : <1.0;>
- output\_shaft\_radius : <0.75;>
- shaft\_length : <10;>
- bolt\_diameter : <0.5;>

Rule for Parameter:

OK Apply Cancel

- ☐ Choose **OK**.

The KF Navigator, with unexpanded nodes, should appear similar to the figure shown below.

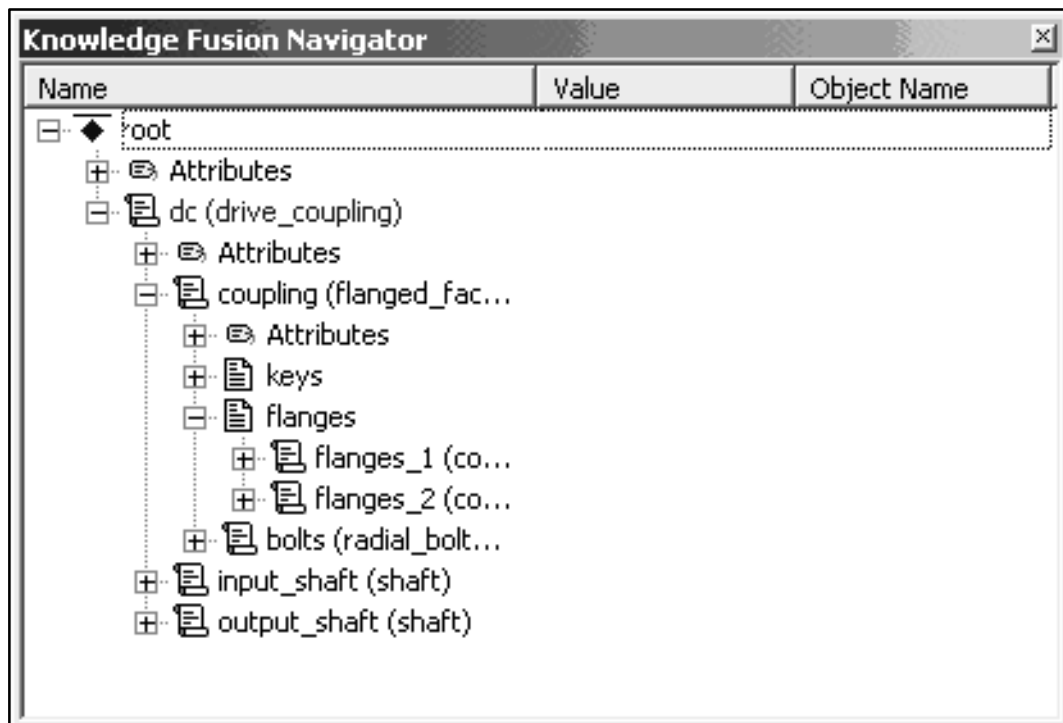


2

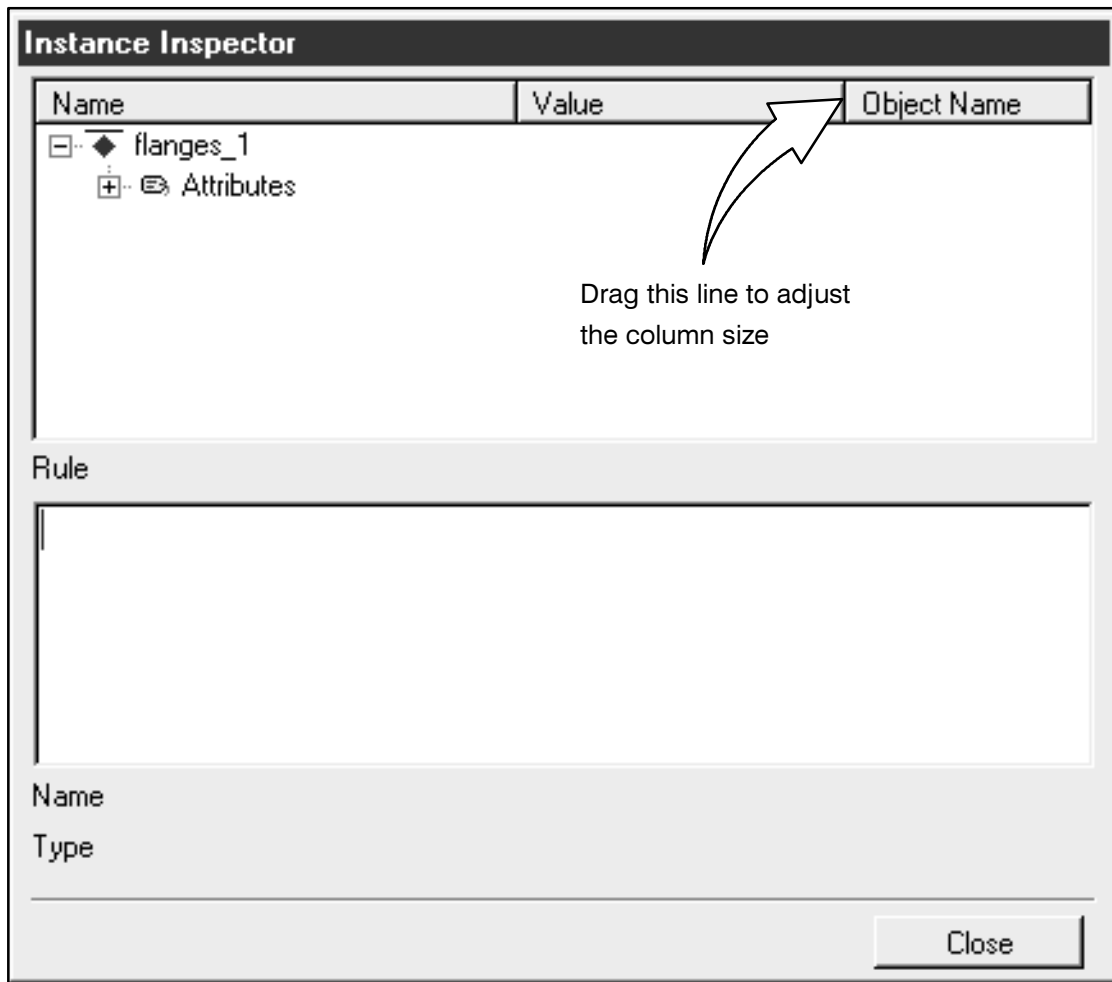
#### Step 4 Inspect a flange cost.

- ☐ In the KF Navigator expand the **dc** by clicking the + sign.
- ☐ Expand the **coupling** by clicking the + sign.
- ☐ Expand the **flanges** by clicking the + sign.

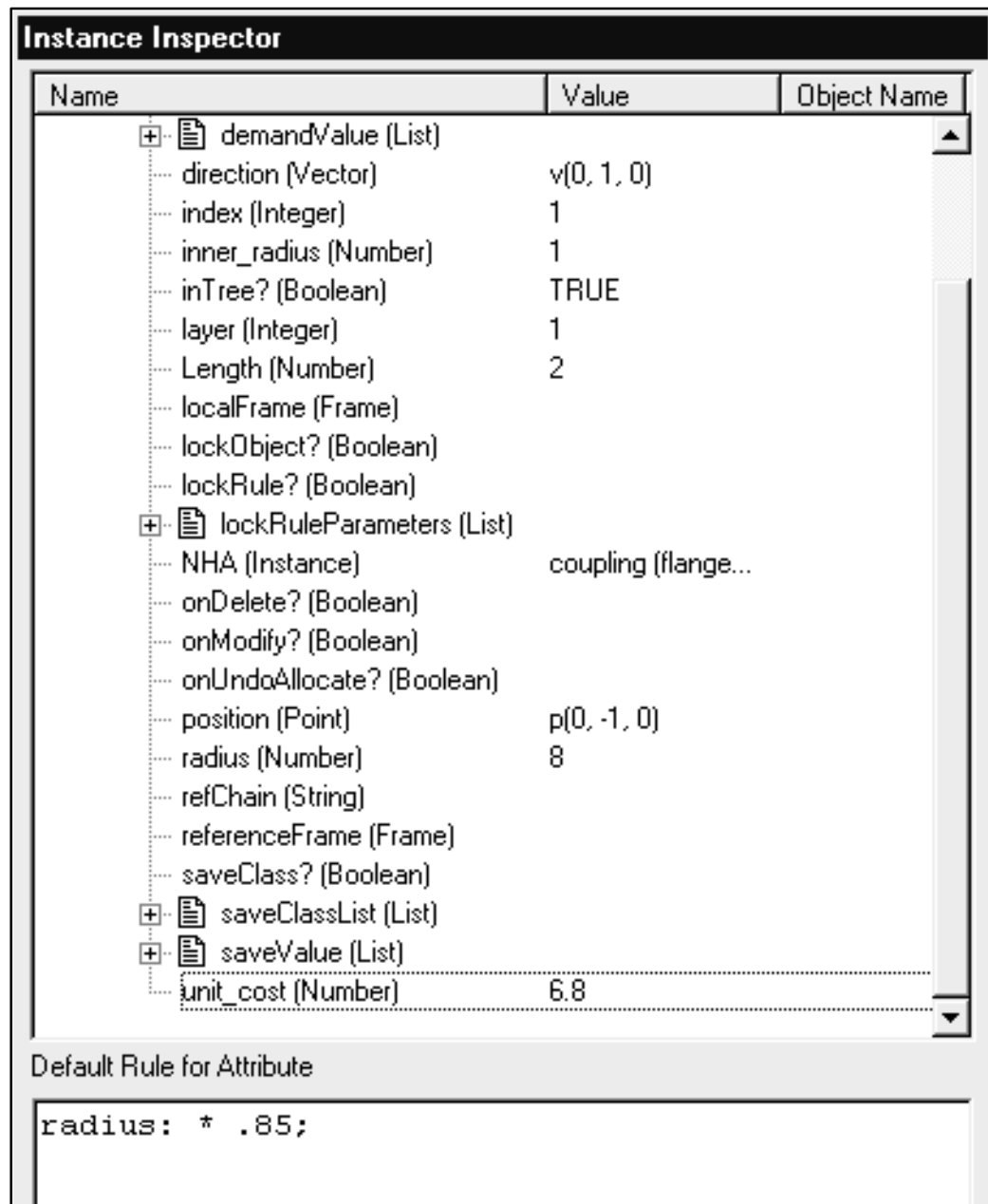
The KF Navigator should appear similar to the figure shown below.



- ☐ Place the cursor over **flanges\_1** and click MB3. Choose **Inspect Instance**. The Instance Inspector dialog displays.
- ☐ Enlarge the dialog and make the columns wider so you can see all the information.



- ☐ In the Instance Inspector, expand the **flanges\_1** Attributes by clicking the + sign.
- ☐ Place the cursor over **unit\_cost** and click MB3. Choose **Show Value**. The Instance Inspector dialog should appear similar to the one shown in the following figure.



- ☐ Close the Instance Inspector dialog.

#### Step 5 Edit the drive coupling child rule.

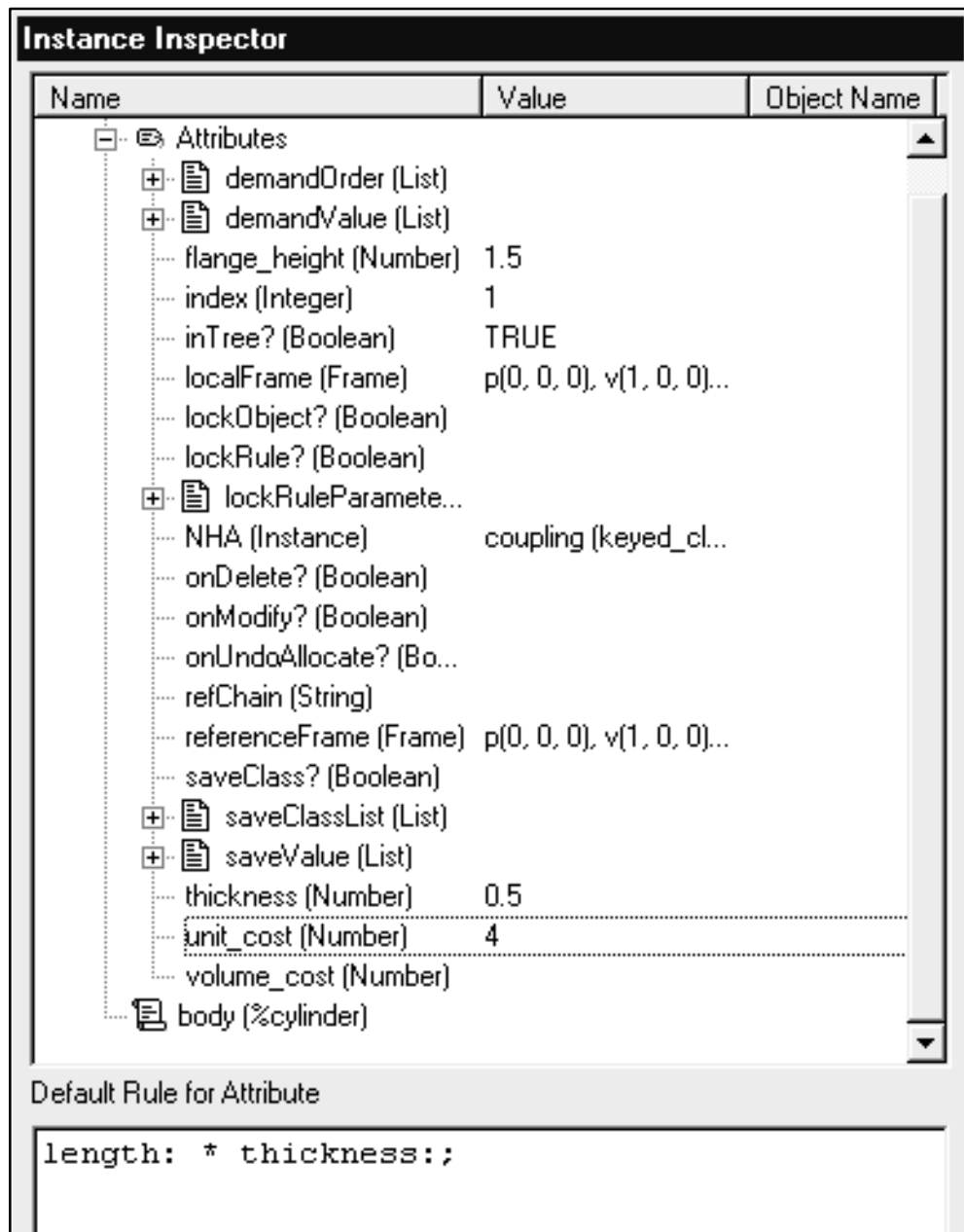
- ☐ In the KF Navigator, place the cursor over **dc** and click MB3. Choose **Edit**.
- ☐ From the Input Parameters box, select **output\_shaft\_radius:<0.75;>**.

- ☐ In the Rule for Parameter box, remove **0.75** and enter **1.0**.
- ☐ Choose **OK**.

**Step 6 Inspect a split\_half cost.**

- ☐ Expand the **split\_half** by clicking the + sign.
- ☐ Place the cursor over **split\_half\_1** and click MB3. Choose **Inspect Instance**. The Instance Inspector dialog displays.
- ☐ Enlarge the dialog and make the columns wider so you can see all the information.
- ☐ In the Instance Inspector, expand the **split\_half\_1** Attributes by clicking the + sign.
- ☐ Place the cursor over **unit\_cost** and click MB3. Choose **Show Value**. The Instance Inspector dialog should appear similar to the one shown in the following illustration.





- ☐ Close Instance Inspector dialog.

### Step 7 Edit the drive coupling child rule.

- ☐ In the KF Navigator, place the cursor over **dc** and click MB3. Choose **Edit**.

- ☐ From the Input Parameters box, select **horsepower: <4.0;>**.
- ☐ In the Rule for Parameter box, remove **4.0** and enter **0.5**.
- ☐ Choose **OK**.
- ☐ Save and close the part.

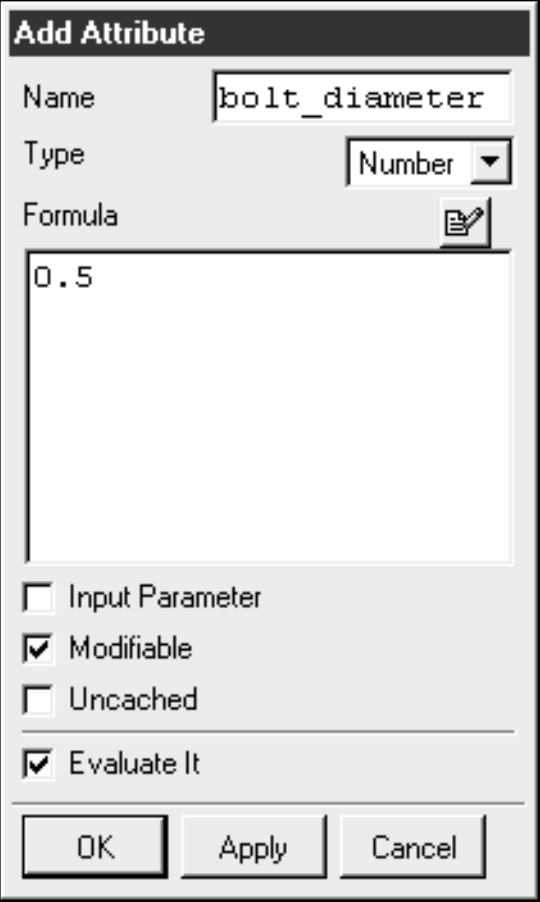
This concludes the activity.



2

## Add Attribute

The MB3 Add Attribute dialog lets you add a Knowledge Fusion attribute to any child rule in the object tree of the KF Navigator. A KF attribute is not the same as a UG attribute. A KF attribute must have a name (user defined) and has a value of a designated type. In a programming environment a KF attribute is similar to a variable.



The screenshot shows the 'Add Attribute' dialog box. The 'Name' field contains 'bolt\_diameter'. The 'Type' dropdown menu is set to 'Number'. The 'Formula' text area contains '0.5'. Below the formula field are four checkboxes: 'Input Parameter' (unchecked), 'Modifiable' (checked), 'Uncached' (unchecked), and 'Evaluate It' (checked). At the bottom of the dialog are three buttons: 'OK', 'Apply', and 'Cancel'.



### Basic Procedure to Add an Attribute

1. Enter a name
2. Choose attribute type
3. Enter a formula
4. Choose behavior options
5. Choose Apply or OK



### *Attribute Data Types*

An attribute's value has a range which is specified by its type. Attributes can have any one of the following types:

**Boolean** – is either TRUE or FALSE.

**Frame** – defines a local KF coordinate system.

**Integer** – can be a positive whole number, or a negative whole number, or zero.

**List** – is an ordered set of other types.

**Name** – is a symbolic constant. An unquoted sequence of alphanumeric characters, the underscore (\_), and the question mark (?). It can be a symbolic constant used to refer to the name of an attribute.

**Number** – is a number that can have a decimal value. Number is a double precision, floating point number. Number is the default type.

**Point** – is a point object.

**String** – is a double-quoted sequence of characters.

**Vector** – is a vector object.

**Instance** – is an instance of a class.

**User** – is a 32-bit value. A value that is not interpreted by the language system.

**Any** – specifies that an attribute can hold any KF data type.

*Attribute Behavior Options*


- **Input Parameter**      Attribute is an input to a class and might receive a value when the class is instantiated.
- **Modifiable**          Attribute can be modified external to Knowledge Fusion.
- **Uncached**            Attribute's value is not saved in memory. The attribute recalculates its value every time it is referenced by another attribute.
- **Evaluate It**          When toggled ON, evaluates the attribute after you create it.



## Activity 2–2: Add Attributes in the KF Navigator

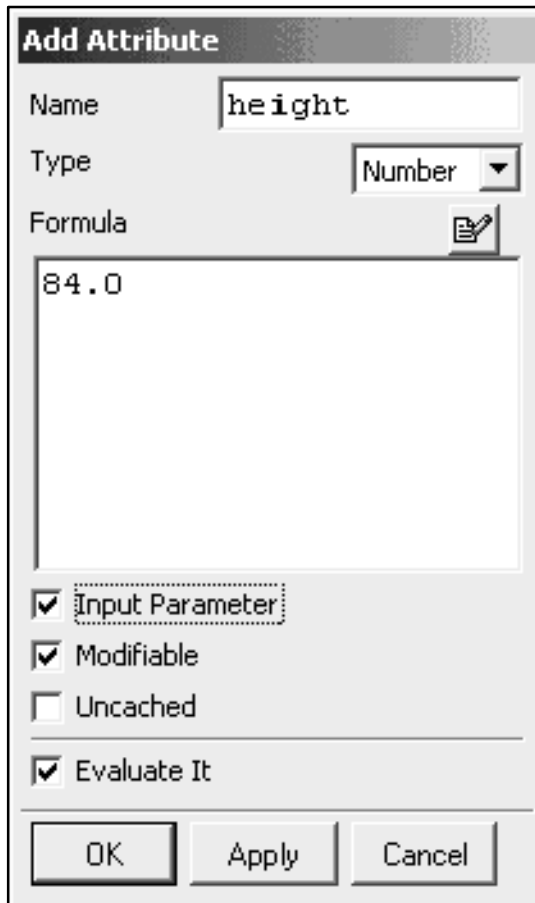
In this activity, you will create four attributes: height, width, thickness, and position in the KF Navigator. These attributes are used in the next activity.

### Step 1 Create a new part and open the KF Navigator.

- 
- ☐ Choose **File**→**New**. Enter **\*\*\*\_door** in the **File name** box, where **\*\*\*** represents your initials. Accept **Inches** as the Units. Choose **OK**.
  - ☐ Click **MB3** from the graphics window. Choose **Replace View**→**TFR–TRI**.
  - ☐ Choose **View**→**Knowledge Fusion Navigator**. Click the **+** sign on **Attributes** under the root node.

### Step 2 Add the height attribute.

- ☐ Place the cursor over **root** and click **MB3**. Choose **Add Attribute**. The Add Attribute dialog displays.
- ☐ Enter **height** in the Name box.
- ☐ Choose **Number** from the Type list.
- ☐ Enter **84.0** in the Formula box.
- ☐ Select the **Input Parameter** check box (it should be checked). Verify that **Modifiable** is checked and that **Uncached** is unchecked. **Evaluate It** should be checked.



**Add Attribute**

Name:

Type:

Formula:

☒ Input Parameter

☒ Modifiable

☐ Uncached

☒ Evaluate It

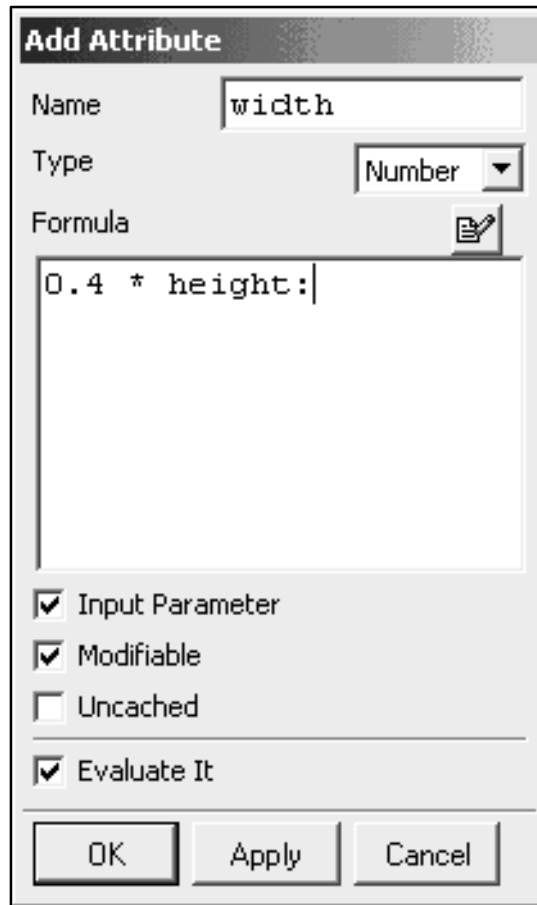

2

- ☐ Choose **Apply**.

### Step 3 Add the width attribute.

- ☐ Enter **width** in the Name box. The Type list should read **Number**.
- ☐ In the Formula box, enter **0.4 \* .** In the KF Navigator, place the cursor over the **height** attribute. Click MB3. Choose **Reference**.


Note that **height:** appears in the Formula box. The colon after height references the value of the height attribute.



**Add Attribute**

Name

Type

Formula 

`0.4 * height:`

☒ Input Parameter

☒ Modifiable

☐ Uncached

☒ Evaluate It

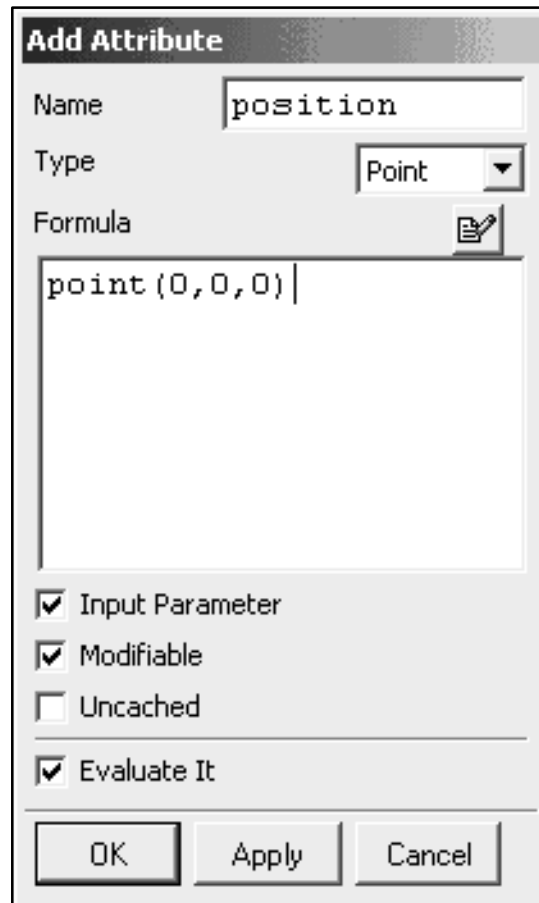
- ☐ Choose **Apply**.

**Step 4 Add the thickness attribute.**

- ☐ Enter **thickness** in the Name box.
- ☐ Enter **2.5** in the Formula box.
- ☐ Choose **Apply**.

**Step 5 Add the position attribute.**

- ☐ Enter **position** in the Name box.
- ☐ Choose **Point** from the Type list.
- ☐ Enter **point(0,0,0)** in the Formula box.



**Add Attribute**

Name: position

Type: Point

Formula: point(0,0,0)

☒ Input Parameter  
☒ Modifiable  
☐ Uncached  
☒ Evaluate It

OK Apply Cancel

**2**

- ☐ Choose **OK**.

The KF Navigator should appear similar to the one shown in the following figure. Note the new attributes in blue color.

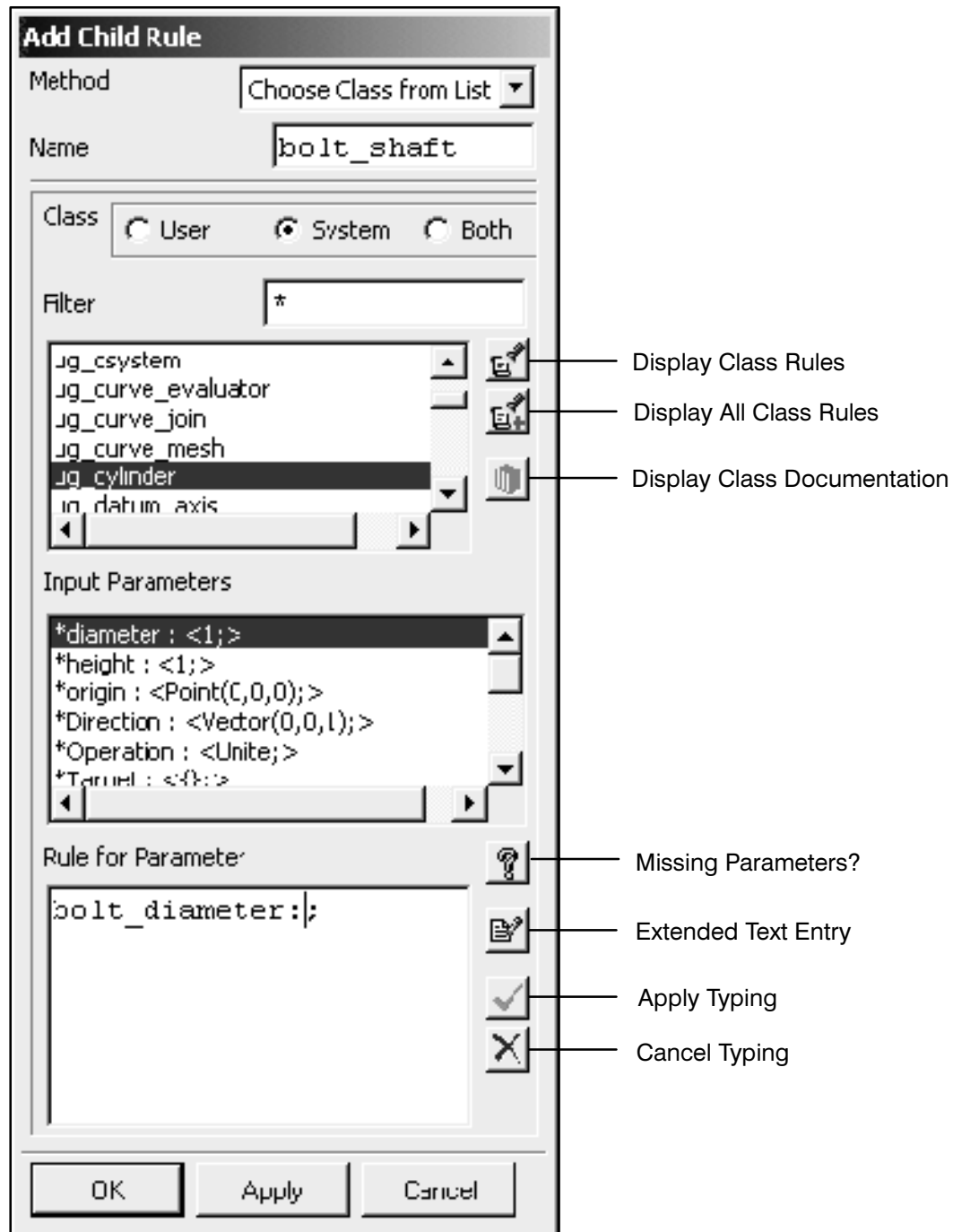
Knowledge Fusion Navigator		
Name	Value	Object Name
[-] ◆ root		
[-] Attributes		
[-] demandOrder (List)		
[-] demandValue (List)		
height (Number)	84	
index (Integer)	0	
inTree? (Boolean)		
localFrame (Frame)	p(0, 0, 0), v(1, ...	
NHA (Instance)	World (%%World)	
onDelete? (Boolean)		
onModify? (Boolean)		
onUndoAllocate? (Boolean)		
position (Point)	p(0, 0, 0)	
refChain (String)		
referenceFrame (Frame)	p(0, 0, 0), v(1, ...	
saveClass? (Boolean)		
[-] saveClassList (List)		
[-] saveValue (List)		
thickness (Number)	2.5	
width (Number)	33.6	

### Step 6 Save the part.

This concludes the activity.

## Add Child Rule

To add a child rule in the Knowledge Fusion Navigator, place the cursor over **root**, press **MB3** and select **Add Child Rule**. This will activate the Add Child Rule dialog.



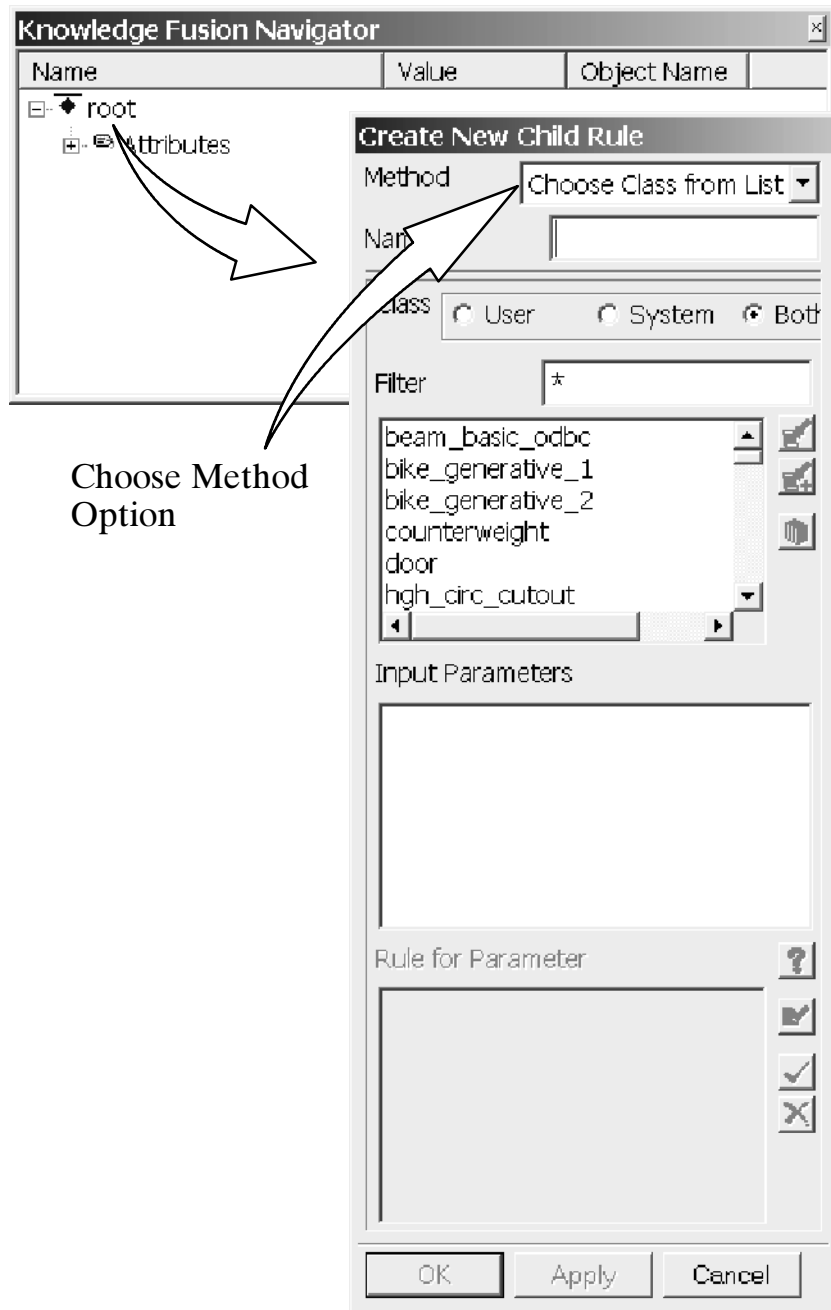


Now, we will look at a process for instantiating an existing class using Add Child Rule dialog.

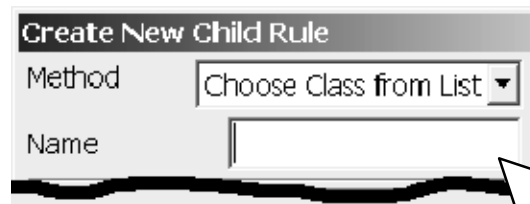
- First, choose Method option, the default one is **Choose Class from List**.

Press MB3 while over root and choose Add Child Rule

Choose Method Option



- Next, enter a name for the instantiated class in the **Name** text field.



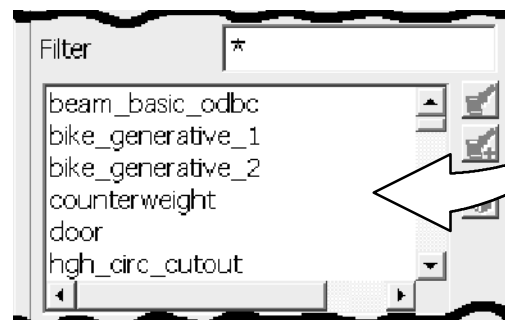
Enter rule  
Name here

- Next, select **User**, **System** or **Both**.



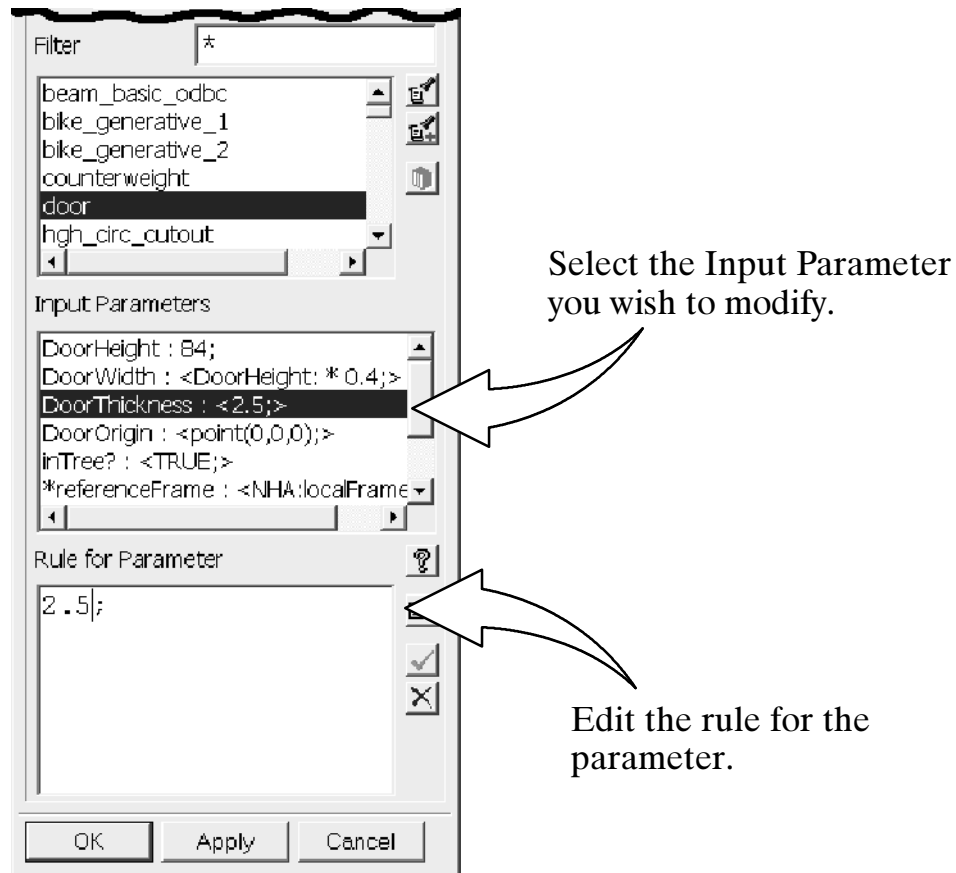
Select the Class type

- Next, select the class that you wish to instantiate in your session.



Select the Class

- Finally, select and modify the Input Parameters you wish to change and select **Apply** or **OK**.



## Activity 2–3: Add Child Rule in the KF Navigator

In this activity, you will add child rules for a door, a window, and a knob. This activity is a continuation from Add Attributes in the KF Navigator and uses the attributes from that activity.

### Step 1 Add the door child rule.

- ☐ If \*\*\*\_door.prt is not open, retrieve this part and open the Knowledge Fusion Navigator. Click the + sign on Attributes under the root node.
- ☐ Place the cursor over **root** and click MB3. Choose **Add Child Rule**.
- ☐ In the Name box, enter **door**.
- ☐ Make sure either **System** or **Both** button is checked.
- ☐ In the Class list select **ug\_block**.
- ☐ From the Input Parameters box, select **\*length:<1;>**.
- ☐ In the Rule for Parameter box, backspace and remove the **1**. Only the semicolon should remain.
- ☐ In the KF Navigator, place the cursor over the **thickness** attribute. Click MB3. Choose **Reference**. Note that **thickness:** followed by a semicolon appears in the Rule for Parameter box. The colon after thickness references the value of the thickness attribute.

The dialog should appear similar to the one shown below.



2

**Add Child Rule**

Method: Choose Class from List

Name: door

Class: ☐ User ☒ System ☐ Both

Filter: \*

Class List:

- ug\_arc
- ug\_block
- ug\_body
- ug\_boolean
- ug\_child\_in\_part
- un\_component

Input Parameters:

- \*Length : <1;>
- \*width : <1;>
- \*height : <1;>
- \*origin : <Point(0,0,0);>
- \*X\_Axis : <Vector(1,0,0);>
- \*Y\_Axis : <Vector(0,1,0);>

Rule for Parameter:

thickness:;

Buttons: OK, Apply, Cancel

- From the Input Parameters box, select **\*width:<1;>**.  
Backspace over the 1 so that only the semicolon remains.

- ☐ In the KF Navigator, place the cursor over the **width** attribute. Click MB3 and choose **Reference**. Note that **width:** followed by a semicolon displays in the Rule for Parameter box.
- ☐ From the Input Parameters box, select **\*height:<1;>**. Backspace over the **1** so that only the semicolon remains.
- ☐ In the KF Navigator, place the cursor over the **height** attribute. Click MB3 and choose **Reference**. Note that **height:** followed by a semicolon displays in the Rule for Parameter box.
- ☐ From the Input Parameters box, select **\*origin:<Point(0,0,0);>**. Backspace until only the semicolon remains.
- ☐ In the KF Navigator, place the cursor over the **position** attribute. Click MB3 and choose **Reference**. Note that **position:** followed by a semicolon displays in the Rule for Parameter box.
- ☐ Choose **Apply**.



## Step 2 Add the window child rule.

- ☐ In the Name Box, enter **window**.
- ☐ From the Class List box, select **ug\_block**.
- ☐ From the Input Parameters box, select **\*length:<1;>**.
- ☐ Backspace over the **1** so that only the semicolon remains.
- ☐ In the KF Navigator, place the cursor over the **thickness** attribute. Click MB3. Choose **Reference**.
- ☐ From the Input Parameters box, select **\*width:<1;>**.
- ☐ Backspace over the **1** so that only the semicolon remains.

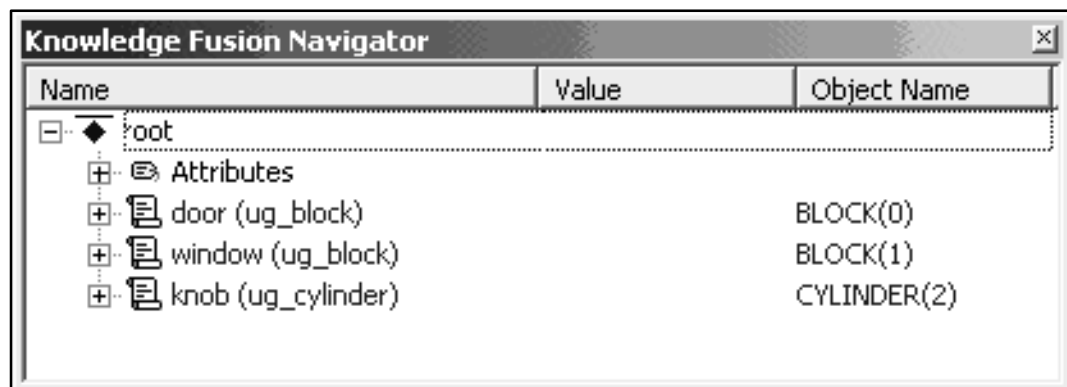
- ☐ In the KF Navigator, place the cursor over the **width** attribute. Click MB3 and choose **Reference**.
- ☐ Type **/4** after **width**:. The following should display in the Rule for Parameter box: **width:/4**;. This rule takes 1/4 the value of the width attribute.
- ☐ From the Input Parameters box, select **\*height:<1>**;
- ☐ Backspace over the **1** so that only the semicolon remains.
- ☐ In the KF Navigator, place the cursor over the **height** attribute. Click MB3 and choose **Reference**.
- ☐ Type **/4** after **height**:. The following should display in the Rule for Parameter box: **height:/4**;. This rule takes 1/4 the value of the height attribute.
- ☐ From the Input Parameters box, select **\*origin:<Point(0,0,0)>**;
- ☐ In the Rule for Parameter box, backspace until only the semicolon remains.
- ☐ Type the following:  
**position:+vector(0, 3\*width:/8, 5\*height:/8)**. You should see the text you just entered followed by a semicolon. This is a slightly more complex rule that positions the window on the door using vector positioning.
- ☐ From the Input Parameters box, select **\*target:<{}>**;
- ☐ Place the cursor inside the braces {}. Type **door**;
- ☐ From the Input Parameters box, select **\*operation:<Unite>**;
- ☐ Replace **Unite** with **Subtract**. This performs the boolean operation of subtracting the window from the door.
- ☐ Choose **Apply**.

**Step 3 Add the knob child rule.**

- ☐ In the Name box, enter **knob**.
- ☐ From the Class list box, scroll down until you see **ug\_cylinder** and then select it.
- ☐ From the Input Parameters box, select **\*diameter:<1;>**.
- ☐ In the Rule for Parameter box, remove **1** and enter **2.0**.
- ☐ From the Input Parameters box, select **\*height:<1;>**.
- ☐ In the Rule for Parameter box, remove **1** and enter **3.0**.
- ☐ From the Input Parameters box, select **\*direction:<Vector(0,0,1);>**.
- ☐ Edit the value to be **Vector(1,0,0)**.
- ☐ From the Input Parameters box, select **\*origin:<Point(0,0,0);>**.
- ☐ Edit the value to be **position: +vector(thickness:, width: -4, height:/2)**. Check that you still have a single semicolon at the end of this value.
- ☐ Choose **OK**.
- ☐ Save the part.

**2**

The KF Navigator, with unexpanded nodes, should appear similar to the figure shown below.





Your part should look like the following figure.




#### Step 4 Test the design.

- ☐ In the KF Navigator expand the **root** attributes by clicking the **+** sign.
- ☐ Place the cursor over the **width** attribute, click MB3 and choose **Edit**. The Edit Attribute dialog displays.
- ☐ Edit **0.4** to be **0.6**. Choose **Apply**. Notice that the door gets wider and so does the window. But the window retains its relative position.
- ☐ Close the part.

This concludes the activity.

## KF Classes

A KF class is a collection of KF rules that achieves a specific task. The class can be instantiated using the Add Child Rule dialog. To view the contents of the class, you can select the class from the list in the dialog and then click the

Display Class Rules button . There are two types of classes available – User Defined Class and UG System Class.

### *User Defined Class*

You are able to extend UG by creating your own classes using the KF language. The user defined class needs to be saved in a .dfa file and a proper Knowledge Fusion search path needs to be set up within UG so that UG can find and recognize it. In some situations, it may be appropriate that the entire geometric description of a part be encapsulated in a set of classes as an application. When the application class is instantiated in UG, the execution of the rules result in modeling features that are fully associative to its generating class. The features know that they are rule driven and the rules know the features they have created.


### *UG System Class*

UG supplies a rich class/function library that the user can utilize, extend and inherit from, as desired. For example, UG offers classes that can create a line, an edge blend, a solid body, or even an assembly. Classes are not limited to the creation of modeling features; they can make use of the KF system supplied functions and the customer's own knowledge bases to create highly automated proprietary processes. The optimization and database classes are powerful built-in functions and will be introduced in later lessons.

## Activity 2 – 4: Instantiating a User Defined Class

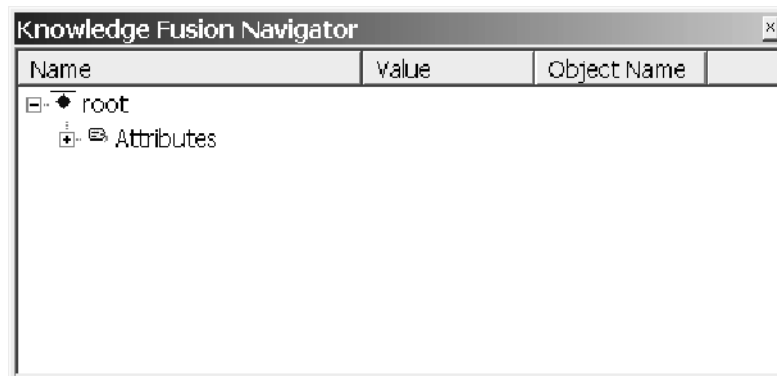
In this activity, you will instantiate an existing user class *mug* which is defined in the *mug.dfa* file.

### Step 1 Set up the Knowledge Fusion search directories.

- ☐ Choose **Preferences→Knowledge Fusion**.
- ☐ Enter the path where the **mug.dfa** file is located in the New Directory box (e.g. C:\<your home directory>\dfa\_files).
- ☐ Choose .
- ☐ Choose **OK**.

### Step 2 Create a new part and open the KF Navigator.

- ☐ Choose **File→New**. Enter **\*\*\*\_mug** in the File name box, where **\*\*\*** are your initials. Accept **Inches** as the Units. Choose **OK**.
- ☐ From the graphics window, click MB3. Choose **Replace View→TFR–TRI**.
- ☐ Choose **View→Knowledge Fusion Navigator** to turn on the KF Navigator.

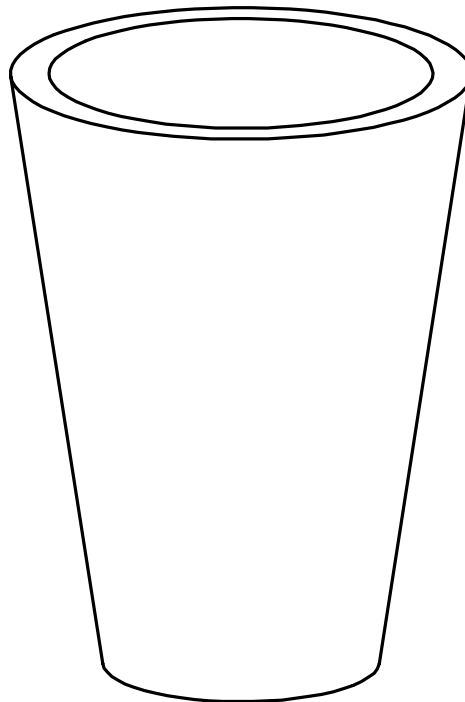


**Step 3 Create an instance of the class mug.**


- ☐ Place the cursor over **root** in the Knowledge Fusion Navigator, click **MB3**, and choose **Add Child Rule**. The Add Child Rule dialog appears.
- ☐ In the Method box, leave the default **Choose Class from List**.
- ☐ Enter **my\_mug** in the Name box.
- ☐ Select the **User** radio button in the Class selection.
- ☐ Select **mug** from the class list.
- ☐ Choose **Apply**.
- ☐ Fit the view.



Notice that a rather simple cup was created. This is due to a boolean rule within the file. We will take a look at the mug class rules at the next step.



**Step 4 View the mug class rules.**

- ☐ Once again, select **mug** from the class list in the Add Child Rule dialog.
- ☐ Click the Display Class Rules button . An Information window contains the class definition appears.

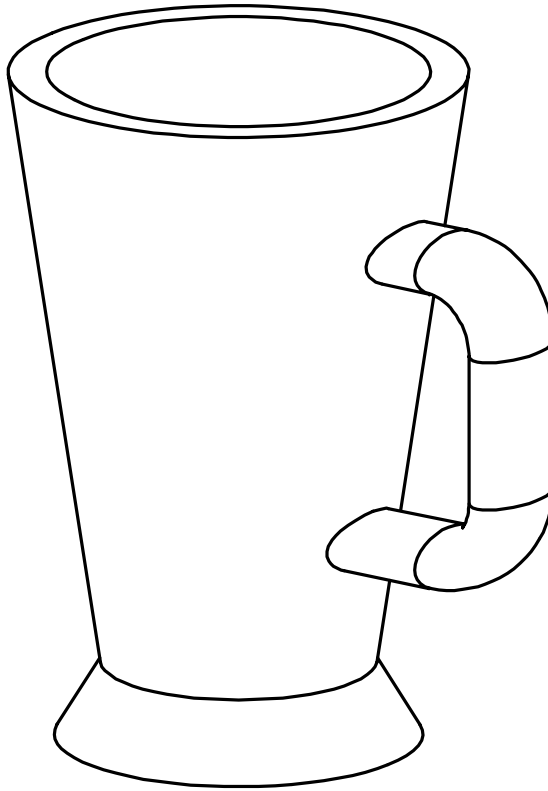
Notice that a boolean parameter **Mug;** was set to **False** in the class definition.

- ☐ Close the Information window and choose **Cancel** in the Add Child Rule dialog.

**Step 5 Turn the cup into a mug.**

- ☐ In the KF Navigator, place the cursor over **my\_mug**, click MB3 and choose **Edit**. The Edit Child Rule dialog appears.
- ☐ In the Input Parameters window of the Edit Child Rule dialog, select the rule **mug: <False;>** and in the Rule for Parameter window change the value from **False;** to **True;** (Be sure to leave the semicolon at the end of the line).
- ☐ Choose **OK** to update the model.

This time, notice that your cup has now turned into a mug. Basically, the boolean rule dictates whether or not a handle and a base are put on the mug. Other items that you can edit include the TopDiameter, Height and WallThickness of the mug, as well as the Length and Width of the handle. Work with this until your instructor is ready to proceed.



**Step 6 Save and Close \*\*\*\_mug.prt.**

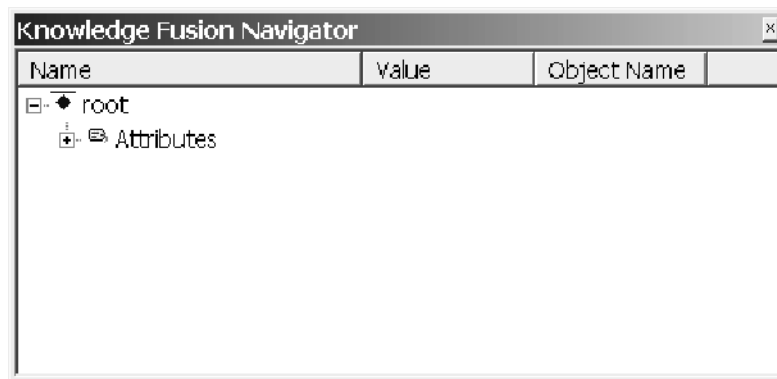
This concludes the activity.

## Activity 2–5: Instantiating a UG System Class

In the activity, you will instantiate a UG–supplied system class that is part of the Knowledge Fusion product and create your own user class from the Knowledge Fusion Navigator for later instantiation.

### Step 1 Create a new part and open the KF Navigator.

- ☐ Choose **File→New**. Enter **\*\*\*\_cylinder** in the File name box, where **\*\*\*** are your initials. Accept **Inches** as the Units. Choose **OK**.
- ☐ From the graphics window, click **MB3**. Choose **Replace View→TFR–TRI**.
- ☐ Choose **View→Knowledge Fusion Navigator** if necessary to turn on the KF Navigator.



### Step 2 Instantiate a ug\_cylinder class based on user defined specifications.

- ☐ Place the cursor over **root** in the Knowledge Fusion Navigator, click **MB3**, and choose **Add Child Rule**. The Add Child Rule dialog appears.

**Create New Child Rule**

Method: Choose Class from List

Name:

Class: ☐ User ☐ System ☒ Both

Filter: \*

beam\_basic\_odbc  
bike\_generative\_1  
bike\_generative\_2  
counterweight  
door  
high\_circ\_outout

Input Parameters

Rule for Parameter

OK Apply Cancel

2

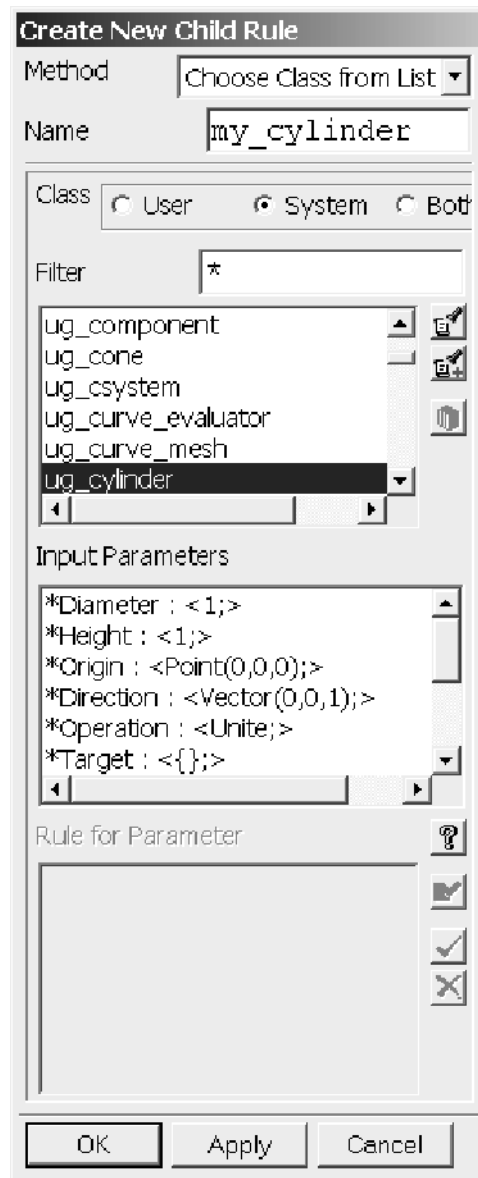
- ☐ In the Name box, enter **my\_cylinder**.
- ☐ Under Class, select the **System** radio button.

Notice that the listing window lists only **ug\_** prefixed classes (as opposed to user defined classes).

- ☐ Select **ug\_cylinder** from the class list.

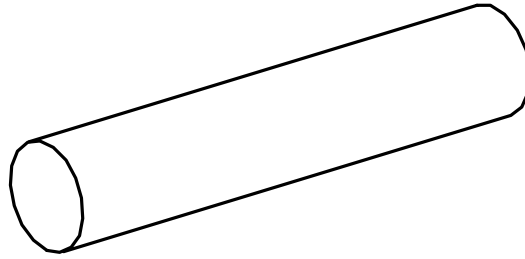


At this point, your Add Child Rule dialog should look as shown below.



- ☐ Under the Input Parameters, simply select each of the following items and enter the value shown. (Be sure to leave the semicolon at the end of the line)
  - **diameter: 3;**
  - **height: 15;**
  - **direction: Vector (0,1,0);**
- ☐ Choose **OK** to create the cylinder.

Your graphics screen should now look as shown below:

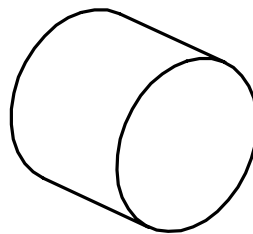


**Step 3 Modify my\_cylinder to reflect new values.**

- ☐ In the KF Navigator, place the cursor over **my\_cylinder**, click MB3 and choose **Edit**. The Edit Child Rule dialog appears.
- ☐ In the Input Parameters window, select the following items one at a time and enter the value shown in the Rule for Parameters window.
  - **diameter: 13;**
  - **height: 12;**
  - **direction: Vector (1,0,0);**
- ☐ Choose **OK** to modify the cylinder.



After the edit, your screen should now look like:

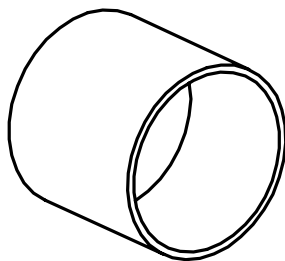


**Step 4 Create a second cylinder and subtract it from my\_cylinder.**

- ☐ Place the cursor over **root** in the Knowledge Fusion Navigator, click **MB3**, and choose **Add Child Rule**. The Add Child Rule dialog appears.

- ☐ In the Name box, enter **sub\_cyl**.
- ☐ Select the class **ug\_cylinder**.
- ☐ In the Input Parameters window, select each of the following items one at a time and enter the value shown in the Rules for Parameters window.
  - **diameter: 12;**
  - **height: 20;**
  - **origin: my\_cylinder:origin: + vector(3,0,0);**
  - **direction: Vector (1,0,0);**
  - **operation: Subtract;**
  - **target: {my\_cylinder:};**
- ☐ Choose **OK** to create and subtract the cylinder.

Your graphics screen should now look as shown below:

**NOTE**

Did you notice what you just did? We've used two primitives in a part file which is typically not recommended in Unigraphics. The reason it is typically not recommended is due to the lack of positional associativity. In this case, however, we have tied the origin of the **sub\_cyl** to the origin of **my\_cylinder** which alleviates the concern over positional associativity.

**Step 5 Save your creation as a new class definition.**

- ☐ Place the cursor over **root** in the Knowledge Fusion Navigator, click **MB3**, and choose **List Rules**. An Information window appears.
- ☐ In the Information window, change **New\_Class\_Name** in the **DefClass** statement to **\*\*\*\_mydesign** where **\*\*\*** represents your initials.
- ☐ If there is a blank line before **#! UG/KF 17.0**, delete that blank line to ensure **#! UG/KF 17.0** is in the first line.
- ☐ In the Information window menu, select **File→Save As**.
- ☐ Browse to **my\_dfa\_files** directory.
- ☐ Enter the file name as: **\*\*\*\_mydesign.dfa**
- ☐ Choose **OK** and close the Information window.


**NOTE**

What we are actually doing here is using a portion of the adoptive approach with reusability. We built a new class using multiple child rules from the current part. This is a very valid approach to KBE. Saving it out to a new dfa file results in the new class that is available to be used by others. Had you simply saved the part file, the definition of the new class would have existed only in the part file.

**Step 6 Save and Close all parts.****Step 7 Create a new part file called \*\*\*\_test to test the design.****Step 8 Add the Knowledge Fusion search directory.**

- ☐ Choose **Preferences→Knowledge Fusion**.
- ☐ Enter the path where the **\*\*\*\_mydesign.dfa** file was just saved in the New Directory box.  
(e.g. **C:\<your home directory>\my\_dfa\_files**)



- ☐ Choose .
- ☐ Choose **OK**.

### Step 9 Reload all classes.

- ☐ Place the cursor over **root** in the Knowledge Fusion Navigator.
- ☐ Press **MB3** and choose **Reload All**.

#### **NOTE**

Because UG is already running, it will not automatically recognize that we have added a new class. Therefore we need to Reload All so it will recognize our newly created class.

### Step 10 Instantiate \*\*\*\_mydesign.

- ☐ Place the cursor over **root** in the Knowledge Fusion Navigator.
- ☐ Press **MB3** and choose **Add Child Rule**.
- ☐ In the Name box, enter **test\_mydesign**.
- ☐ Under Class, select the **User** radio button. Notice that the listing window lists only user defined classes (as opposed to system classes).
- ☐ Choose **\*\*\*\_mydesign**.
- ☐ Choose **OK** to instantiate your new class.

### Step 11 Close all parts.

This concludes the activity.

**SUMMARY**

In this lesson you:

- Learned how to add attributes using the KF Navigator and the Add Attribute dialog.
- Learned how to add child rules using the KF Navigator and the Add Child Rule dialog.
- Edited an existing class using the KF Navigator and the Edit Child Rule dialog.
- Instantiated existing classes using the KF Navigator and the Add Child Rule dialog.
- Created your own class using the KF Navigator and the List Rules option.

**2**

2

**(This Page Intentionally Left Blank)**

# Design Control

## Lesson 3

### PURPOSE

To learn the different ways to control your design in Knowledge Fusion.

### OBJECTIVES

Upon completion of this lesson, you will be able to:

- Control the topology by using rules to define the class.
- Control and examine the feature mass properties.
- Control the impact of interactive edits.
- Use `ug_expression` class to create expressions.



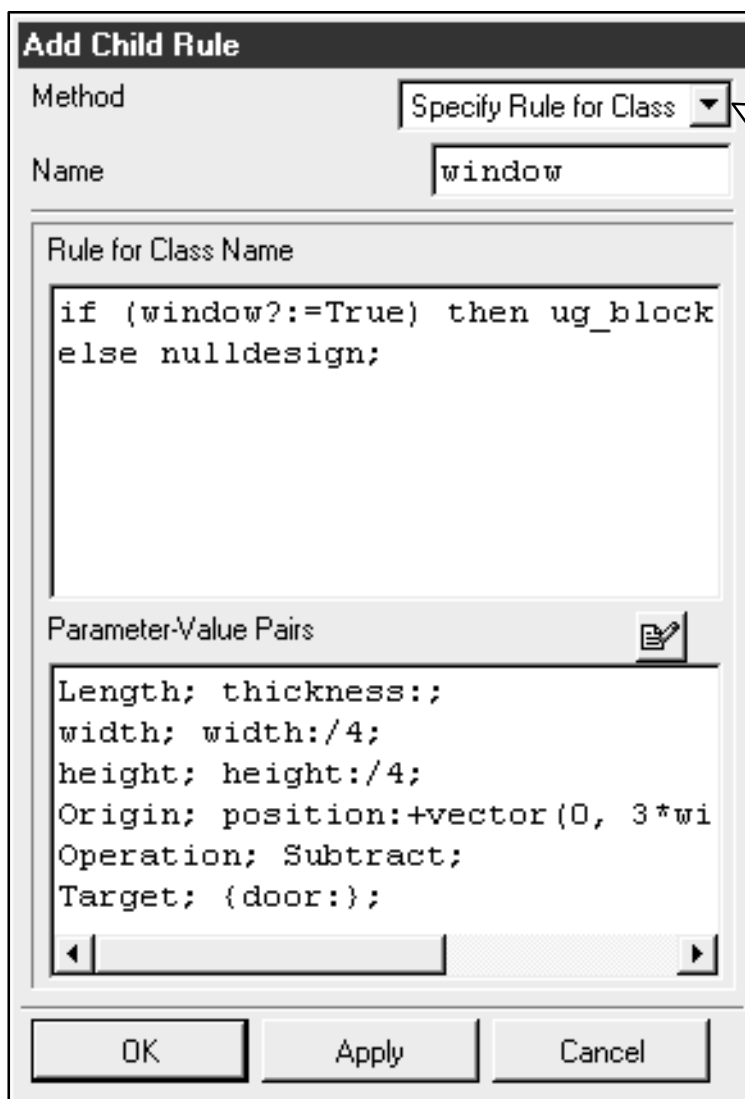
This lesson contains the following activities.

Activity	Page
3-1 Control the Door to Have a Dynamic Window ..	3-5
3-2 Examine the Mass Properties of the Door .....	3-9
3-3 Control Interactive Edits of the Door .....	3-13
3-4 Use UG Expression to Control a Block .....	3-18



## Controlling Topology

In the previous lesson, we used the Add Child Rule dialog in the Knowledge Fusion Navigator to instantiate classes from list, either user defined classes or UG system classes. Then we can control the class rules by editing the parameters or attributes of the class instantiated. Is there a way that we can dynamically control what class will be instantiated or saying the name of the class can be controlled by a rule? Here is the answer: We can use the method option *Specify Rule for Class* in the Add Child Rule dialog or the Edit Child Rule dialog to achieve this.

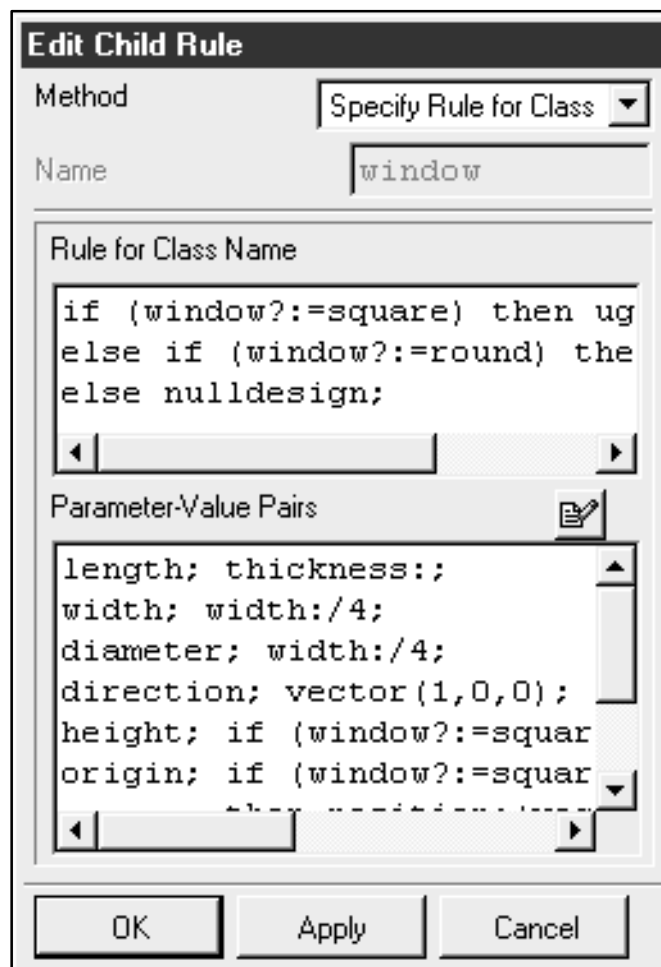


Choose the Method  
Option – Specify  
Rule for Class

The Procedure to Add Child Rule using the method of Specify Rule for Class:

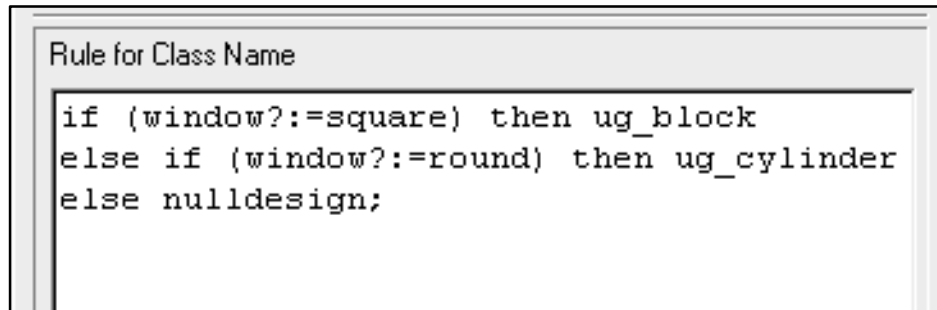
1. Enter an instance name
2. Enter a rule that returns a class name, e.g.:  
if (A:=1) then ug\_point  
else NullDesign;
3. Enter any parameter/value pairs, for example:  
position, Point(1, 1, 1);
4. Choose Apply or OK

We can apply the method of Specify Rule for Class to an already existing child rule via Edit Child Rule dialog. Just place your cursor over the child rule you want to modify and click MB3, then choose Edit. After the Edit Child Rule dialog pops up, choose Specify Rule for Class from the Method option list, and then follow the procedure exactly as we did for the Add Child Rule dialog above except that we need not to enter the name (The Name field is greyed out).

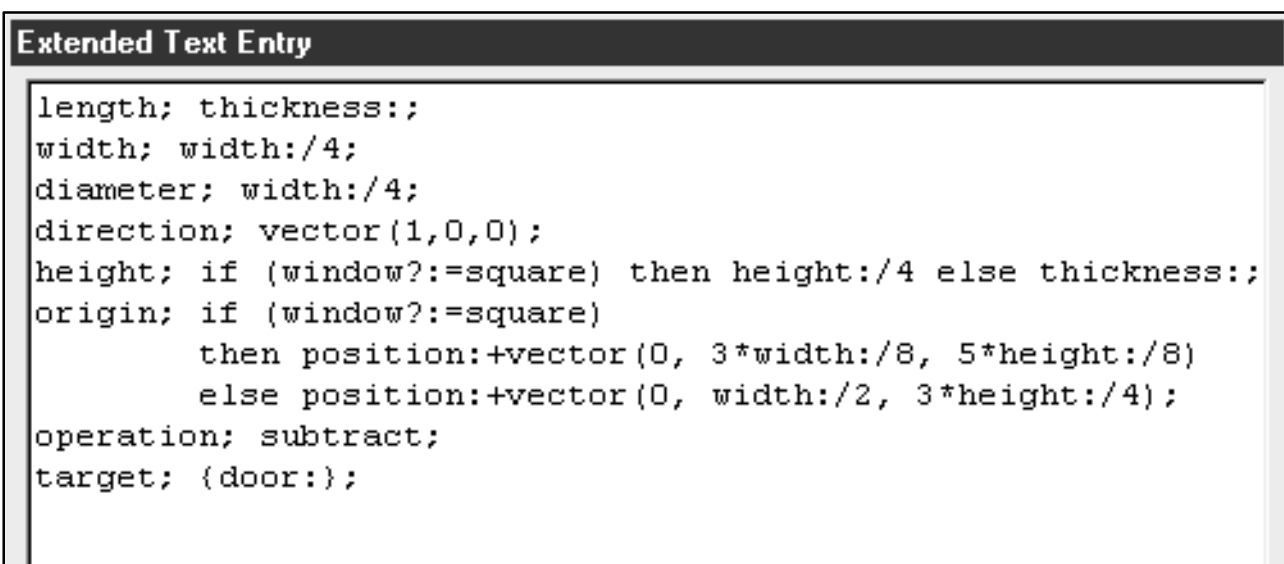


Let's take a look at an example of a dynamic window which can have different shapes controlled by an attribute.

- Rule for Class Name



- Parameter–Value Pairs



The inputs in the Parameter–Value Pairs will be passed to dynamic children, in this example are ug\_block and ug\_cylinder.

**NOTE**

Those inputs that are not parameters to a class are ignored. For example, if the ug\_cylinder class is picked, the length and width parameter in the Parameter–Value Pairs will be ignored. Those inputs that are common but have different meanings need to use if, then, else, such as the height and origin parameters in the above example.

## Activity 3– 1: Control the Door to Have a Dynamic Window

In this activity, you will become familiar with controlling the topology by using rules to define the class so as to create a dynamic window for the door.

### Step 1 Add the window? attribute.

- ☐ If **\*\*\*\_door.prt** is not open, retrieve this part and open the KF Navigator. Click the **+** sign on Attributes under the root node.
- ☐ Place the cursor over **root** and click MB3. Choose **Add Attribute**. The Add Attribute dialog displays.
- ☐ Enter **window?** in the Name box.
- ☐ From the Type list, choose **Name**.
- ☐ In the Formula box, enter **square**.
- ☐ Select the **Input Parameter** check box (it should be checked). Verify that **Modifiable** is checked and that **Uncached** is unchecked. **Evaluate It** should be checked.
- ☐ Choose **OK**.



### Step 2 Edit the window child rule.

- ☐ Place the cursor over the **window** child node and click MB3. Choose **Edit**. The Edit Child Rule dialog displays.
- ☐ From the Method list, choose **Specify Rule for Class**.
- ☐ In the Rule for Class Name box, enter:

```
if (window?:=square) then ug_block  
else if (window?:=round) then ug_cylinder  
else nulldesign;
```

- ☐ Click on the Extended Text Entry button  and enter the following in the Extended Text Entry dialog:

```
length;thickness;;
width;width:/4;
diameter;width:/4;
direction;vector(1,0,0);
height;if (window?:=square) then height:/4 else thickness;;
origin;if (window?:=square)
    then position:+vector(0, 3*width:/8, 5*height:/8)
    else position:+vector(0, width:/2, 3*height:/4);
operation;subtract;
target;{door:};
```

- ☐ Choose **OK** on the Extended Text Entry dialog.
- ☐ Choose **OK** on the Edit Child Rule dialog.
- ☐ Save the part.

### Step 3 Test the design.

- ☐ In the KF Navigator expand the root Attributes by clicking the + sign.
- ☐ Place the cursor over the **window?** attribute, click MB3 and choose **Edit**. The Edit Attribute dialog displays.
- ☐ Edit **square** to be **round**. Choose **Apply**. Notice that the window changes to round.
- ☐ Edit **round** to be **none**. Choose **Apply**. Notice that the window is deleted.
- ☐ Close the part.

This concludes the activity.

## Controlling Color, Layer, Suppression Status

We can control the color, layer and suppression status (Suppress?) from the Input Parameters in the Add Child Rule dialog or the Edit Child Rule dialog.

- Suppress?: True or False
- Color: Integer input
- Layer: Integer input (1 to 256)

**Add Child Rule**

Method: Choose Class from List

Name: line

Class: ☐ User ☒ System ☐ Both

Filter: \*

Class List:

- ug\_line
- ug\_mass\_properties
- ug\_note
- ug\_number\_product\_attribute
- ug\_odbc\_database
- ug\_odbc\_recordset
- ug\_offset\_angle\_chamfer

Input Parameters:

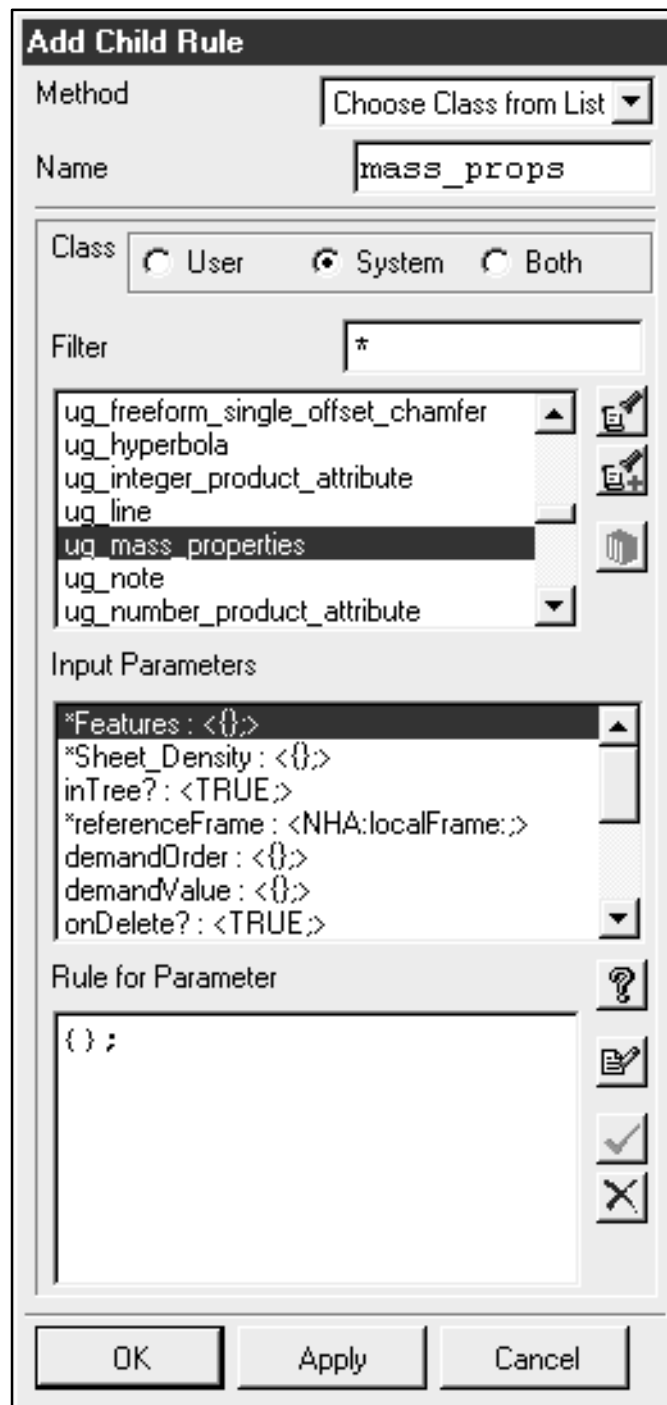
- onDelete? : <TRUE;>
- lockRule? : <FALSE;>
- lockRuleParameters : <{};>
- lockObject? : <FALSE;>
- \*Suppress? : <FALSE;>
- \*color : <2;>
- \*layer : <1;>

Rule for Parameter

Suppression Status,  
Color, and Layer  
Control Parameters

## Mass Properties

We can control and examine the feature mass properties using `ug_body` class for a single feature, or using `ug_mass_properties` class for multiple features.



## Activity 3–2: Examine the Mass Properties of the Door

In this activity, you will change the door color and examine its mass properties via `ug_body` class.

### Step 1 Add the `door_body` child rule.

- ☐ If `***_door.prt` is not open, retrieve this part and open the KF Navigator.
- ☐ Place the cursor over **root** and click MB3. Choose **Add Child Rule**.
- ☐ In the Name box, enter **door\_body**.
- ☐ In the Class list select **ug\_body**.
- ☐ From the Input Parameters box, select **\*feature:<{};>**.
- ☐ Place the cursor inside the braces `{}`. Type **door:**.
- ☐ From the Input Parameters box, select **\*color:<2;>**.
- ☐ In the Rule for Parameter box, remove **2** and enter **11**.
- ☐ Choose **OK**.
- ☐ Save the part.





**Step 2 Examine the mass of the door solid.**

- ☐ Place the cursor over **door\_body** and click MB3. Choose **Inspect Instance**.
- ☐ In the Instance Inspector dialog, expand the **door\_body** Attributes by clicking the **+** sign.
- ☐ See how the mass of the door solid is affected by the window attribute by changing the **window?** attribute using **Edit Attribute** in the Knowledge Fusion Navigator and using **Show Value** on the **mass** attribute in the Instance Inspector.
- ☐ Close the part.

This concludes the activity.



## Controlling Interactive Edit

Knowledge Fusion and the interactive system both create the identical set of UG objects. The Knowledge Fusion information and all information made available through the interactive system of the same object are always synchronized. If an object is edited interactively, say a modeling dimension, then Knowledge Fusion will automatically reflect that change; the reverse is also true.

You can control the impact of interactive edits when an object has a corresponding Knowledge Fusion rule. The following parameters from the Input Parameters box in the Add Child Rule dialog or the Edit Child Rule dialog are supplied for this purpose.

- onDelete?
- lockRule?
- lockRuleParameters
- lockObject?

**Edit Child Rule**

Method: Choose Class from List

Name: door

Class: ☐ User ☐ System ☒ Both

Filter: \*

Class List: bike\_generative, bolt, drive\_coupling, mug, nut, un\_arc

**Input Parameters**

demandOrder : <{}>  
demandValue : <{}>  
onDelete? : <TRUE>  
lockRule? : <FALSE>  
lockRuleParameters : <{}>  
lockObject? : <FALSE>  
\*Suppress? : <FALSE>

Rule for Parameter

Interactive Edits  
Control Parameters

3

- **onDelete? (Boolean)**

TRUE means the instance can be deleted outside of KF. When used, the onDelete? value is usually defined by a rule that evaluates to True or False rather than a static value of False.

- **lockRule? (Boolean)**

TRUE means that the instance cannot be changed outside of KF (i.e. Modeling). For example, if a ug\_block instance was locked with lockRule?, then the size of the block cannot be changed in Modeling by editing expressions or by editing feature parameters, but it can be changed by KF rules or with Edit Child Rule.

- **lockRuleParameters**

This is a list of parameter names that need to be locked against changes in Modeling. (If lockRule? is TRUE, lockRuleParameters is ignored.) For example, to prevent the user from changing the Length or Width of an instance of ug\_block in Modeling by editing expressions or editing feature parameters, set the value of lockRuleParameters in the block instance to {Length,Width}.

- **lockObject? (Boolean)**

TRUE means that the instance cannot be changed by Knowledge Fusion, even if a KF rule attempts to change a parameter value. However, the instance can be modified in Modeling.

## Activity 3–3: Control Interactive Edits of the Door

In this activity, you will use Knowledge Fusion rule to control the impact of interactive edits from Modeling and the Expression sub–system.

### Step 1 Edit the door width in Modeling.

- ☐ If **\*\*\*\_door.prt** is not open, retrieve it and open the KF Navigator.
- ☐ Expand the Attributes under **door** child (NOT the root Attributes) by clicking the **+** sign.
- ☐ Notice the current value of the **width** attribute.
- ☐ Choose **Application→Modeling**.
- ☐ Choose **Edit→Feature→Parameters**.
- ☐ Select **BLOCK(0)**
- ☐ Choose **OK**.
- ☐ Choose the **Feature Dialog** button.
- ☐ Change the Y Length to **40**, and hit **OK** until the block updates.

Notice the **width** attribute is now blue and its value is 40.

- ☐ Place the cursor over **width** and click MB3. Choose **Delete**.

Notice the width attribute is once again black and back to its original value.



**Step 2 Lock the door width and try to edit it in Modeling.**

- ☐ Place the cursor over **door** and click MB3. Choose **Edit**. The Edit Child Rule dialog displays.
- ☐ From the Input Parameters box, select **lockRuleParameters:<{};>**.
- ☐ Place the cursor inside the braces {}. Type **width**.
- ☐ Choose **OK**.
- ☐ Choose **Edit→Feature→Parameters**.
- ☐ Select **BLOCK(0)**.
- ☐ Change the Y Length to **40** and the Z Length to **90**, and choose **OK** until the part updates.

Notice the **width** attribute is still black and its current value remains the same, while the **height** attribute is now blue and its value is 90. The reason that we can not change the **width** outside of KF is that it was locked by lockRuleParameters.

- ☐ Place the cursor over **height** and click MB3. Choose **Delete**.

Notice the **height** attribute is once again black and back to its original value.

**Step 3 Lock the door, window, and knob.**

- ☐ Place the cursor over **door** and click MB3. Choose **Edit**. The Edit Child Rule dialog displays.
- ☐ From the Input Parameters box, select **onDelete?:<TRUE>**.
- ☐ In the Rule for Parameter box, change **TRUE** to **FALSE**.
- ☐ From the Input Parameters box, select **lockRule?:<FALSE>**.
- ☐ In the Rule for Parameter box, change **FALSE** to **TRUE**.



- ☐ From the Input Parameters box, select **lockRuleParameters:{width};**.
- ☐ In the Rule for Parameter box, change **{width}** to **{}**.
- ☐ Choose **OK**.
- ☐ Place the cursor over **window** and click MB3. Choose **Edit**. The Edit Child Rule dialog displays.
- ☐ From the Input Parameters box, select **onDelete?:<TRUE>**.
- ☐ In the Rule for Parameter box, change **TRUE** to **FALSE**.
- ☐ From the Input Parameters box, select **lockRule?:<FALSE>**.
- ☐ In the Rule for Parameter box, change **FALSE** to **TRUE**.
- ☐ Choose **OK**.
- ☐ Place the cursor over **knob** and click MB3. Choose **Edit**. The Edit Child Rule dialog displays.
- ☐ From the Input Parameters box, select **onDelete?:<TRUE>**.
- ☐ In the Rule for Parameter box, change **TRUE** to **FALSE**.
- ☐ From the Input Parameters box, select **lockRule?:<FALSE>**.
- ☐ In the Rule for Parameter box, change **FALSE** to **TRUE**.
- ☐ Choose **OK**.
- ☐ Save the part.



**Step 4 Test the design.**

- ☐ Try changing any of the feature parameters using **Edit→Feature→Parameters**.

You will get an error message indicating that “Knowledge Fusion error – Cannot modify object controlled by rule”. This is because the objects were locked by lockRule?.

- ☐ Try suppressing any of the features using **Edit→Feature→Suppress**

You will get the same error message as above due to the same reason.

- ☐ Try deleting any of the features using **Edit→Feature→Delete**

You will get the same error message as above due to the same reason.

- ☐ Close the part.

This concludes the activity.

## ***KF and the Expression Tool***

Expressions are the mechanism used within Unigraphics to establish parametric relationships. In particular, expressions control features so features are edited by modifying their controlling expressions. Also, a single expression can be used both directly and indirectly to control multiple features and associations. Consequently, if a single expression is used to control several features, the editing of that expression causes each of the dependent features to update.

In Knowledge Fusion, we can also create expressions using the `ug_expression` class. The expressions available via the Knowledge Fusion are the same expressions that are visible in interactive Unigraphics in the Expression Editor. With the `ug_expression` class it is easy to swap from KF to expressions and vice versa. It is a powerful tool because it becomes easy to embed KF in your already existing expression-driven CAD design.





## *Activity 3–4: Use UG Expression to Control a Block*

In this activity, you will use the `ug_expression` class to create UG expressions, and apply them to control the angular rotation and the length of a block.

### **Step 1 Create a new part and open the KF Navigator.**

- ☐ Choose **File**→**New**. Enter **\*\*\*\_expression** in the File name box, where **\*\*\*** are your initials. Accept **Inches** as the Units. Choose **OK**.
- ☐ Choose **View**→**Knowledge Fusion Navigator** to turn on the KF Navigator.

### **Step 2 Instantiate a `ug_expression` class that will control the angular rotation of a block**

- ☐ Place the cursor over **root** in the Knowledge Fusion Navigator, click **MB3**, and choose **Add Child Rule**. The Add Child Rule dialog appears.
- ☐ Enter **angle** in the Name box.
- ☐ Check the **System** button in the Class selection.
- ☐ Select **`ug_expression`** from the class list.
- ☐ From the Input Parameters box, select **\*value: <0;>**.
- ☐ In the Rule for Parameter box, change the value to **10**.
- ☐ Choose **Apply**.

**Step 3 Instantiate a second `ug_expression` class that will control the length of a block.**

- ☐ Enter **length** in the Name box.
- ☐ Select **ug\_expression** from the class list.
- ☐ From the Input Parameters box, select **\*value: <0;>**.
- ☐ In the Rule for Parameter box, change the value to **0.5**.
- ☐ Choose **Apply**.

**Step 4 Instantiate a `ug_csystem` class to control the placement (origin) and orientation of the block.**

- ☐ Enter **block\_csys** in the Name box.
- ☐ Select **ug\_csystem** from the class list.
- ☐ In the Input Parameters window, select the following items one at a time and enter the value shown in the Rules for Parameters window.
  - **origin: Point (1, 1, 1);**
  - **x\_axis: Vector (cos (angle:value:), sin (angle:value:), 0);**
  - **y\_axis: Perpendicular (block\_csys:x\_axis:);**
- ☐ Choose **Apply**.

**Step 5 Instantiate a `ug_block` class for the block.**

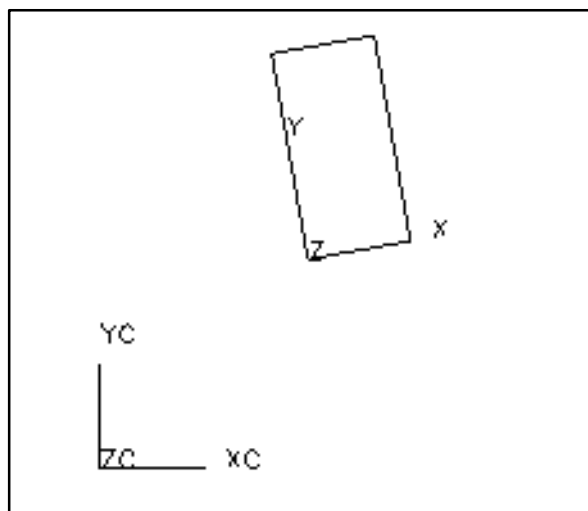
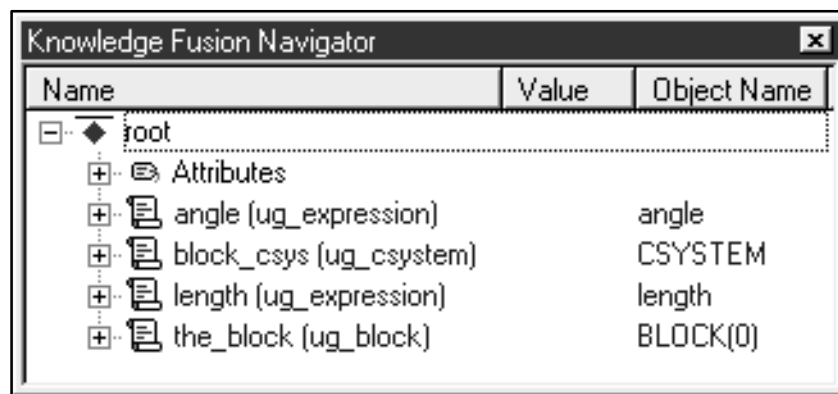
- ☐ Enter **the\_block** in the Name box.
- ☐ Select **ug\_block** from the class list.

- ☐ Set the following attributes, as indicated, in the Rule for Parameter box. Leave the remaining attributes at their default value.

- **length:** length:value;;
- **referenceFrame:** block\_csys:csystem;;

- ☐ Choose **OK**.

Your object tree should appear as follows, and your block object should appear in the graphics area as shown below.



**Step 6 Test the design.**

- ☐ Choose **Application**→**Modeling** to start the Modeling application.
- ☐ Choose **Tools**→**Expression ...** from the main menu to open the Expressions dialog. You should see the following modifiable values in the dialog:  
**angle=10**  
**length=0.5**
- ☐ Change these values in the dialog and the block object's orientation and length will change accordingly.
- ☐ Close the Expressions dialog.
- ☐ Go to the Knowledge Fusion Navigator, use the Edit Child Rule dialog to change the values of **angle** and **length** child rules. The block object's orientation and length will change accordingly.



This concludes the activity.

## SUMMARY

In this lesson you:

- Learned how to control the topology by using rules to define the class.
- Learned how to control and examine the feature mass properties.
- Learned how to control the impact of interactive edits.
- Used `ug_expression` class to create expressions.

3



# Adoption

## Lesson 4

---

**PURPOSE**

To give insight into the adoptive approach using Knowledge Fusion.

**OBJECTIVES**

Upon completion of this lesson, you will be able to:

- Adopt existing objects into the Knowledge Fusion application.
- Add rules to control the adopted features.

This lesson contains the following activities.

Activity	Page
4-1 Adopting Existing Objects .....	4-6
4-2 Adoption with Blend Position Change .....	4-12

## ***The Adoption Concept***

Traditional Knowledge Based Engineering applications have been based mainly on a programming language and have been generative in their approach. That is to say that the resulting engineering documentation resulted directly from a text file.

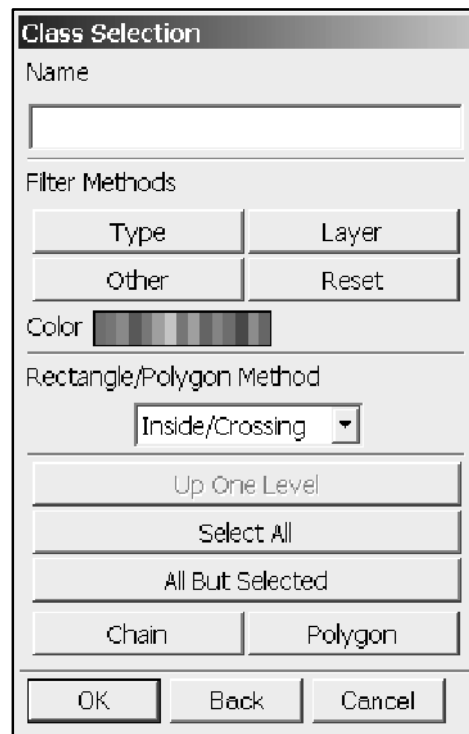
With the advent of the UG Knowledge Fusion application, users will now have the ability to bring existing geometric objects into their session and then apply simple or complex engineering rules to drive the geometry. This approach is referred to as the Adoptive approach to KBE.

A common usage of KF is to subject a UG object to engineering rule control. UG has a process called adoption that automatically creates a rule for an existing UG object. The adoptive approach enables geometric creation to take place by the usual, efficient, graphical techniques in UG. Additionally, adoption allows the use of engineering rules and knowledge bases by informing the language about selected parameters of the adopted object. This removes the burden of textually describing a complex modeling feature while enabling the use of knowledge bases and rules to manipulate the object.

These rules, along with the definition of geometry, may then be formatted to a text file for instantiation in another session. This approach is referred to as Adoption with Reusability. Further more, because of the interpretive feature of the Knowledge Fusion language, the user is then free to simply edit the resultant text file to make modifications.

## Adopt Existing Object

The Adopt Existing Object choice from the Main Menu (**Tools**→**Knowledge Fusion**→**Adopt Existing Object**) opens the standard UG Class Selection dialog shown in the following illustration, which allows you to incorporate object specifications not created by UG/Knowledge Fusion, in other words, the process of exposing (a subset of) an existing UG object's defining attributes in Knowledge Fusion.





This is a powerful capability for the UG/Knowledge Fusion application, because it allows you to bring objects in previously created UG part files under UG/KF control. Adoption is particularly useful in cases where you might want to interactively create a feature in Modeling, and then adopt it into a UG/KF part file. We can adopt UG features, bodies, curves, or other UG object types. Available object types can be reviewed by choosing the Type option button in the Class Selection dialog. Features included in the KF system class definitions are directly adoptable, meaning that their defining parameters appear in the KF Navigator's Input Parameters list box, available for edit.

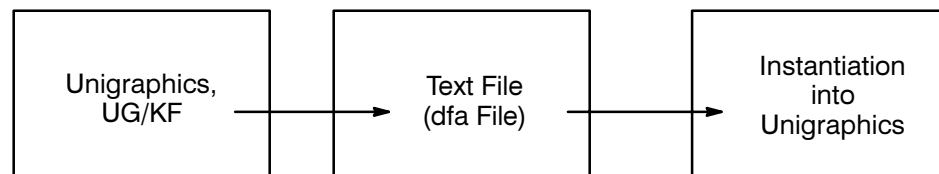
**NOTE**

Not all features have system classes. If there is a system class for an adopted object, Knowledge Fusion will control object via class parameters; If there is NO system class for an adopted object, the object will be controlled by expressions. As an example, the creation parameters for a boss (a non – KF supported feature) are Diameter, Height, and Taper Angle. Once a boss is adopted, these three parameters appear in the object tree as children expressions under the adopted boss. As such, they can, of course, be selected and edited.

## The Procedure of Adoption

The basic procedure to adopt a UG object is as follows:

- Step 1** With a part file of interest open, choose the Adopt Existing Object command, which opens the Class Selection dialog.
- Step 2** Select the object(s) for adoption in the graphics area, or from the Model Navigator, and choose OK on the Class Selection dialog.
- Step 3** Turn on the Knowledge Fusion Navigator. Now, in the Object Tree, under the root area, the object(s) previously selected for adoption appear in blue.
- Step 4** (This is an optional step for Adoption with Reusability) Place your cursor over root, press MB3 and select List Rules to show the rules in the Listing Window which can then be saved to a dfa file (as completed in the previous lesson).



## Activity 4–1: Adopting Existing Objects

In this activity, you will use traditional techniques to manually create a solid body consisting of several features. Then you will adopt some of these features and add rules to control and interrelate the features.

Some of the features have corresponding Knowledge Fusion classes and some do not. Compare how these two types of features get adopted into Knowledge Fusion and how you can control them.

**NOTE**

It is assumed that you already understand the modeling techniques for manually creating blocks, bosses, holes, and blends. Not all of the details for creating features will be provided.

### Step 1 Open a new part and enter the Modeling application.

- ☐ Choose **File**→**New**. Enter **\*\*\*\_boss** in the File name box, where **\*\*\*** represents your initials. Accept **Inches** as the Units. Choose **OK**.
- ☐ Click MB3 from the graphics window. Choose **Replace View**→**TFR–TRI**.
- ☐ Choose **View**→**Model Navigator**.
- ☐ Choose **Application**→**Modeling**.

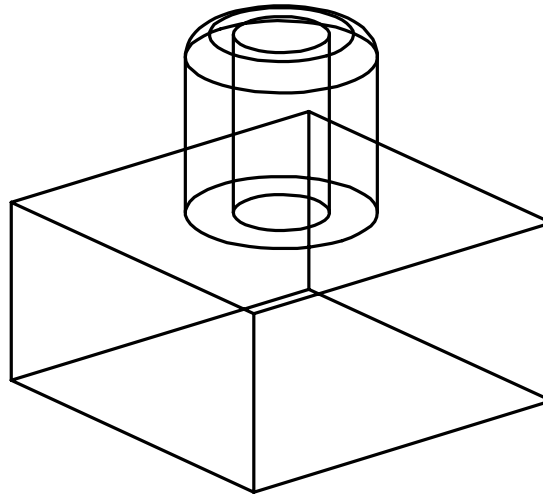
### Step 2 Create the geometry.

- ☐ Create a **Block** to form the base for this object.

Length:	2
Width:	2
Height:	1
- ☐ Add a **Boss** centered in the top face of the Block.

Diameter:	1
Height:	1
Taper Angle:	0

- ☐ Add a **Hole** centered in the top face of the Boss.  
Diameter: **0.5**  
Depth: **1**  
Tip Angle: **0**
- ☐ Add an **Edge Blend** to the top outside edge of the Boss.  
Radius: **0.125**

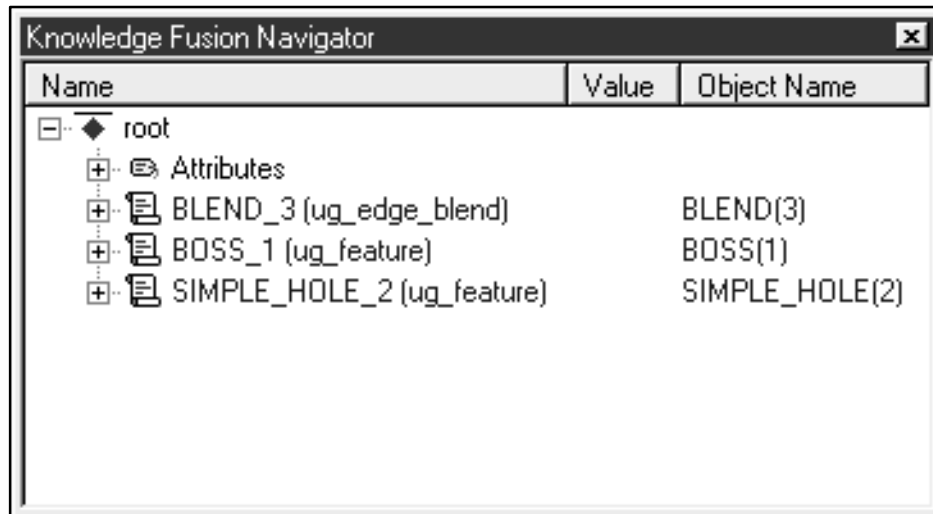


**Step 3 Adopt all of the features except the Block into Knowledge Fusion.**

- ☐ Choose **View**→**Knowledge Fusion Navigator**.
- ☐ Choose **Tools**→**Knowledge Fusion**→**Adopt Existing Object**, the Class Selection dialog displays.
- ☐ In the Model Navigator, press **Ctrl** key to select **BOSS(1)**, **SIMPLE\_HOLE(2)**, and **BLEND(3)**.
- ☐ Choose **OK** on the Class Selection dialog.

The Boss, Blend and Hole were adopted into Knowledge Fusion. The KF Navigator should appear similar to the one shown in the following figure.



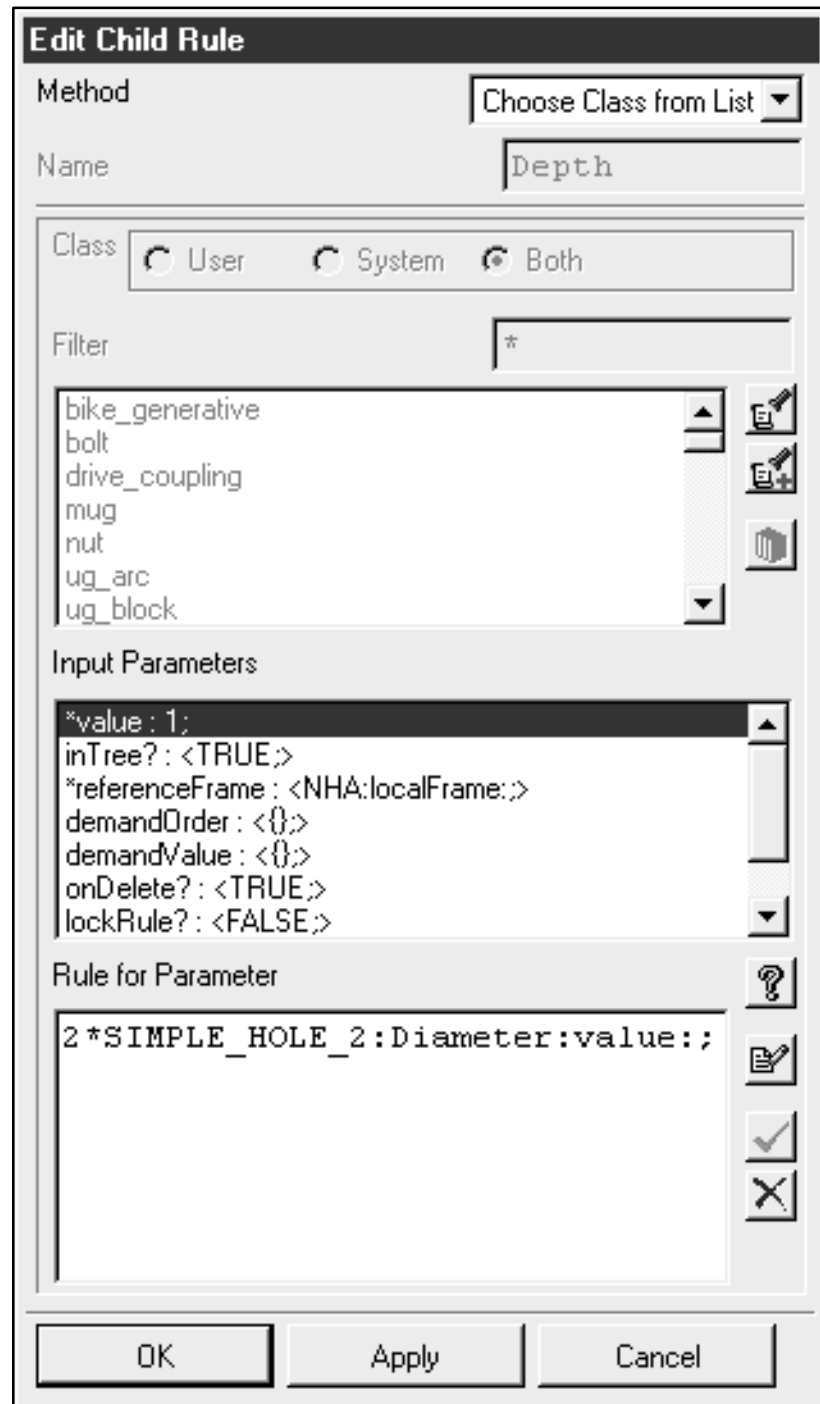


**NOTE** Notice that the Blend was adopted into Knowledge Fusion as an instance of the class `ug_edge_blend`, while the Boss and Hole were adopted as instance of `ug_feature`. Any feature that does not have a corresponding class is adopted as an instance of `ug_feature`.

**Step 4 Add a rule to the Hole object that relates its depth to its diameter.**

- ☐ In the KF Navigator expand **SIMPLE\_HOLE\_2** by clicking the + sign.
- ☐ Place the cursor over **Diameter** and click MB3. Choose **Inspect Instance**.
- ☐ In the Instance Inspector, expand the Diameter Attributes by clicking the + sign.
- ☐ In the KF Navigator, place the cursor over **Depth** and click MB3. Choose **Edit**. The Edit Child Rule dialog displays.
- ☐ From the Input Parameters box, select **\*value: 1;**
- ☐ In the Rule for Parameter box, backspace over the 1 so that only the semicolon displays. Enter **2\***.

- ❑ In the Instance Inspector, place the cursor over the **value** attribute. Click MB3 and choose **Reference**. The Edit Child Rule dialog should look like this:



- ❑ Choose **OK** to close Edit Child Rule dialog.

**Step 5 Add rules to the Boss object that relates its diameter and height to the Hole's diameter.**

- ☐ In the KF Navigator expand **BOSS\_1** by clicking the **+** sign.
- ☐ Place the cursor over **BOSS\_1 Diameter** and click MB3. Choose **Edit**. The Edit Child Rule dialog displays.
- ☐ From the Input Parameters box, select **\*value: 1;**.
- ☐ In the Rule for Parameter box, backspace over the **1** so that only the semicolon displays. Enter **2\***.
- ☐ In the Instance Inspector, place the cursor over the **value** attribute. Click MB3 and choose **Reference**.
- ☐ Choose **OK** to close Edit Child Rule dialog.
- ☐ Place the cursor over **BOSS\_1 Height** and click MB3. Choose **Edit**. The Edit Child Rule dialog displays.
- ☐ From the Input Parameters box, select **\*value: 1;**.
- ☐ In the Rule for Parameter box, backspace over the **1** so that only the semicolon displays. Enter **2\***.
- ☐ In the Instance Inspector, place the cursor over the **value** attribute. Click MB3 and choose **Reference**.
- ☐ Choose **OK** to close Edit Child Rule dialog.

**Step 6 Add a rule to the Blend object that relates its radius to the Hole's diameter.**

- ☐ Place the cursor over **BLEND\_3** and click MB3. Choose **Edit**. The Edit Child Rule dialog displays.
- ☐ From the Input Parameters box, select **radius: 0.125;**.
- ☐ In the Rule for Parameter box, backspace over the **0.125** so that only the semicolon displays. Enter **0.25\***.

- ☐ In the Instance Inspector, place the cursor over the **value** attribute. Click MB3 and choose **Reference**.
- ☐ Choose **OK** to close Edit Child Rule dialog.
- ☐ Close the Instance Inspector dialog.
- ☐ Save the part.

#### Step 7 Test the rules.

- ☐ Place the cursor over SIMPLE\_HOLE\_2 **Diameter** and click MB3. Choose **Edit**. The Edit Child Rule dialog displays.
- ☐ From the Input Parameters box, select **\*value: 0.5;**.
- ☐ In the Rule for Parameter box, backspace over **0.5** so that only the semicolon displays. Enter any value from **0.1** to **1.0**.
- ☐ Choose **Apply**.
- ☐ Notice that all of the geometry updates accordingly.
- ☐ Close the part.

This concludes the activity.



## Activity 4–2: Adoption with Blend Position Change

In this activity, you will use several ways to adopt blends and edges from an existing object. Then you will add rules to control which edges get blended.

### Step 1 Open the part file.

- ☐ In your **parts** directory, open the part file **knf\_adoption.prt**.
- ☐ Save the file as **\*\*\*\_adoption.prt** at your working directory.
- ☐ Choose **Application→Modeling...**
- ☐ Make sure that Model Navigator and Knowledge Fusion Navigator are open.

### Step 2 Adopt blends from Model Navigator.

- ☐ Choose **Tools→Knowledge Fusion→Adopt Existing Object**, the Class Selection dialog displays.
- ☐ Press **Ctrl** key to select both **BLEND(4)** and **BLEND(5)** from the Model Navigator.
- ☐ Choose **OK** from the Class Selection dialog.

Notice that the two blends were adopted as child rule in the Knowledge Fusion Navigator.

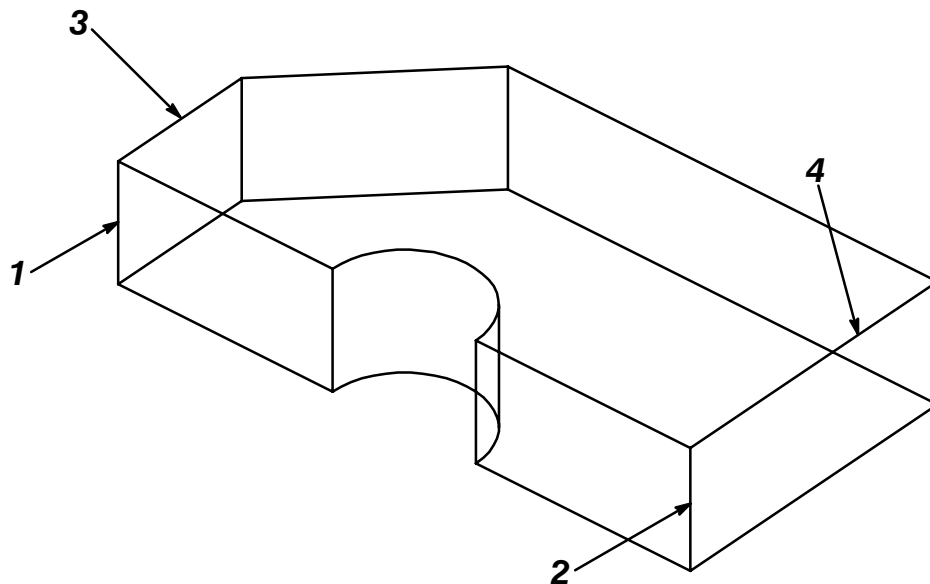
### Step 3 Suppress the following features in order one by one.

- ☐ In the Model Navigator, place the cursor over **HOLLOW(7)** and click MB3. Choose **Suppress**. (You can also achieve this by just clicking on the checkbox beside the feature name in the Model Navigator.)

- ☐ In the Model Navigator, place the cursor over **BLEND(5)** and click MB3. Choose **Suppress**.
- ☐ In the Model Navigator, place the cursor over **BLEND(4)** and click MB3. Choose **Suppress**.

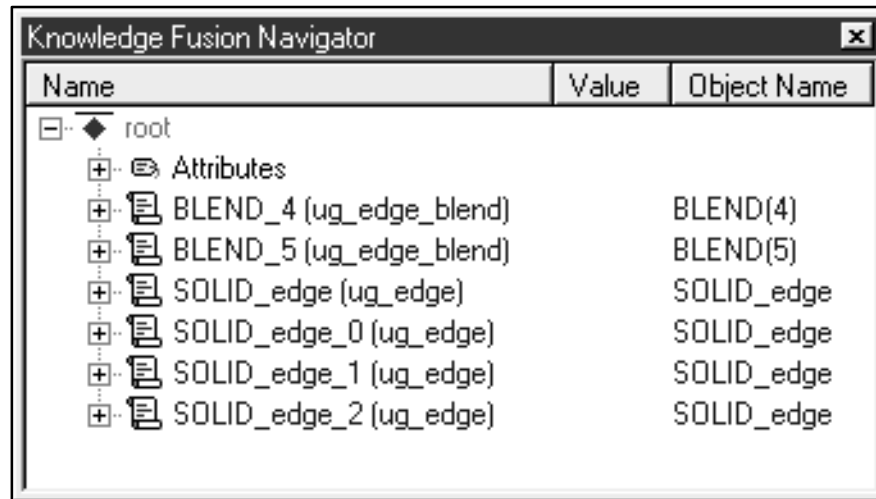
**Step 4 Adopt edges from the graphics window.**

- ☐ Choose **Tools**→**Knowledge Fusion**→**Adopt Existing Object**, the Class Selection dialog displays.
- ☐ Click **Type** button under Filter Methods, Choose **Edge** and click **OK**.
- ☐ Pick the following four edges in order one by one from the graphics window.



- ☐ Choose **OK** from the Class Selection dialog.

The KF Navigator should appear similar to the one shown in the following figure.



**NOTE** The four edges (1–4) were adopted into Knowledge Fusion at an order of SOLID\_edge, SOLID\_edge\_0, SOLID\_edge\_1 and SOLID\_edge\_2 correspondingly.

#### Step 5 Unsuppress the features that were suppressed before.

- ☐ In the Model Navigator, place the cursor over **BLEND(4)** and click MB3. Choose **Unsuppress**. (You can also achieve this by checking on the checkbox beside the feature name in the Model Navigator.)
- ☐ In the Model Navigator, place the cursor over **BLEND(5)** and click MB3. Choose **Unsuppress**.
- ☐ In the Model Navigator, place the cursor over **HOLLOW(7)** and click MB3. Choose **Unsuppress**.

#### Step 6 Add Blend\_Position attribute.

- ☐ From KF Navigator, place the cursor over **root** and click MB3. Choose **Add Attribute**. The Add Attribute dialog displays.
- ☐ Enter **Blend\_Position** in the Name box.
- ☐ Choose **Integer** from the Type list.

☐ Enter **1** in the Formula box.

☐ Choose **Apply**.

**Step 7 Add the edge\_list\_1 attribute.**

☐ Enter **edge\_list\_1** in the Name box.

☐ Choose **List** from the Type list.

☐ In the Formula box, enter:

**{SOLID\_edge;; SOLID\_edge\_0;}**

☐ Choose **Apply**.

**Step 8 Add the edge\_list\_2 attribute.**

☐ Enter **edge\_list\_2** in the Name box.

☐ Choose **List** from the Type list.

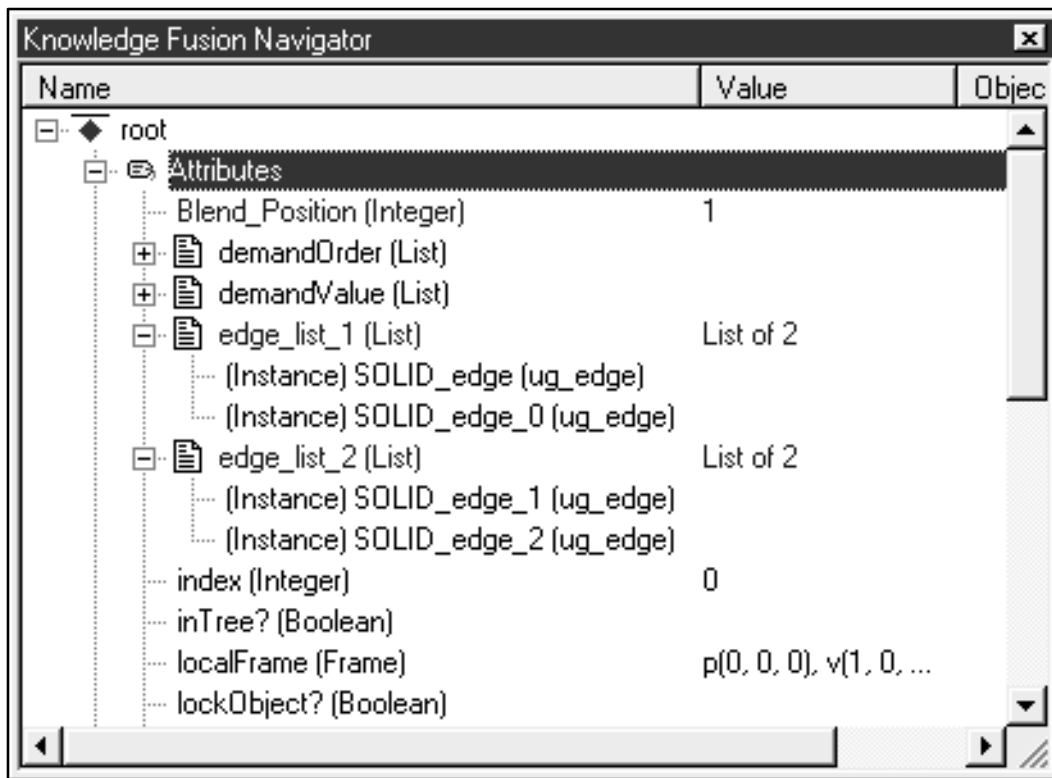
☐ In the Formula box, enter:

**{SOLID\_edge\_1;; SOLID\_edge\_2;}**

☐ Choose **OK**.

The KF Navigator should appear similar to the one shown in the following figure.





### Step 9 Change the Edge\_Blend\_References rule for BLEND\_4.

- ☐ From KF Navigator, double click **BLEND\_4**.
- ☐ Select **Edge\_Blend\_References:** from the Input Parameters box.
- ☐ Change the rule as following:  
 $\{nth(\text{Blend\_Position};, \text{edge\_list\_1:})\}$
- ☐ Choose **OK**.

### Step 10 Change the Edge\_Blend\_References rule for BLEND\_5.

- ☐ From KF Navigator, double click **BLEND\_5**.
- ☐ Select **Edge\_Blend\_References:** from the Input Parameters box.

- ☐ Change the rule as following:  
`{nth(Blend_Position:, edge_list_2:)}`
- ☐ Choose **OK**.

### Step 11 Test the design.

- ☐ Place the cursor over Blend\_Position and click MB3. Choose **Edit**. The Edit Attribute dialog displays.
- ☐ Change the value from 1 to 2 in the Dynamic Rule Formula box.
- ☐ Choose **OK**.

Notice the blend position change.

This concludes the activity.

## SUMMARY

In this lesson you:

- Used the adoptive approach to add rules to existing UG objects.
- Used the reference option to tie values together.



# User Defined Features

## Lesson 5

---

**PURPOSE**

To introduce User Defined Features in Knowledge Fusion.

**OBJECTIVES**

Upon completion of this lesson, you will be able to:

- Export User Defined Features.
- Position UDF by reference frame.
- Position UDF by reference geometry.
- Create custom dialog for UDF.
- Use KF rule to swap UDFs.

This lesson contains the following activities.

Activity	Page
5-1 Create a Knowledge Enabled Boss UDF . . . . .	5-5
5-2 Reference Frame Positioning of a UDF . . . . .	5-11
5-3 Reference Geometry Positioning of a UDF . . . . .	5-13
5-4 Customizing UDF Dialog . . . . .	5-20
5-5 Swapping User Defined Features . . . . .	5-28



## ***UDF Overview***

User Defined Features allow you to control and use features in Knowledge Fusion that are not currently supported by system classes. User Defined Features is one of the ways by which a user realizes re-use of Knowledge Fusion rules.

### *Knowledge Fusion rules in UDF*

When you export a User Defined Feature, all relevant Knowledge Fusion rules in the work part are saved with the UDF and thus become part of the UDF definition. The Knowledge Fusion rules that are brought into the receiving part when the UDF is instantiated also become available to the user.

### *UDF Application*

A UDF may be applied (instantiated) interactively two different ways:

- In Modeling, using Insert→Form Feature→User Defined Feature. (If the UDF contains Knowledge Fusion rules, it is automatically adopted into Knowledge Fusion.)
- In the Knowledge Fusion Navigation Tool, using “Add Child Rule” to create an object of the ug\_udfs class. KF provides the system class ug\_udfs that lets you instantiate a UDF. Input to this class is the UDF name, an optional library, and a list of parameters and references to be resolved. Additionally, there are also a number of strategies to choose from by which the UDF can be positioned in the receiving part.

## Exporting User Defined Features

All user defined features must be created and saved as a user defined feature file. The file can then be read in as a user defined feature in the Modeling application, and the data added as a feature to a target solid.

**NOTE**

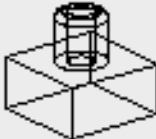
When you create a model for a UDF it is generally best if you define all features with Relative rather than Absolute definitions. For example, an EXTRUDE feature that is defined with an Absolute extrude direction of XC (X direction) will have problems updating when brought into a part in orientations other than the one in which it was created. On the other hand, an EXTRUDE feature created with a Relative extrude direction, such as a Relative Datum Axis, will update regardless of the orientation in which it is brought.

When you have finishing building a part that you want to save as a UDF, choose **File→Export→User Defined Feature**. The Select Feature for UDFS dialog then displays. Refer to the Unigraphics Modeling User Manual for additional information.

**Select Feature for UDFS**

UDF's name

Save To Library



Features In Part Features In UDF

Filter  ☐ Add Children Features

BLOCK(0)  
BOSS(1)  
SIMPLE\_HOLE(2)  
**BLEND(3)**

▶

◀

BOSS(1)  
SIMPLE\_HOLE(2)  
**BLEND(3)**

UDF References Prompts

New Prompt

Available Expressions

▶

◀

UDF Input Parameters

New Parameter Prompt

URL document location

## Activity 5–1: Create a Knowledge Enabled Boss UDF

In this activity, you will become familiar with creating knowledge enabled user defined features.

### Step 1 Export a user defined feature.

- ☐ If **\*\*\*\_boss.prt** is not open, retrieve this part.

**NOTE**

If you haven't completed the **\*\*\*\_boss.prt** in Lesson 4, you can copy the staged part **knf\_boss.prt** from the **parts\Staged\Lesson\_4** directory.

- ☐ Choose **File→Export→User Defined Feature**.
- ☐ In the UDF's name box, enter **boss1**.
- ☐ If you have set up a **kf\_UDF** library, from the Save to Library list, choose **kf\_UDF**.
- ☐ Click **Capture Image From Graphics Window**.
- ☐ Select all features in Features In Part list.
- ☐ Click the right arrow.
- ☐ Select **SIMPLE\_HOLE(2)** in Features In UDF list.
- ☐ Select **p11** in Available Expressions list.
- ☐ Click the right arrow.
- ☐ Select **p11 -> Diameter** in UDF Input Parameters list.

- ☐ In the New Parameter Prompt box, enter **Hole Diameter**.
- ☐ Choose **OK**.

**Step 2 Create a new part and enter the Modeling application.**

- ☐ Choose **File→New**. Enter **\*\*\*\_boss1** in the File name box. Accept **Inches** as the Units. Choose **OK**.
- ☐ From the graphics window, click MB3. Choose **Replace View→TFR–TRI**.
- ☐ Choose **View→Model Navigator**.
- ☐ Choose **Application→Modeling**.

**Step 3 Insert the UDF.**

- ☐ Choose **Insert→Form Feature→User Defined**.
- ☐ If you have set up a kf\_UDF library, from the Library Type list, choose **kf\_UDF**.
- ☐ Select **boss1**.
- ☐ Choose **OK**.
- ☐ Save the part.

**Step 4 Edit the UDF using Knowledge Fusion.**

- ☐ Choose **View→Knowledge Fusion Navigator**.
- ☐ Place the cursor over **boss1\_4** and click MB3. Choose **Edit**. The Edit Child Rule dialog displays.
- ☐ From the Input Parameters box, select **\*Parameters: {{Diameter, 0.5}};**.

- ☐ In the Rule for Parameter box, change the value for Diameter to any value from **0.1** to **1.0**.
- ☐ Choose **OK**.

**Step 5 Edit the UDF using feature parameter editing in Modeling.**

- ☐ Choose **Edit→Feature→Parameters**.
- ☐ Select **boss1(4)**.
- ☐ Choose **OK**.
- ☐ Change the value for Diameter to any value from **0.1** to **1.0**.
- ☐ Choose **OK**.
- ☐ Choose **OK**.

**Step 6 Edit the UDF using the Expression Subsystem.**

- ☐ Choose **Tools→Expression**.
- ☐ Select **Diameter\_4=...**
- ☐ Change the value for Diameter\_4 to any value from **0.1** to **1.0**.
- ☐ Choose **OK**.
- ☐ Close the part.

This concludes the activity.



## ***UDF Positioning Strategies***

### ***Reference Frames***

Positioning by reference frame assumes you are instantiating the UDF in a dfa file or through the Add Child Rule option in the Knowledge Fusion Navigation Tool. If the positioning geometry is part of the UDF definition, you can use reference frames to position your UDF. With this strategy, the positioning geometry is localized inside the UDF definition. Additionally, all the geometry inside the UDF is dependent directly on the reference frame that positions the UDF in the receiving part.

The advantage with this strategy is that the UDF itself may be created to the library, then positioned as though independent of the parent geometry. What is actually happening is that the parent geometry in the UDF is being made fully dependent on the reference frame.

Positioning by reference frames is not the “normal” or conventional way for UDF placement, however, it gives you some control similar to simple “mating conditions” (or may be used to do something similar to simple mating conditions if desired).

For example, let’s say we have a boss on the top face of a block. If we include both the boss and the block in the UDF definition, then we can position this UDF using the reference frame input parameter to the `ug_udfs` class. The resulting position and orientation of the UDF is a concatenation of the reference frame and the WCS of the UDF. So if the WCS of the UDF was at (1,2,3) and rotated about Z 30 degrees and you specified a reference frame at (1,2,3) with a Z rotation of 30 degrees, the final position of the block (if it was created at the WCS origin) would be (2,4,6) and rotated 60 degrees.

Steps to be taken when creating a UDF to position using reference frames would typically involve:

1. Create your base geometry using the absolute coordinate system.
2. Adopt the parent geometry into Knowledge Fusion (i.e., datums that control sketches).
3. Adopt any other geometry you are interested in controlling using rules.
4. At UDF creation time, select all the features you want in the UDF, plus the parent geometry.
5. At instantiation time, position by reference frame only.



## **Reference Geometry**

If the positioning geometry is not part of the UDF definition, you can use reference geometry to position your UDF. This method requires resolving geometric references before the UDF can be instantiated. Positioning of the UDF is normally or often dependent exclusively on the pre-existing geometry in the reference list. (Assuming the reference geometry is being used for positioning.)

This is the conventional way to position a UDF. One of the advantages of using this method is that the same process is used for positioning the UDF interactively in modeling, programmatically in a dfa file, or when adding a child rule in the Knowledge Fusion Navigator.

Steps to be taken when creating a UDF to position using reference geometry would typically involve:

1. Create your base geometry using the absolute coordinate system.
2. Adopt any geometry you are interested in controlling using rules.
3. At UDF creation time, select all the features you want in the UDF, excluding the parent geometry.
4. At instantiation time, position by selecting or referencing pre-existing geometry.

## Activity 5–2: Reference Frame Positioning of a UDF

In this activity, you will use the UDF from the previous activity, reposition the UDF by changing the reference frame.

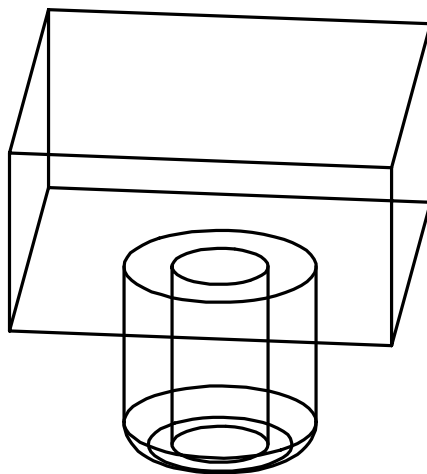
### Step 1 Retrieve **\*\*\*\_boss1.prt** and open the KF Navigator.

- ☐ If **\*\*\*\_boss1.prt** is not open, retrieve this part.
- ☐ Choose **View→Knowledge Fusion Navigator**.

### Step 2 Edit the position of the UDF.

- ☐ Place the cursor over **boss1\_4** and click MB3. Choose **Edit**. The Edit Child Rule dialog displays.
- ☐ From the Input Parameters box, select **\*referenceFrame: FrameXY(Point\_(0,0...**
- ☐ In the Rule for Parameter box, remove the current rule and enter:  
**FrameXY( point(1, 2, 3), vector(1, 1, 0), vector(1, –1, 0) );**
- ☐ Choose **OK**.

Notice the UDF position change as shown in the following figure.



☐ Close the part.

This concludes the activity.

### Activity 5–3: Reference Geometry Positioning of a UDF

In this activity, you will create a boss UDF that can be positioned using reference geometry. You will add this UDF to a part using the traditional modeling method, then you will add it to the part again using the KF method.

#### Step 1 Retrieve boss.prt and export a user defined feature.

- ☐ If **\*\*\*\_boss.prt** is not open, retrieve this part.
- ☐ Choose **File→Export→User Defined Feature**.
- ☐ In the UDF's Name box, enter **boss2**.
- ☐ If you have set up a **kf\_UDF** library, from the Save to Library list, choose **kf\_UDF**.
- ☐ Click **Capture Image From Graphics Window**.
- ☐ Select **BOSS(1)**, **SIMPLE\_HOLE(2)** and **BLEND(3)** features in Features In Part list.

**NOTE**

**BLOCK(0)** is not selected at this time. That means the positioning geometry is not part of the UDF definition, we have to use reference geometry to position this UDF.

- ☐ Click right arrow.
- ☐ Select **SIMPLE\_HOLE(2)** in Features In UDF list.
- ☐ Select **p11** in Available Expressions list.
- ☐ Click right arrow.
- ☐ Select **p11 -> Diameter** in UDF Input Parameters list.

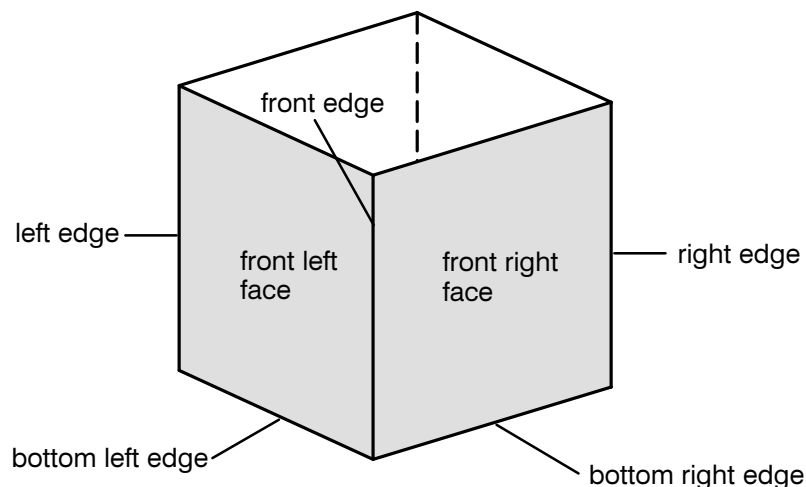
- ☐ In the New Parameter Prompt box, enter **Hole Diameter**.
- ☐ Choose **OK**.

**Step 2 Create a new part and create a block.**

- ☐ Choose **File**→**New**. Enter **\*\*\*\_boss2** in the File name box. Accept **Inches** as the Units. Choose **OK**.
- ☐ From the graphics window, click MB3. Choose **Replace View**→**TFR–TRI**.
- ☐ Choose **View**→**Model Navigator**.
- ☐ Choose **Application**→**Modeling**.
- ☐ Create a **Block 4 x 4 x 4**.

**Step 3 Insert the UDF using Modeling.**

- ☐ Choose **Insert**→**Form Feature**→**User Defined**.
- ☐ If you have set up a **kf\_UDF** library, from the Library Type list, choose **kf\_UDF**.
- ☐ Select **boss2**.
- ☐ Select the front right face of the block.



- ☐ Choose **OK**.
- ☐ Choose Horizontal dimension.
- ☐ Select front edge of the block.
- ☐ Select bottom right edge of the block.
- ☐ Enter **2.0** and Choose **OK**.
- ☐ Choose Vertical dimension.
- ☐ Select the right edge of the block.
- ☐ Enter **2.0** and Choose **OK**.

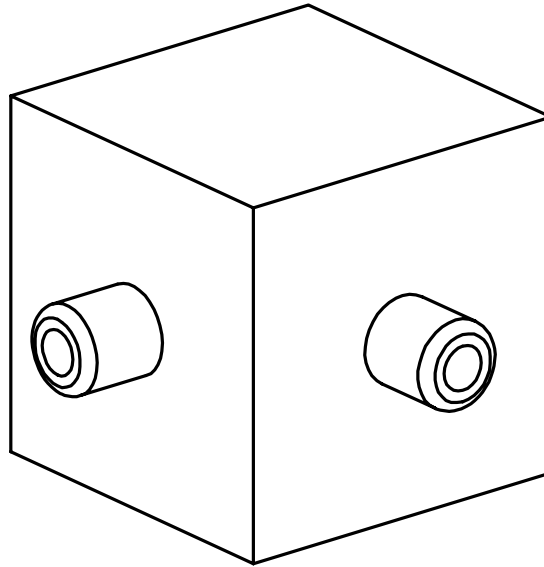
#### **Step 4 Insert the UDF using Knowledge Fusion.**

- ☐ Choose **View→Knowledge Fusion Navigator**.
- ☐ Place the cursor over **root** and click MB3. Choose **Add Child Rule**.
- ☐ In the Name box, enter **boss\_udf**.
- ☐ In the Class list select **ug\_udfs**.
- ☐ From the Input Parameters box, select **\*Name:<“”;>**.
- ☐ In the Rule for Parameter box, place the cursor between the quotes and enter **boss2**.
- ☐ From the Input Parameters box, select **\*Library:<“”;>**.
- ☐ In the Rule for Parameter box, place the cursor between the quotes and enter **kf\_UDF** if you have set up a kf\_UDF library.

- ☐ From the Input Parameters box, select **\*References:<{}>**.
- ☐ Place the cursor over white space in the KF Navigator and click MB3. Choose **Reference by Select**.
- ☐ Select the front left face of the block in the graphics window.
- ☐ Choose **OK** in the Class Selection dialog.
- ☐ From the Input Parameters box, select **\*usePositioningDialog?:<False>**.
- ☐ In the Rule for Parameter box, change **False** to **True**.
- ☐ Choose **OK**.
- ☐ Choose Horizontal dimension.
- ☐ Select left edge of the block.
- ☐ Select bottom left edge of the block.
- ☐ Enter **2.0** and Choose **OK**.
- ☐ Choose Vertical dimension.
- ☐ Select the front edge of the block.
- ☐ Enter **2.0** and Choose **OK**.

Your part should look like the following figure.





☐ Save the part.

This concludes the activity.



## Customizing the UDF Dialog

When instantiating a UDF using Modeling, a dialog is presented based on the input expressions and references required by the UDF definition. However, with the Knowledge Fusion class you can configure a customized dialog.

In Knowledge Fusion, each input parameter has a type. The types available are: Boolean, Integer, Number, Name, Point, String, Vector, List, Instance, User, and Any. For a UDF custom dialog, only the following types are supported: Boolean, Integer, Number, Name, Point, String, and Vector.

The input entry dialog for each parameter data type is listed in the following table.

KF Data Type	UG Entry dialog
Integer	Integer input box, slider, or option menu
Number	Number input box, slider, or option menu
Name	Name input box or option menu
String	String input box, option menu, or file selection box
Boolean	Toggle switch
Point	Point subfunction dialog
Vector	Vector subfunction dialog

### Input Parameters

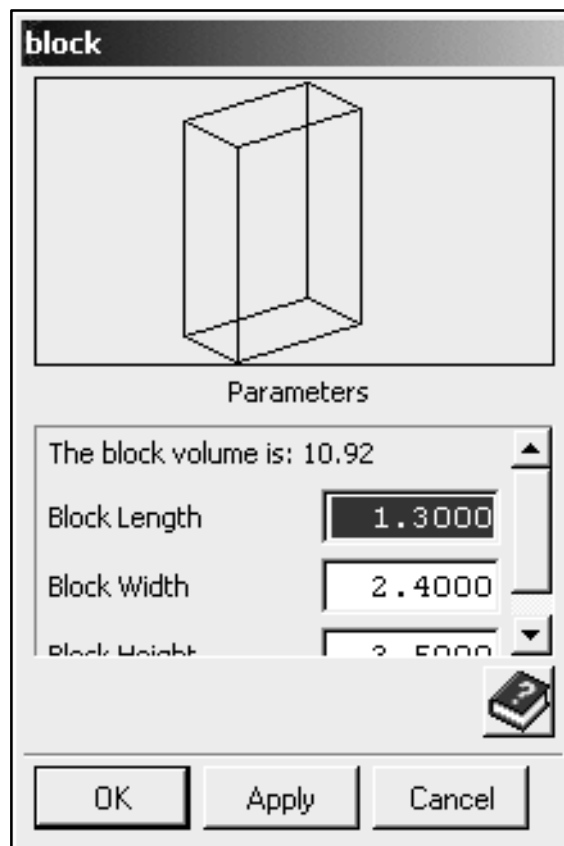
The UDF parameters are identified by a list attribute called `input_parameters`. It contains all the parameters that need to be added to the custom dialog. The order of the list is the order in which the parameters are presented in the dialog.

The parameter label for each UG dialog widget is specified by a string attribute, the name of which is the name of the parameter with the extension “\_label”. For example, if there is a parameter named “Integer:”, the label is defined by an attribute named “Integer\_label:”

## Output Parameters

Elements in the input\_parameters list that are of string type or are references to string attributes become output parameters in the UDF dialog. In the example below, the volume entry in the input\_parameters list would be an output parameter since it is an attribute of type string. Notice that length, width, and height in the input\_parameters list are names, and so these become input parameters in the UDF custom dialog.

```
(list) input_parameters: { Volume:, length, width, height};
(string) Volume: "The block volume is: " + format("%g",
               length*width*height);
(number modifiable) length: 1.3;
(string) length_label: "Block Length";
(number modifiable) width: 2.4;
(string) width_label: "Block Width";
(number modifiable) height: 3.5;
(string) height_label: "Block Height";
```



## Activity 5–4: Customizing UDF Dialog

In this activity, you will create a custom dialog for a boss UDF.

### Step 1 Create a new part and enter the Modeling application.

- ☐ Choose **File**→**New**. Enter **\*\*\*\_boss3** in the File name box. Accept **Inches** as the Units. Choose **OK**.
- ☐ From the graphics window, click MB3. Choose **Replace View**→**TFR–TRI**.
- ☐ Check to ensure **Model Navigator** and **Knowledge Fusion Navigator** are open.
- ☐ Choose **Application**→**Modeling**.

### Step 2 Create the geometry.

- ☐ Create a **Block 2 x 2 x 1**.
- ☐ Create a **Point** at **(1, 1, 1)**.
- ☐ Add a **Boss** centered on the Point just created.

Diameter:	<b>1</b>
Height:	<b>1</b>
Taper Angle:	<b>0</b>
- ☐ Add a **Hole** centered in the top face of the Boss.

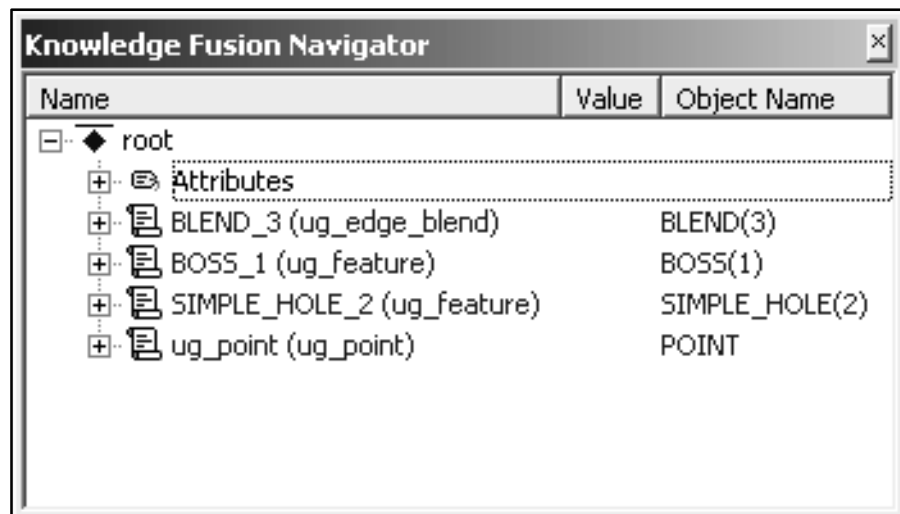
Diameter:	<b>0.5</b>
Depth:	<b>1</b>
Tip Angle:	<b>0</b>
- ☐ Add an **Edge Blend** to the top outside edge of the Boss.

Radius:	<b>0.125</b>
---------	--------------

**Step 3 Adopt the geometry.**

- ☐ Choose **Tools**→**Knowledge Fusion**→**Adopt Existing Object**, the Class Selection dialog displays.
- ☐ In the Model Navigator, press **Ctrl** key to select **BOSS(1)**, **SIMPLE\_HOLE(2)**, and **BLEND(3)**.
- ☐ Choose **OK** on the Class Selection dialog.
- ☐ Choose **Tools**→**Knowledge Fusion**→**Adopt Existing Object**, the Class Selection dialog displays.
- ☐ Select the point in the graphics window.
- ☐ Choose **OK** on the Class Selection dialog.

The KF Navigator should look like this:

**Step 4 Add UI customization.**

- ☐ Click the **+** sign on Attributes under the root node.
- ☐ Place the cursor over root and click MB3. Choose **Add Attribute**. The Add Attribute dialog displays.
- ☐ Enter **dia** in the Name box.

- ☐ From the Type list, choose **Number**.
- ☐ In the Formula box, enter **0.5**.
- ☐ Choose **Apply**.
- ☐ Enter **position** in the Name box.
- ☐ From the Type list, choose **Point**.
- ☐ In the Formula box, enter **point(1,1,1)**.
- ☐ Choose **Apply**.
- ☐ Enter **dia\_label** in the Name box.
- ☐ From the Type list, choose **String**.
- ☐ In the Formula box, enter **“Hole Diameter”**.
- ☐ Choose **Apply**.
- ☐ Enter **position\_label** in the Name box.
- ☐ From the Type list, choose **String**.
- ☐ In the Formula box, enter **“Boss Position”**.
- ☐ Choose **Apply**.
- ☐ Enter **input\_parameters** in the Name box.
- ☐ From the Type list, choose **List**.
- ☐ In the Formula box, enter **{dia, position}**.
- ☐ Choose **OK**.
- ☐ In the KF Navigator expand **SIMPLE\_HOLE\_2** by clicking the **+** sign.

- ☐ Place the cursor over **HOLE\_2 Diameter** and click MB3. Choose **Edit**. The Edit Child Rule dialog displays.
- ☐ From the Input Parameters box, select **\*value: 0.5;**.
- ☐ In the Rule for Parameter box, backspace and remove the **0.5**. Only the semicolon should remain.
- ☐ In the KF Navigator, place the cursor over the **dia** attribute under root. Click MB3. Choose **Reference**.
- ☐ Choose **OK**.
- ☐ Place the cursor over **ug\_point** and click MB3. Choose **Edit**. The Edit Child Rule dialog displays.
- ☐ From the Input Parameters box, select **\*position: Point(1,1,1);**.
- ☐ In the Rule for Parameter box, backspace and remove the rule. Only the semicolon should remain.
- ☐ In the KF Navigator, place the cursor over the **position** attribute under root. Click MB3. Choose **Reference**.
- ☐ Choose **OK**.
- ☐ Save the part.

#### **Step 5 Export a user defined feature.**

- ☐ Choose **File→Export→User Defined Feature**.
- ☐ In the UDF's Name box, enter **boss3**.
- ☐ If you have set up a **kf\_UDF** library, from the Save to Library list, choose **kf\_UDF**.
- ☐ Click **Capture Image From Graphics Window**.

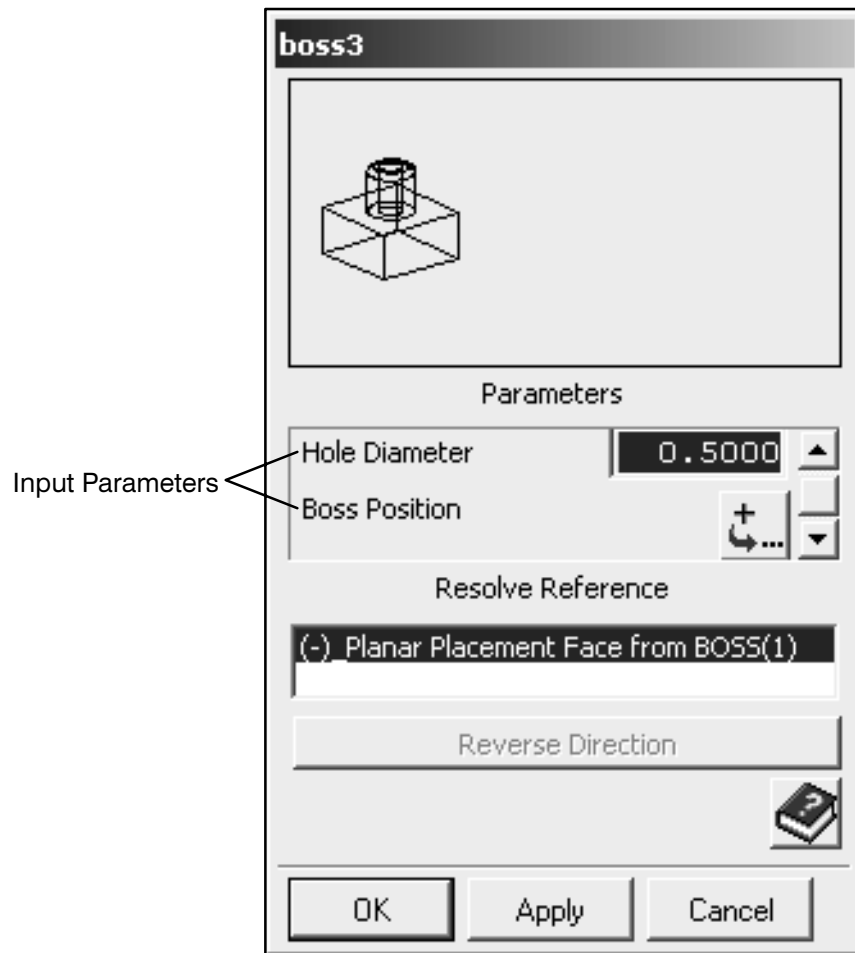
- ☐ Select **BOSS(1)**, **SIMPLE\_HOLE(2)** and **BLEND(3)** features in Features In Part list.
- ☐ Click right arrow.
- ☐ Choose **Add Geometry**. A selection dialog displays.
- ☐ Click **Type** under Filter Methods.
- ☐ Select **Point** and choose **OK**.
- ☐ Select the point you created in the graphics window.
- ☐ Choose **OK** in the selection dialog.
- ☐ Choose **OK** in the UDF dialog.

**Step 6 Create a new part and create a block and 2 points.**

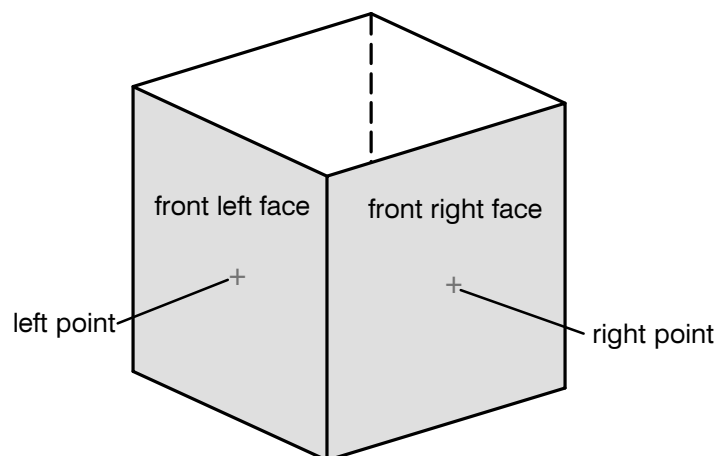
- ☐ Choose **File**→**New**. Enter **\*\*\*\_boss3\_dialog** in the File name box. Accept **Inches** as the Units. Choose **OK**.
- ☐ From the graphics window, click MB3. Choose **Replace View**→**TFR–TRI**.
- ☐ Create a **Block 4 x 4 x 4**.
- ☐ Create a **Point** at **(4, 2, 2)**.
- ☐ Create a **Point** at **(2, 0, 2)**.

**Step 7 Insert the UDF.**

- ☐ Choose **Insert**→**Form Feature**→**User Defined**.
- ☐ If you have set up a **kf\_UDF** library, from the Library Type list, choose **kf\_UDF**.
- ☐ Select **boss3**. The boss3 UDF custom dialog appears as shown in the following figure.



- ☐ Select the front right face of the block.

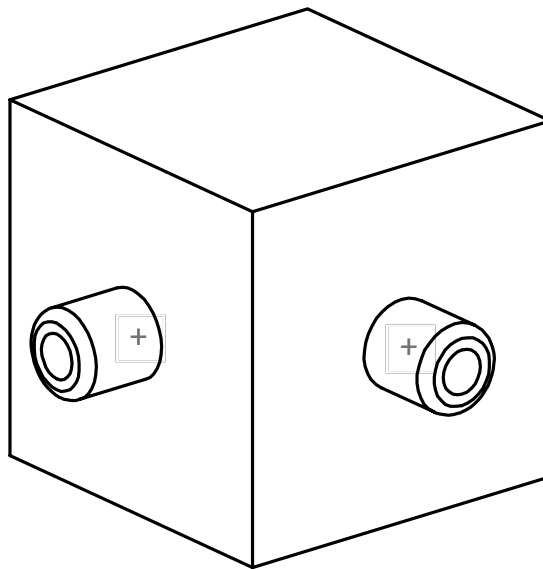


- ☐ Choose **Boss Position** button in the dialog.



- ☐ Select the right point in the graphics window.
- ☐ Choose **Apply** in the dialog.
- ☐ Select **Planar Placement Face from BOSS(1)** in the Resolve Reference box.
- ☐ Select the front left face of the block.
- ☐ Choose **Boss Position** button in the dialog.
- ☐ Select the left point in the graphics window.
- ☐ Choose **OK** in the dialog.

Your part should look like the following figure.



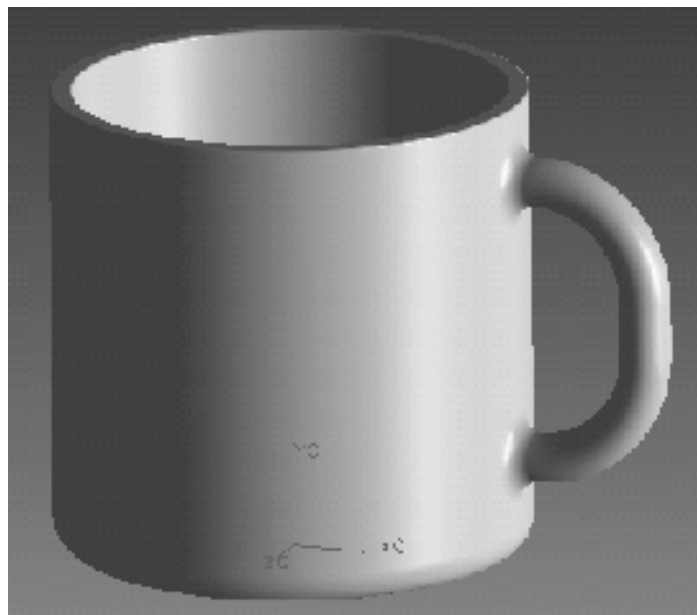
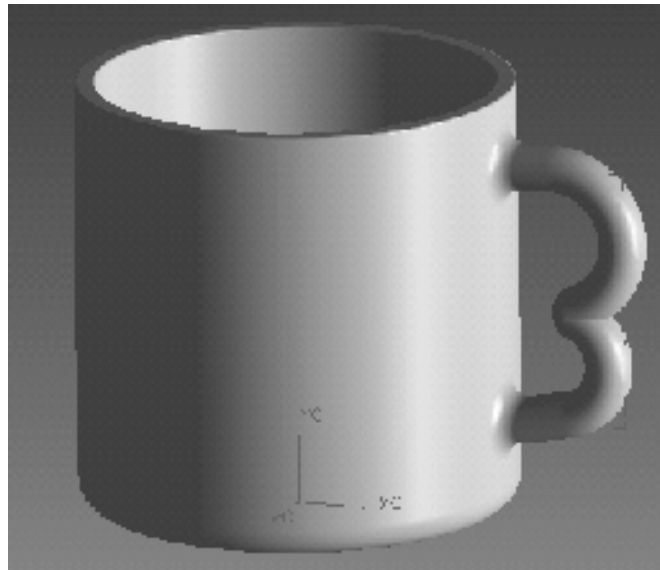
- ☐ Save the part.

This concludes the activity.

## Swap User Defined Features

You can swap User Defined Features using Knowledge Fusion rules as shown in the following example.

Name; if type: = 1 then “cup\_handle\_fancy” else “cup\_handle\_standard”;



## *Activity 5–5: Swapping User Defined Features*

In this activity, you will create two User Defined Features from two solid bodies consisting of several features. You will then programmatically swap between these two UDFs.

**Step 1** Open the part file **knf\_udf\_1.prt** and save as **\*\*\*\_udf\_1.prt**, where **\*\*\*** represents your initials.

**Step 2** Create the first User Defined Feature.

- ☐ Choose **File→Export→User Defined Feature**.
- ☐ In the UDF's name box, enter **udf1**.
- ☐ If you have set up a **kf\_UDF** library, from the Save to Library list, choose **kf\_UDF**.
- ☐ Capture the top view for the image.
- ☐ Select all features **except** **BLOCK(0)** in Features In Part list.
- ☐ Click the right arrow.
- ☐ Select **EXTRUDED(2)** in Features In UDF list.
- ☐ Select **depth** in Available Expressions list.
- ☐ Click the right arrow.
- ☐ Choose **OK**.

**Step 3** Open the part file knf\_udf\_2.prt and save as \*\*\*\_udf\_2.prt, where \*\*\* represents your initials.

**Step 4** Create the second User Defined Feature.

- ☐ Choose **File→Export→User Defined Feature**.
- ☐ In the UDF's name box, enter **udf2**.
- ☐ If you have set up a kf\_UDF library, from the Save to Library list, choose **kf\_UDF**.
- ☐ Capture the top view for the image.
- ☐ Select all features **except** BLOCK(0) in Features In Part list.
- ☐ Click the right arrow.
- ☐ Select **EXTRUDED(2)** in Features In UDF list.
- ☐ Select **depth** in Available Expressions list.
- ☐ Click the right arrow.
- ☐ Choose **OK**.

**Step 5** Create a part file to use the UDFs.

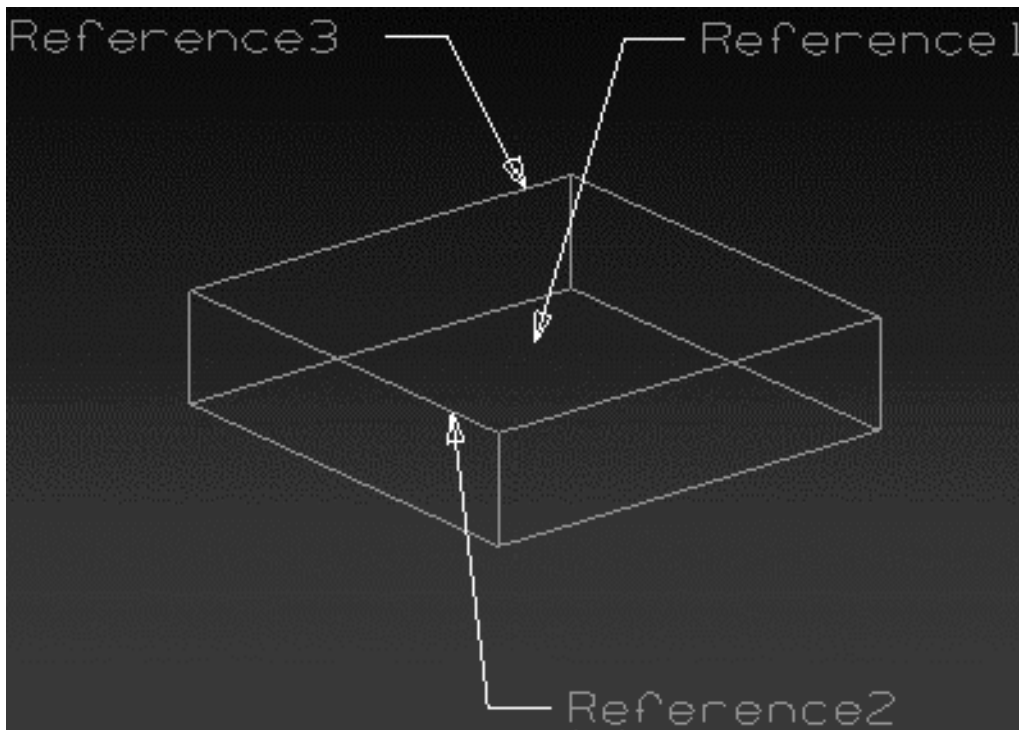
- ☐ Create a new part file, using **File→New**, and name it \*\*\*\_base.prt.
- ☐ Click MB3 from the graphics window. Choose **Replace View→TFR–TRI**.
- ☐ Choose **Application→Modeling**.

**Step 6 Create a block to form the base geometry that the UDF will be added to.**

- ☐ Choose **Insert**→**Form Feature**→**Block**.
- ☐ Set the dimensions of the block: **Length=2, Width=2, Height=0.5**.
- ☐ Accept the default location and click **OK** to create the block.

**Step 7 Insert the UDF.**

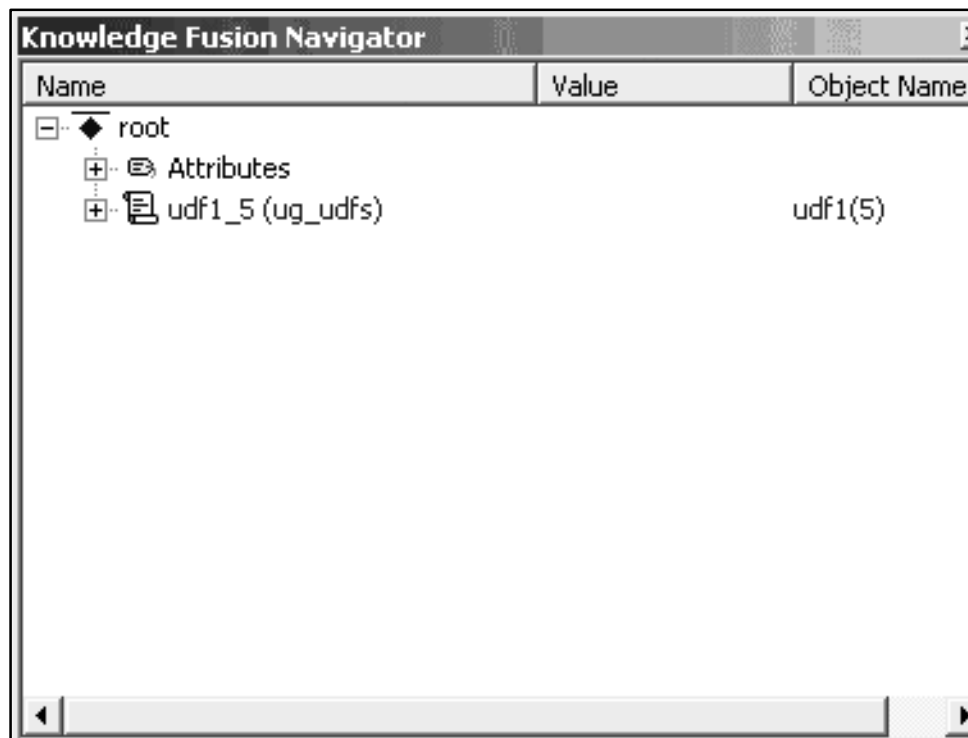
- ☐ Select **Insert**→**Form Feature**→**User Defined**.
- ☐ Select **udf1** to insert.
- ☐ Leave the default value for **depth**.
- ☐ Select the required references, see the following figure for selection. Pick the edges at the end that the arrows point.
- ☐ Choose **OK**.



**Step 8 Adopt the UDF into Knowledge Fusion.**

- ☐ Choose **View**→**Model Navigator**.
- ☐ Choose **View**→**Knowledge Fusion Navigator**.
- ☐ Select **Tools**→**Knowledge Fusion**→**Adopt Existing Objects**.
- ☐ Select the **udf1** feature from the Model Navigator.
- ☐ Choose **OK** in the Class Selection dialog.

The Knowledge Fusion Navigator should appear similar to the one shown in the figure below.

**Step 9 Create the Knowledge Fusion rules to swap the UDF.**

- ☐ In the KF Navigator, place the cursor over **root** and click MB3. Choose **Add Child Rule**.
- ☐ Enter in **Type** for the Name.

- ☐ Scroll down and select the **ug\_expression** class.
- ☐ Select **OK** to create the rule.

This class creates an expression with the expression name matching the rule name. We will use this expression to control the udf. If the value is 0, the udf1 will be used, if the value is 1, then udf2 will be used.

- ☐ In the KF Navigator, place the cursor over **udf1** and click MB3. Choose **Edit**.
- ☐ Select **Name** in the Input Parameters box. Enter in the following equation for the Name attribute:

**if Type:value:=0 then “udf1” else “udf2”;**

- ☐ Choose **OK**.

### Step 10 Swap the UDF.

- ☐ Select **Tools→Expression**.
- ☐ Change the value of **Type** from **0** to **1**.
- ☐ Choose **Apply**.

You should now see udf2 swap in place of udf1.

This concludes the activity.

**SUMMARY**

In this lesson you:

- Learned how to export User Defined Features.
- Repositioned UDF by changing the reference frame.
- Positioned UDF using reference geometry.
- Created custom dialog for UDF.
- Used KF rule to swap UDFs.





**(This Page Intentionally Left Blank)**

# Assemblies

## Lesson 6

6

**PURPOSE**

To introduce the UG/KF Assemblies concepts.

**OBJECTIVES**

Upon completion of this lesson, you will be able to:

- Create an assembly using Knowledge Fusion.
- Use the class `ug_child_in_part`.
- Use the class `ug_component`.

This lesson contains the following activity.

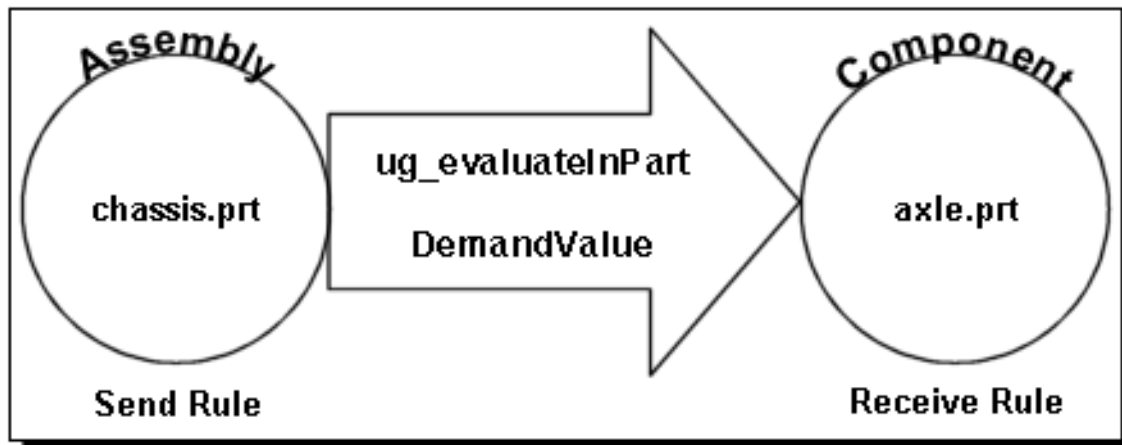
Activity	Page
6–1 Create a Simple Chassis Assembly . . . . .	6–4

## UG/KF Assemblies Concepts

6

### *Assemblies and the Knowledge Fusion World*

UG/KF Assemblies allows you to create an assembly with both its structure and components under the control of KF rules. Each Unigraphics part file has its own KF world bound to it. The KF world maintains the rules and relationships that binds the rules to UG objects. When you enter the KF Navigator you see the KF world, its rules and attributes, for the work part only. With respect to assembly modeling, there does not exist a single KF world with rules that cross part boundaries. For example, you cannot create a rule whose evaluation resides in another part. However, the same effect is possible with a messaging scheme that allows the KF worlds to communicate with each other.



KF uses the function `ug_evaluateInPart` to pass messages between the KF worlds in different part files. This function sends a message (transmits a rule) to another part in order to create a rule or to edit that rule if it already exists. Both assembly structure and component content can be specified and controlled with this function. The general format of this function is:

(String) variable: `ug_evaluateInpart (String Part_Name, String Rule_Name, String Rule_Text);`

For example to create geometry in another part:

```
ug_evaluateInpart ("component1.prt", "Pt1", "{class; ug_point, position,
point(1,2,3)}");
```

*Evaluating a rule in another part*

The class `ug_child_in_part` has been provided as an alternative to `ug_evaluateInPart`. This is generally much easier to use. The class `ug_child_in_part` has a syntax similar to the (Child) rule. Here is an example of using `ug_child_in_part`:

- `Target_file_name`           “comp.prt”
- `Parameters`                {class, ug\_block, origin, point(1,2,3), ...}

By using the function `ug_child_in_part`, we can execute a child rule in another part. It also supplies the means by which part files can communicate with each other. Messages will be sent to another part in order to create a rule or edit that rule if it already exists. The part must already be loaded or locatable using current load options.

*Reading a KF attribute in another part*

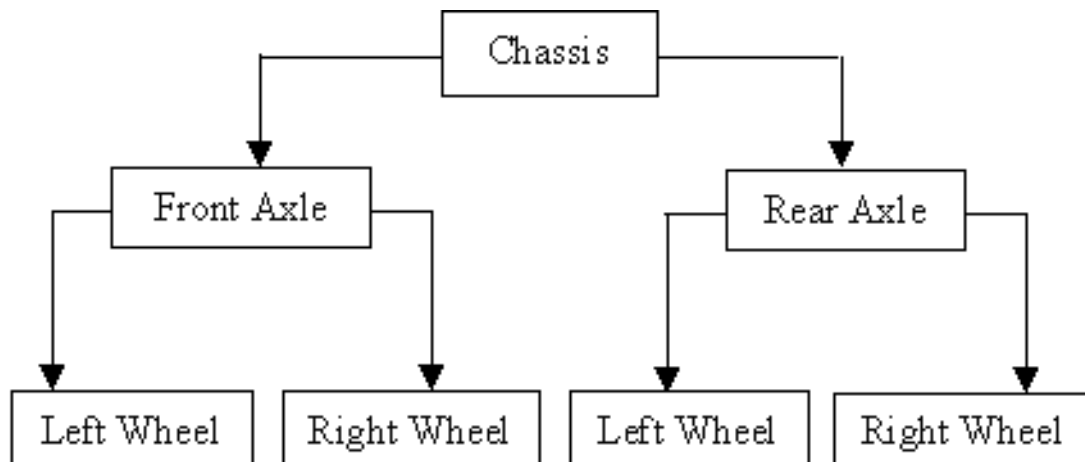
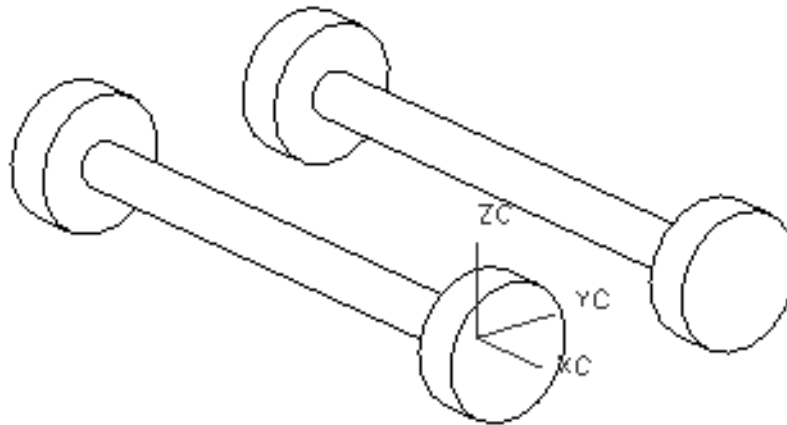
Existing attribute values can be read from a component (or any other part) using the function `ug_askKFattrValue`. However, `ug_askKFattrValue` cannot be used to read the value of an attribute whose value might be changing as a result of `ug_evaluateInPart`. Here is an example of fetching the value of an attribute of a rule in another part:

```
(any) my_attribute_value: ug_askKFattrValue(String Part_Name,
String Rule_Name);
```

```
(any) color: ug_askKFattrValue("target_part", "body:color");
```

## Activity 6–1: Create a Simple Chassis Assembly

In this activity, you will create a simple chassis assembly as shown below using Knowledge Fusion.



**Step 1 Create new parts and open the KF Navigator.**

- ☐ Choose **File→New**. Enter **\*\*\*\_wheel** in the File name box. Accept **Inches** as Units. Choose **OK**.
- ☐ Choose **File→New**. Enter **\*\*\*\_axle** in the File name box. Accept **Inches** as Units. Choose **OK**.
- ☐ Choose **File→New**. Enter **\*\*\*\_chassis** in the File name box. Accept **Inches** as Units. Choose **OK**.
- ☐ From the graphics window, click MB3. Choose **Replace View→TFR–TRI**.
- ☐ Choose **View→Knowledge Fusion Navigator**.

**Step 2 In \*\*\*\_chassis.prt, instantiate a cylinder for \*\*\*\_wheel.prt.****NOTE**

Be careful not to switch to \*\*\*\_wheel.prt, we will be working within \*\*\*\_chassis.prt through the whole activity.

- ☐ Place the cursor over **root** and click MB3. Choose **Add Child Rule**.
- ☐ In the Name box, enter **wheel**.
- ☐ In the Class list select **ug\_child\_in\_part**.
- ☐ From the Input Parameters box, select **\*Target\_File\_Name:<“”;>**.
- ☐ In the Rule for Parameter box, place the cursor between the quotes and enter **\*\*\*\_wheel.prt**.
- ☐ From the Input Parameters box, select **\*Parameters:<{};>**.

- ☐ In the Rule for Parameter box, place the cursor between the braces and enter **class, ug\_cylinder, diameter, 3.0**.
- ☐ Choose **OK**.

**NOTE**

In this step, although you were working in **\*\*\*\_chassis.prt**, you actually instantiated a cylinder in **\*\*\*\_wheel.prt** by using the **ug\_child\_in\_part** class. You will not see anything in the graphics area. But if you switch to **\*\*\*\_wheel.prt**, you will see the cylinder that was instantiated. Be sure to switch back to **\*\*\*\_chassis.prt** to continue the next step if you have switched to **\*\*\*\_wheel.prt**.

**Step 3 In \*\*\*\_chassis.prt, instantiate a cylinder for \*\*\*\_axle.prt.**

- ☐ Place the cursor over **root** and click MB3. Choose **Add Child Rule**.
- ☐ In the Name box, enter **axle**.
- ☐ In the Class list select **ug\_child\_in\_part**.
- ☐ From the Input Parameters box, select **\*Target\_File\_Name:<“”;>**.
- ☐ In the Rule for Parameter box, place the cursor between the quotes and enter **\*\*\*\_axle.prt**.
- ☐ From the Input Parameters box, select **\*Parameters:<{};>**.
- ☐ In the Rule for Parameter box, place the cursor between the braces and enter **class, ug\_cylinder, height, 12.0**.
- ☐ Choose **OK**.

NOTE: Same as explained above, you will not see anything in the graphics area of **\*\*\*\_chassis.prt**. You can view the cylinder in the **\*\*\*\_axle.prt**. Be sure to switch back to **\*\*\*\_chassis.prt** to continue the next step.

**Step 4** In **\*\*\*\_chassis.prt**, instantiate two wheel components for **\*\*\*\_axle.prt**.

- ☐ Place the cursor over **root** and click MB3. Choose **Add Child Rule**.
- ☐ In the Name box, enter **right\_wheel**.
- ☐ In the Class list select **ug\_child\_in\_part**.
- ☐ From the Input Parameters box, select **\*Target\_File\_Name:<“”;>**.
- ☐ In the Rule for Parameter box, place the cursor between the quotes and enter **\*\*\*\_axle.prt**.
- ☐ From the Input Parameters box, select **\*Parameters:<{};>**.
- ☐ In the Rule for Parameter box, place the cursor between the braces and enter **class, ug\_component, file\_name, “\*\*\*\_wheel.prt”, origin, point(0, 0, -1)**.
- ☐ Choose **Apply**.
- ☐ In the Name box, enter **left\_wheel**.
- ☐ In the Class list select **ug\_child\_in\_part**.
- ☐ From the Input Parameters box, select **\*Target\_File\_Name:<“”;>**.
- ☐ In the Rule for Parameter box, place the cursor between the quotes and enter **\*\*\*\_axle.prt**.
- ☐ From the Input Parameters box, select **\*Parameters:<{};>**.
- ☐ In the Rule for Parameter box, place the cursor between the braces and enter **class, ug\_component, file\_name, “\*\*\*\_wheel.prt”, origin, point(0,0,12)**.
- ☐ Choose **OK**.

NOTE: You can switch to **\*\*\*\_axle.prt** to view the updated graphics. Be sure to switch back to **\*\*\*\_chassis.prt** to continue the next step.

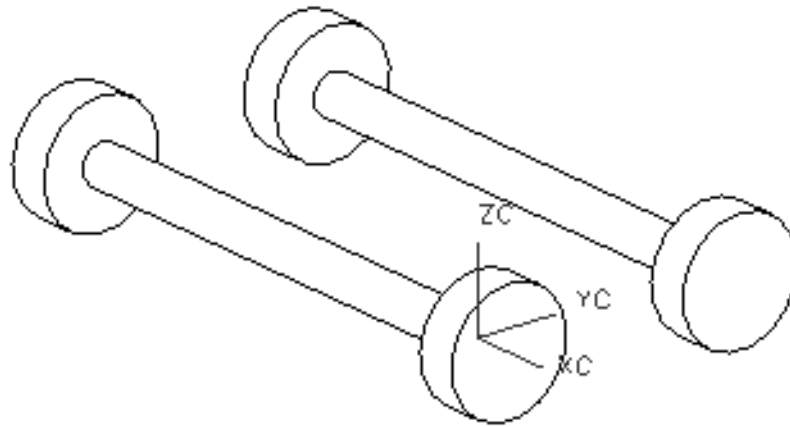


**Step 5** In **\*\*\*\_chassis.prt**, instantiate two axle components.

- ☐ Place the cursor over **root** and click MB3. Choose **Add Child Rule**.
- ☐ In the Name box, enter **rear\_axle**.
- ☐ In the Class list select **ug\_component**.
- ☐ From the Input Parameters box, select **\*File\_Name:<">**.
- ☐ In the Rule for Parameter box, place the cursor between the quotes and enter **\*\*\*\_axle.prt**.
- ☐ From the Input Parameters box, select **\*X\_axis: <Vector(1,0,0)>**.
- ☐ In the Rule for Parameter box, change **Vector(1,0,0)** to **Vector(0,-1,0)**.
- ☐ From the Input Parameters box, select **\*Y\_axis: <Vector(0,1,0)>**.
- ☐ In the Rule for Parameter box, change **Vector(0,1,0)** to **Vector(0,0,1)**.
- ☐ Choose **Apply**.
- ☐ In the Name box, enter **front\_axle**.
- ☐ In the Class list select **ug\_component**.
- ☐ From the Input Parameters box, select **\*File\_Name:<">**.
- ☐ In the Rule for Parameter box, place the cursor between the quotes and enter **\*\*\*\_axle.prt**.
- ☐ From the Input Parameters box, select **\*Origin:<Point(0,0,0)>**.
- ☐ In the Rule for Parameter box, change **Point(0,0,0)** to **Point(0,6,0)**.

- ☐ From the Input Parameters box, select  
\*X\_axis:<Vector(1,0,0);>.
- ☐ In the Rule for Parameter box, change **Vector(1,0,0)** to  
**Vector(0,-1,0)**.
- ☐ From the Input Parameters box, select  
\*Y\_axis:<Vector(0,1,0);>.
- ☐ In the Rule for Parameter box, change **Vector(0,1,0)** to  
**Vector(0,0,1)**.
- ☐ Choose **OK**.

Your \*\*\*\_chassis.prt will now look like this:



#### Step 6 Display the Assembly Navigator.

- ☐ Choose **View→Assembly Navigator**.
- ☐ Save the parts.

This concludes the activity.

## SUMMARY

In this lesson you:

- Created a chassis assembly using Knowledge Fusion.
- Used the class `ug_child_in_part`.
- Used the class `ug_component`.



Optimization  
Lesson 7

PURPOSE

To introduce the optimization concepts in Knowledge Fusion.

OBJECTIVES

Upon completion of this lesson, you will be able to:

- Use `ug_optimize` class to run optimization in Knowledge Fusion.

This lesson contains the following activity.

Activity	Page
7-1 Optimize the Volume of a Squeeze Bottle . . . . .	7-5



## Optimization Concepts

Optimization is an automatic method of improving your product based on engineering constraints. It's an iterative method to find the best configuration possible by varying parameters to meet a particular objective without violating constraints. Unigraphics uses a third party optimizer from Altair Engineering called HyperOpt.

During the optimization process, the system attempts to drive the output of the model toward a given goal by iterating the values of one or more Design Variables and avoiding violation of given constraints.

7

An optimization problem consists of the following attributes:

- **Objective:** This describes what you're trying to achieve. The objective consists of the Objective Rule to be measured at each iteration and the type of goal to be reached. The objective rule is a KF reference chain to a number attribute. You may choose to Minimize the objective rule, Maximize the objective rule, or drive the object rule toward a target value.
- **Design Variables:** This identifies the parameters to be varied during the iteration process. One or more design variables can be specified. They should be chosen so that varying their values makes a difference in the objective or constraint measurements. Each design variable consists of the name of the KF attribute to be modified and reasonable upper and lower limits of its value.

NOTE: since the optimization is an iterative solution, adding more design variables increases the number of iterations required to achieve convergence.

- **Constraints:** This describes other limits of the model other than design variable limits. You can specify zero or more constraints. At each iteration, the optimizer compares the value of each constraint attribute against its limit. If a constraint value falls outside of its limit, the model is considered to be in an invalid state. The optimizer backs up to a valid state and tries different values for the design variables. Each constraint consists of a reference chain to a number attribute, the type of constraint (minimum or maximum), and the limit value.
- **Convergence Criteria:** This tells the optimizer how accurate the solution needs to be. It consists of three numbers that the optimizer uses to determine if it is done improving the model.
  - **Relative:** If one minus the ratio of the last 2 objective values is less than this value, the solution is converged. A smaller value here most likely increases the number of iterations required.
  - **Absolute:** This value is multiplied by the first objective result, and if the difference in the last 2 objective results is less than this, then it is converged. A smaller value here most likely increases the number of iterations required.
  - **Maximum number of iterations:** Limit on the number of iterations.

NOTE: The smaller the convergence criteria values, the more iterations it may take to converge.



## General Optimization Process

### How it works

1. UG model updates Design Variable values.
2. Constraints and Objective are calculated based on UG model.
3. Constraints and Objective are sent to HyperOpt optimizer.
4. If not converged, optimizer returns new Design Variable values for next iteration.

7

### Example

Here is an example of using ug\_optimize class:

```
(child) my_opt: {  
    class; ug_optimize;  
    objective; { SOLID_RIGHT:mass:, TARGET, weight: };  
    design_variables; { { pocket_depth:, Value,2.0,20.0 }, ... };  
    constraints; { { SOLID_RIGHT:volume:, Lower, 5000.0 }, ... };  
    optimizer_controls; { 0.05, 0.005, 0.025, 20 };  
    view_graph; false;  
};
```

## Activity 7–1: Optimize the Volume of a Squeeze Bottle

In this activity, you will adopt the geometry of a squeeze bottle into Knowledge Fusion. Then you will use `ug_optimize` class to optimize its volume to a target value.



### Step 1 Open the part file and enter the Modeling application.

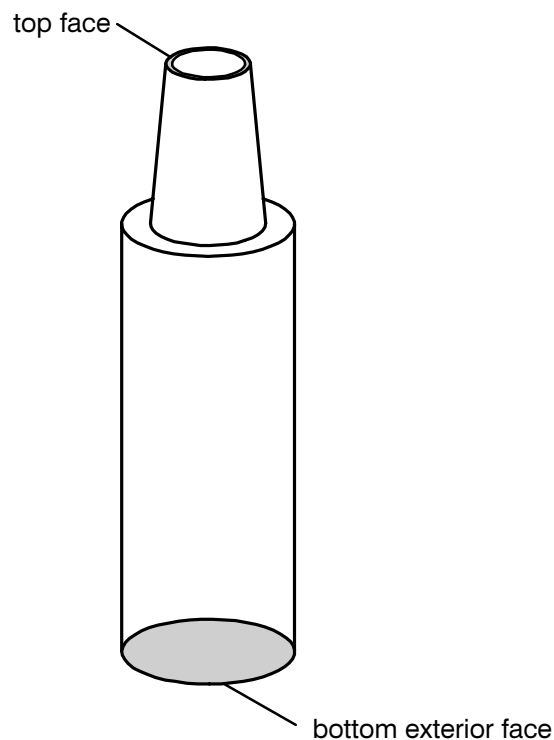
- ☐ In your **parts** directory, open the part file **knf\_bottle.prt**.
- ☐ Save the file as **\*\*\*\_bottle.prt** at your working directory.
- ☐ Choose **View→Knowledge Fusion Navigator**.
- ☐ Choose **View→Model Navigator**.
- ☐ Choose **Application→Modeling**.

### Step 2 Adopt the geometry.

- ☐ Choose **Tools→Knowledge Fusion→Adopt Existing Object**.
- ☐ In the Model Navigator, press **Ctrl** to select **CYLINDER(0)** and **BOSS(1)**.



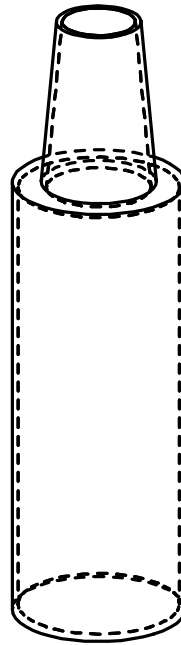
- ☐ Choose **OK** on the Class Selection dialog.
- ☐ Choose **Tools**→**Knowledge Fusion**→**Adopt Existing Object**.
- ☐ Choose **Type** in the Class Selection dialog.
- ☐ Select **Face** and choose **OK**.
- ☐ Select the top face and the bottom exterior face of bottle as shown in the following figure.



- ☐ Choose **Type** in the Class Selection dialog.
- ☐ Select **Solid Body** and choose **OK**.
- ☐ Select the interior solid body contained within the dashed line as shown in the following figure.

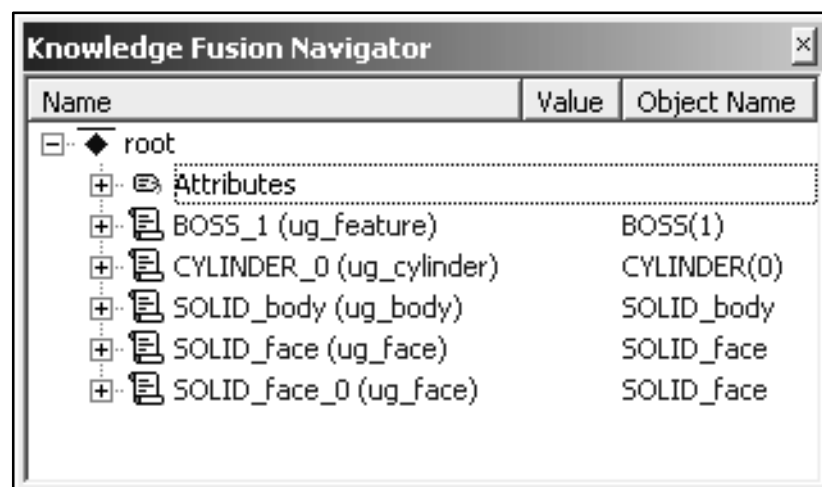
**NOTE**

Be sure to select the solid body (not the feature), watch the cue line when making the selection.



- ☐ Choose **OK** on the Class Selection dialog.

The KF Navigator should look like this:



### Step 3 Setup the optimization.

- ☐ Place the cursor over **root** and click MB3. Choose **Add Child Rule**.

- ☐ In the Name box, enter **bottle\_opt**.
- ☐ In the Class list select **ug\_optimize**.
- ☐ From the Input Parameters box, select **\*Design\_Variables:<{};>**.
- ☐ In the Rule for Parameter box, type the following rule, the final enter should look like:

```
{
{CYLINDER_0:, diameter, 1.5, 5.5},
{CYLINDER_0:, height, 6.5, 10.5},
{BOSS_1:Diameter:, value, 0.5, 4.5},
{BOSS_1:Height:, value, 0.4, 4.5},
{BOSS_1:Taper_Angle:, value, 2.5, 7.5}
};
```

- ☐ From the Input Parameters box, select **\*Constraints:<{};>**.
- ☐ In the Rule for Parameter box, type the following rule, the final enter should look like:

```
{ {first(ug_askMinimumDistance(SOLID_face:,
SOLID_face_0:)), 11.5, Upper} };
```

- ☐ From the Input Parameters box, select **\*Objective:<{};>**.
- ☐ In the Rule for Parameter box, type the following rule, the final enter should look like:

```
{SOLID_body:Volume:, Target, 60.0};
```

- ☐ Choose **OK**.
- ☐ Save the part.



**Step 4 Run the optimization.**

- ☐ In the KF Navigator expand **bottle\_opt** by clicking the + sign.
- ☐ Expand the bottle\_opt **Attributes** by clicking the + sign.
- ☐ Place the cursor over **Go\_Optimize** and click MB3. Choose **Show Value**. The optimization is executed.
- ☐ Close the Microsoft Excel graph window.
- ☐ In the KF Navigator expand **SOLID\_body** by clicking the + sign.
- ☐ Expand SOLID\_body **Attributes** by clicking the + sign.
- ☐ Place the cursor over **Volume** and click MB3. Choose **Show Value**. The value should be very close to the target value of 60.0.

**Step 5 Change the target value and re-run the optimization.**

- ☐ Place the cursor over **bottle\_opt** and click MB3. Choose **Edit**. The Edit Child Rule dialog displays.
- ☐ From the Input Parameters box, select **\*Objective:{SOLID\_body:Volume...**
- ☐ In the Rule for Parameter box, change **60.0** to **120.0**.
- ☐ From the Input Parameters box, select **\*View\_Graph:<true;>**.
- ☐ In the Rule for Parameter box, change **true** to **false**.
- ☐ Choose **OK**.
- ☐ Place the cursor over **Go\_Optimize** and click MB3. Choose **Show Value**. The optimization is executed.

NOTE: This time you will not see the Microsoft Excel graph window running to show the optimizing process because the View\_Graph option was turned off.

- ☐ Place the cursor over **Volume** attribute of SOLID\_body and click MB3. Choose **Show Value**. The value should be very close to the new target value of 120.0.
- ☐ Close the part.

This concludes the activity.



**SUMMARY**

In this lesson you:

- Used `ug_optimize` class to run optimization in Knowledge Fusion.

**7**



**(This Page Intentionally Left Blank)**

# Spreadsheet Access

## Appendix A



### PURPOSE

To introduce the spreadsheet concepts in Knowledge Fusion.

### OBJECTIVES

Upon completion of this lesson, you will be able to:


- Use ug\_spreadsheet class to access the spreadsheet data.

This lesson contains the following activity.

Activity	Page
A–1 Use Spreadsheet to Control Block Parameters . .	A–6



## Spreadsheet Concepts

 Knowledge Fusion has connectivity to spreadsheets stored in part files allowing access to spreadsheet data from MS/Excel or Xess. This functionality works for spreadsheet data stored in part files only. You can access numbers, strings and formulae stored in the spreadsheet, which can reside in the current part file or any other part file. You can access the spreadsheet data that is stored in either the Gateway or Modeling application.

The Knowledge Fusion class, `ug_spreadsheet`, provides an easy-to-use mechanism to read and write data into spreadsheets. You must instantiate a spreadsheet in Knowledge Fusion to gain access to the data, which can then be used in your designs. Once a `ug_spreadsheet` class is instantiated you can execute the methods to read and write data to the spreadsheet. Data contained in Part Families spreadsheet can also be read, using this class, but you cannot write or modify the data. For more information on Part Families in Knowledge Fusion, refer to `ug_part_family`.

Please note the following items:

- If you are accessing a spreadsheet contained in a part other than the current part then the name of the part specified in the parameter `part_file`: must be valid and this part file must be loaded in the current UG session. The “.prt” extension to the file name is optional.
- Data contained in the spreadsheet is modifiable only by using the methods available in the spreadsheet class. You can save the spreadsheet data from Knowledge Fusion. However, if the part is not the current part then you must save the part file separately. Knowledge Fusion does not automatically save the part file while saving the spreadsheet data.
- Knowledge Fusion launches the spreadsheet application automatically, but you should not attempt to modify the contents interactively. Such updates are not saved.

- Since a spreadsheet can contain blank lines, cells, and sheets you cannot determine the size of the data like total number of rows or columns. We suggest the use of Named Ranges of tables to access this information. A Named Range can be used on multiple rows, columns, and sheets to specify the cells of interest.
- You will face current limitations while accessing spreadsheet data in part files saved in Excel or Xess, while working across native Windows and UNIX environments.
- If you are creating User Defined Features from objects that refer to spreadsheets then you must either enter the full name of the base part before you export the User Defined Features or you must edit the User Defined Feature part file's spreadsheet parameter to contain its name explicitly defined. For example if the exported User Defined Feature part file name is sa0001234.prt then you must edit the part file parameter of the spreadsheet in this part to be "sa0001234".



## ***The ug\_spreadsheet Class***

The `ug_spreadsheet` class provides a connection to a spreadsheet. You can then query or add to the data in the spreadsheet. You must have the spreadsheet application available on your system to access the data.

Knowledge Fusion opens the part file (if it is not the current part) and connects to the spreadsheet when it is first instantiated, usually when the `ug_spreadsheet` object is created in the part or during part update. Knowledge Fusion does not close the part or the spreadsheet unless you explicitly call `File→Close` and choose the part. The referring part file need not be open while instantiating the class or during update. Knowledge Fusion automatically loads the part file and connects to the spreadsheet. If you have written any data into the spreadsheet, you should explicitly save the part file that contains this spreadsheet. Knowledge Fusion does not do this for you.

### ***The ug\_spreadsheet Class Definition***

This class provides access to spreadsheets from Knowledge Fusion by supporting both creation and loading of spreadsheets that already exist in the part. These will be either Gateway or Modeling spreadsheets. For a Part Family spreadsheet, only reading the spreadsheet is supported.

On UNIX machines, this class connects to XESS spreadsheets and on Windows NT, it connects to Excel. A cell is always identified by its row, column and sheet indexes.

- Input parameters:
  - **spreadsheet\_name:** either GATEWAY, MODELING, or PART\_FAMILY (GATEWAY is the default).
  - **part\_file:** which part this spreadsheet is associated with (default is “”, which indicates current part).
- Operations supported:
  - ask/set current worksheet
  - ask the number of sheets in the spreadsheet
  - insert/delete specified number of sheets from the spreadsheet
  - ask/set a formula in the specified cell
  - ask/set a numeric value in the specified cell
  - ask/set a string value in the specified cell
  - retrieve a number, string or formula, depending on the contents, from the specified cell
  - retrieve the evaluated value of the specified cell
  - erase a range of cells in the spreadsheet
  - clear the contents of the entire spreadsheet
  - delete an entire row/column
  - save the spreadsheet in the part file



## Activity A–1: Use Spreadsheet to Control Block Parameters

A

In this activity, you will create a spreadsheet in a part file. Then you will instantiate a `ug_spreadsheet` class to access the data stored in the spreadsheet to control block parameters.

### Step 1 Create a new part and create a spreadsheet.

- ☐ Choose **File**→**New**. Enter **\*\*\*\_spreadsheet** in the File name box. Accept **Inches** as the Units. Choose **OK**.
- ☐ Choose **Application**→**Gateway...** if Unigraphics is not in the Gateway application.
- ☐ Choose **Tools**→**Spreadsheet**.
- ☐ Enter the following data:

2	2
4	4
2	6
6	2
6	6
1	6

- ☐ Save and exit the Excel spreadsheet.
- ☐ Save the part.

### Step 2 Create a new part and open the KF Navigator.

- ☐ Choose **File**→**New**. Enter **\*\*\*\_block** in the File name box. Accept **Inches** as the Units. Choose **OK**.
- ☐ From the graphics window, click MB3. Choose **Replace View**→**TFR – TRI**.

- ☐ Choose **View→Knowledge Fusion Navigator**. Click the **+** sign on Attributes under the root node.



### Step 3 Add a row attribute.

- ☐ Place the cursor over **root** and click MB3. Choose **Add Attribute**. The Add Attribute dialog displays.
- ☐ Enter **row** in the Name box.
- ☐ From the Type list, choose **Integer**.
- ☐ In the Formula box, enter **1**.
- ☐ Choose **OK**.

### Step 4 Add the spreadsheet child rule.

- ☐ Place the cursor over **root** and click MB3. Choose **Add Child Rule**.
- ☐ In the Name Box, enter **ss**.
- ☐ From the Class List box, select **ug\_spreadsheet**.
- ☐ From the Input Parameters box, select **\*part\_file:<"">**.
- ☐ In the Rule for Parameter box, place the cursor between the quotes and enter **\*\*\*\_spreadsheet.prt**.
- ☐ Choose **Apply**.

### Step 5 Add the block child rule.

- ☐ In the Name Box, enter **block**.
- ☐ From the Class List box, select **ug\_block**.
- ☐ From the Input Parameters box, select **\*Length:<1>**.



- ☐ Backspace over the **1** so that only the semicolon displays.
- ☐ Enter **ss:ask\_number:(row:, 1)**.
- ☐ From the Input Parameters box, select **\*Width:<1;>**.
- ☐ Backspace over the **1** so that only the semicolon displays.
- ☐ Enter **ss:ask\_number:(row:, 2)**.
- ☐ Choose **OK**.
- ☐ Save the part.

#### Step 6 Edit the row.

- ☐ In the KF Navigator expand the root Attributes by clicking the **+** sign.
  - ☐ Place the cursor over the **row** attribute, click MB3 and choose **Edit**. The Edit Attribute dialog displays.
  - ☐ Edit **1** to be any value from **1** to **6**. Choose **Apply**.
- Notice that the block dimensions change.
- ☐ Close the part.

This concludes the activity.



## SUMMARY

In this lesson you:

- Used `ug_spreadsheet` class to access the spreadsheet data.







**(This Page Intentionally Left Blank)**

# Database Access

## Appendix B

---

**PURPOSE**

To introduce the external database access functionality of Knowledge Fusion.

**OBJECTIVES**

Upon completion of this lesson, you will be able to:

- Create a Microsoft Access database.
- Create a data source using ODBC driver.
- Use `ug_odbc_database` class to connect to an external database.
- Use `ug_odbc_recordset` class to access data in an ODBC database.

This lesson contains the following activity.

**Activity****Page**

B-1 Use a Database to Control Door Properties . . . . . B-7

## ***The ODBC Interface***

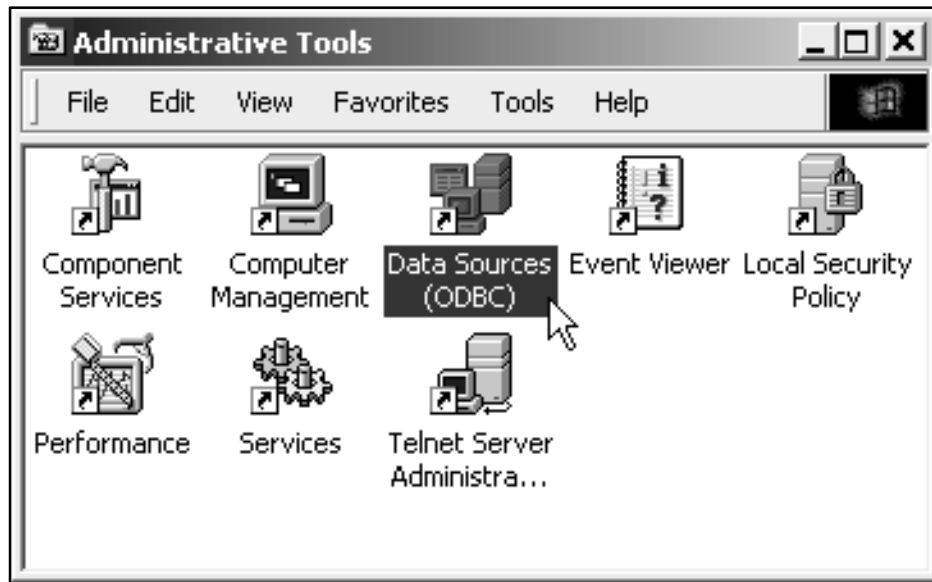
Knowledge Fusion has a built-in Open Database Connectivity (ODBC) interface for external database access, allowing access from a Windows 2000 platform to ODBC data sources. This functionality works as a one-way, server-based utility. Knowledge Fusion will query the database for requested information, and Knowledge Fusion can write information to the database. But you, the user, must initiate updates to the Knowledge Fusion model when changes are made to the database.

Knowledge Fusion language Open Database Connectivity (ODBC) classes, `ug_odbc_database` and `ug_odbc_recordset`, provide an easy-to-use mechanism with which to read and write data in ODBC databases. As discussed below, the `ug_odbc_database` class connects to the external database. Then, the `ug_odbc_recordset` executes the string query SQL statement to read or write data.



## ***ug\_odbc\_database class***

This class opens a connection to an ODBC database. You can then create ODBC\_Recordset instances that can query or add to the tables in the database. The ODBC “data source” must be set up prior to using this design. Use the ODBC Data Sources tool available from Windows Administrative Tools for this purpose. See ODBC Drivers and also the Windows 2000 Operating System documentation for further information on the use of the ODBC Data Sources function.



Knowledge Fusion connects to the database when it is first required, usually because of an action in a `ug_odbc_recordset` instance. Knowledge Fusion normally disconnects from the database when the instance is deleted, via an Knowledge Fusion update or via exiting from Knowledge Fusion. The `reuse?:` parameter, however, affects this behavior.

- Input parameters:

- **dsn:** A string specifying the “data source name”, i.e., the database.
- **userID:** A string specifying the user ID to log in to the database. Default is “ADMIN”.
- **password:** A string specifying the password to log in to the database. Default is “”.
- **loginTimeout:** The number of seconds to wait when logging in before getting an error. Default is 15.
- **initialQueryTimeout:** The number of seconds to allow a query to run before getting an error. Default is 15.
- **reuse?:** Reuse database connection flag. Default is TRUE.



## ***ug\_odbc\_recordset class***

A `ug_odbc_recordset` instance represents a table in a database, or an SQL “select” query on a table in a database. Depending on the options that you specify, you may be able to modify the data as well.

The query is executed when the first action in the `ug_odbc_recordset` instance requires it. The first record in the query, if there is at least one, is made to be the current record when the query is first executed. If there are no records in the table, no error is generated. However, most of the moving and “reading” methods require at least one record in the resulting recordset.

- Input parameters:

- **database:** This must be a `ug_odbc_database` instance.
- **Type:** This is a name, either Dynaset or Snapshot. Dynaset recordsets can be modified. Snapshot recordsets are generally faster but cannot be modified.
- **sqlStatement:** This is a string containing a complete SQL “select” statement that defines the data in the recordset.
- **tablename:** A string containing the name of the table to select from.
- **fields:** A String that defines the fields to select. “\*” selects all fields. To select multiple fields, separate field names by commas.
- **where:** A string that specifies a filter for records.
- **orderBy:** A string that specifies how to sort the recordset.



## ODBC Drivers

The Knowledge Fusion ug\_odbc Library does not come with any ODBC drivers. These must be supplied by a third party.

The Knowledge Fusion ug\_odbc Library has been tested with one of the drivers included with Microsoft Office, the driver for Microsoft Access (.mdb) databases. This is the only driver that is officially supported at this time.

Microsoft Office includes other drivers as well, but these have not been tested and are not currently supported. The Knowledge Fusion ug\_odbc library has no driver-specific code in it; any ODBC driver is expected to work.

You select a driver and a file from which to make a data source. The name of the data source is what you need to supply to the ODBC\_Database design.

To create a data source using any ODBC driver:

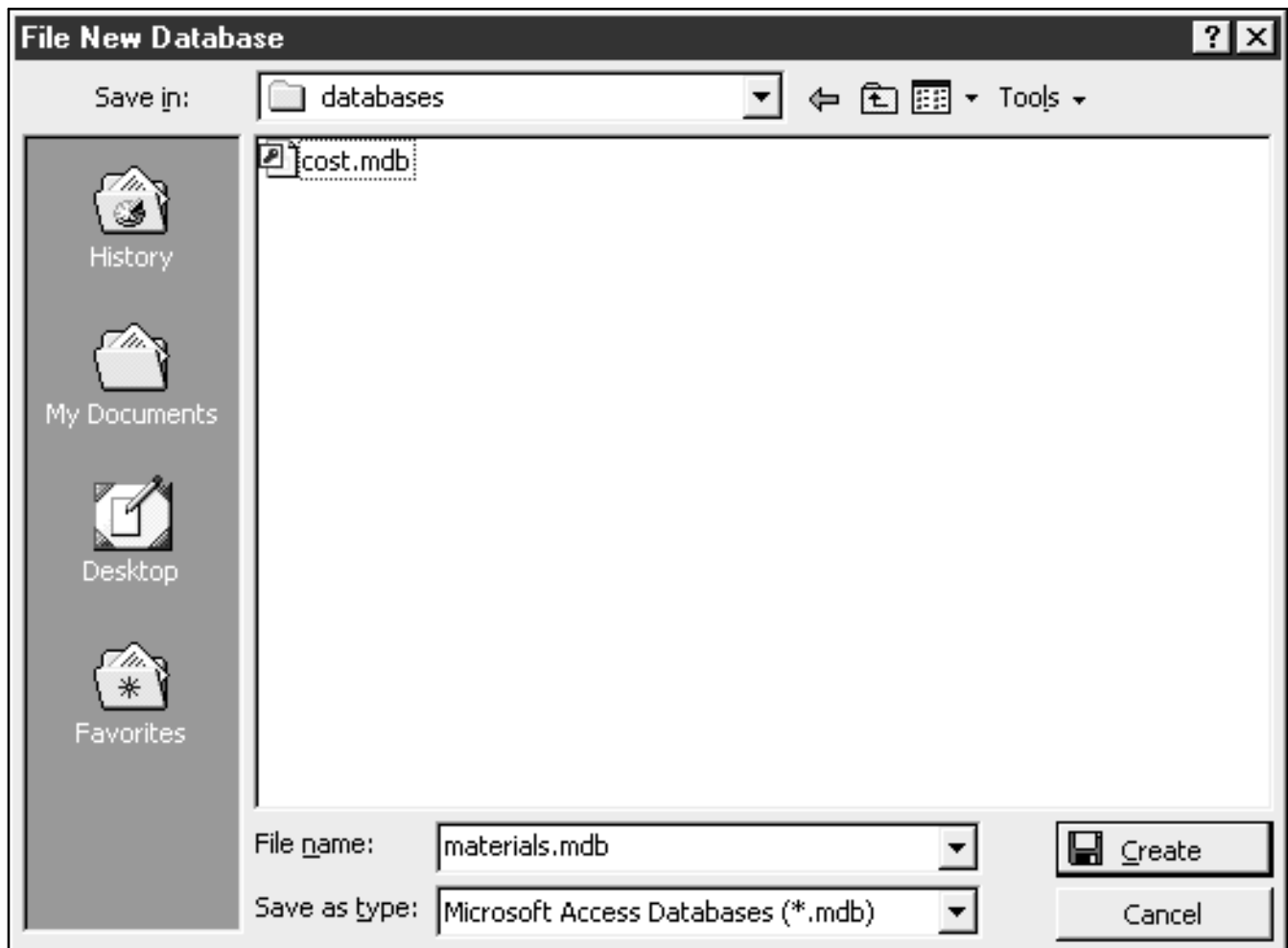
- Go to the Windows Control Panel, double click on **Administrative Tools** icon, and then double click on **Data Sources (ODBC)** icon. Choose the **User DSN** tab, choose the **Add...** button, choose the above referenced Microsoft Access driver, and finally choose **Finish**. This much is the same for all types of drivers.
- A second, ODBC Microsoft Access Setup dialog appears, which is customized for each type of driver. This dialog requires you to type in a string (name) for the DSN in the top text field, and a short description string in the lower text field. Select your .mdb file with the **Select...** button. And choose **OK** until the process completes. The “System Database” is used for security features.

## Activity B–1: Use a Database to Control Door Properties

In this activity, you will use the Knowledge Fusion interface to ODBC database to control door material and weight.

### Step 1 Create a Microsoft Access database.

- ☐ Start up Microsoft Access by going to **Start→Programs→Microsoft Access**.
- ☐ Check the **Blank Access database** radio button and choose **OK**.
- ☐ In the Save in directory, browse to **<your home>\databases**. Enter **materials.mdb** in the File name box. Choose **Create**.





- ☐ Double click **Create table by entering data** icon.



- ☐ Fill in the fields of the table as the following:

Table1 : Table					
	Material	Density	Strength	Modulus	Field5 ▲
	Oak	0.084	22000	9000000	
	A242	0.254	50000	29000000	
	Molded-21	0.054	20000	19000000	
	302	0.286	75000	28000000	
	Fiber-36	0.051	42000	10600000	
	6061-T6	0.980	27000	10000000	
▶					

Record: 7 of 21

**NOTE** You need to double click on the Field tab to replace the Field name with the attribute name. (e.g. replace **Field1** with **Material** and so on)

- ☐ Close and save the table as **properties**. Click **No** when you are prompted if you want to create a primary key.

- ☐ Exit Microsoft Access.

## Step 2 Define a data source.

- ☐ On the desktop, choose **Start→Settings→Control Panel**.
- ☐ Double click on **Administrative Tools**.
- ☐ Double click on **Data Sources (ODBC)**.
- ☐ Choose the **User DSN** tab.
- ☐ Choose **Add...**
- ☐ Select Microsoft Access Driver (\*.mdb).
- ☐ Choose **Finish**.
- ☐ Enter **materials** in the Data Source Name box.
- ☐ Choose **Select...**
- ☐ Find and select the **materials.mdb** database you created in last step.
- ☐ Choose **OK**.
- ☐ Choose **OK**.
- ☐ Choose **OK**.




## Step 3 Retrieve \*\*\*\_door.prt and add a material attribute.

- ☐ If \*\*\*\_door.prt is not open, retrieve this part and open the KF Navigator. Click the **+** sign on Attributes under the root node.
- ☐ Place the cursor over **root** and click MB3. Choose **Add Attribute**. The Add Attribute dialog displays.

- ☐ Enter **material** in the Name box.
- ☐ From the Type list, choose **String**.
- ☐ In the Formula box, enter “**6061–T6**”.
- ☐ Select the **Input Parameter** check box (it should be checked). Verify that **Modifiable** is checked and that **Uncached** is unchecked. **Evaluate It** should be checked.
- ☐ Choose **OK**.




**Step 4 Add the database child rule.**

- ☐ Place the cursor over **root** and click MB3. Choose **Add Child Rule**. The Add Child Rule dialog displays.
- ☐ In the Name Box, enter **db**.
- ☐ From the Class List box, select **ug\_odbc\_database**.
- ☐ From the Input Parameters box, select **\*dsn:**.
- ☐ In the Rule for Parameter box, enter “**materials**”.
- ☐ Choose Apply Typing .
- ☐ Choose **Apply**.

**Step 5 Add the table child rule.**

- ☐ In the Name Box, enter **table**.
- ☐ From the Class List box, select **ug\_odbc\_recordset**.
- ☐ From the Input Parameters box, select **\*database:**.
- ☐ In the Rule for Parameter box, enter **db:**.

- ☐ From the Input Parameters box, select **\*sqlStatement:**.
- ☐ In the Rule for Parameter box, enter  
**“select \* from properties where material=” +  
 “” + material: + “”**.
- ☐ Choose Apply Typing .
- ☐ Choose **OK**.

### Step 6 Add properties, density, and weight attributes.



- ☐ Place the cursor over **root** and click MB3. Choose **Add Attribute**. The Add Attribute dialog displays.
- ☐ Enter **properties** in the Name box.
- ☐ From the Type list, choose **List**.
- ☐ In the Formula box, enter  

```
@{
    table:movefirst();
    table:getrecord();
};
```
- ☐ Choose **Apply**.
- ☐ Enter **density** in the Name box.
- ☐ From the Type list, choose **Number**.
- ☐ In the Formula box, enter **nth(2, properties:);**.
- ☐ Choose **Apply**.
- ☐ Enter **weight** in the Name box.
- ☐ From the Type list, choose **Number**.
- ☐ In the Formula box, enter **density: \* door\_body:volume;;**.

- ☐ Choose **OK**.
- ☐ Save the part.

**Step 7 Edit the material.**

- ☐ Place the cursor over the **material** attribute, click MB3 and choose **Edit**. The Edit Attribute dialog displays.
- ☐ Edit “**6061–T6**” to be any value in the Material column of the database created in step 1.
- ☐ Choose **Apply**.

Notice the change on the weight value. (To update the weight value, place the cursor over the **weight** attribute, click MB3 and choose **Show Value**.)

- ☐ Close the part.

This concludes the activity.



**SUMMARY**

In this lesson you:

- Created a Microsoft Access database.
- Created a data source using ODBC driver.
- Used `ug_odbc_database` class to connect to an external database.
- Used `ug_odbc_recordset` class to access data in an ODBC database.





**(This Page Intentionally Left Blank)**

## Glossary

---

**ABS** – Absolute coordinate system.

**Absolute Coordinate System** – Coordinate system in which all geometry is located from a fixed or absolute zero point.

**active view** – One of up to 49 views per layout in which you can directly work.

**angle** – In Unigraphics, an angle measured on the X-Y plane of a coordinate system is positive if the direction that it is swept is counterclockwise as viewed from the positive Z axis side of the X-Y plane. An angle swept in the opposite direction is said to be negative.

**arc** – An incomplete circle; sometimes used interchangeably with the term “circle.”

**ASCII** – American Standard Code for Information Interchange. It is a set of 8-bit binary numbers representing the alphabet, punctuation, numerals, and other special symbols used in text representation and communications protocol.

**aspect ratio** – The ratio of length to height which represents the change in size of a symbol from its original.

**assembly** – A collection of piece parts and sub-assemblies representing a product. In Unigraphics, an assembly is a part file which contains components.

**assembly part** – A Unigraphics part file which is a user-defined, structured combination of sub-assemblies, components and/or objects.

**associativity** – The ability to tie together (link) separate pieces of information to aid in automating the design, drafting, and manufacture of parts in Unigraphics.

**attributes** – Pieces of information that can be associated with Unigraphics geometry and parts such as assigning a name to an object.

**block font** – A Unigraphics character font which is the default font used for creating text in drafting objects and dimensions.

**body** – Class of objects containing sheets and solids (see solid body and sheet body).

**bottom-up modeling** – Modeling technique where component parts are designed and edited in isolation of their usage within some higher level assembly. All assemblies using the component are automatically updated when opened to reflect the geometric edits made at the piece part level.





**boundary** – A set of geometric objects that describes the containment of a part from a vantage point.

**CAD/CAM** – Computer Aided Design/Computer Aided Manufacturing.

**category, layer** – A name assigned to a layer, or group of layers. A category, if descriptive of the type of data found on the layers to which it is assigned, will assist the user in identifying and managing data in a part file.

**chaining** – A method of selecting a sequence of curves which are joined end-to-end.

**circle** – A complete and closed arc, sometimes used interchangeably with the term “arc.”

**component** – A collection of objects, similar to a group, in an assembly part. A component may be a sub-assembly consisting of other, lower level components.

**component part** – The part file or “master” pointed to by a component within an assembly. The actual geometry is stored in the component part and referenced, not copied, by the assembly. A separate Unigraphics part file that the system associates with a component object in the assembly part.

**cone direction** – Defines the cone direction using the Vector Subfunction.

**cone origin** – Defines the base origin using the Point Subfunction.

**half angle** – The half vertex angle defines the angle formed by the axis of the cone and its side.

**constraints** – Refer to the methods you can use to refine and limit your sketch. The methods of constraining a sketch are geometric and dimensional.

**construction points** – Points used to create a spline. Construction points may be used as poles (control vertices), defining points, or data points. See POLES, DEFINING POINTS, and DATA POINTS.

**control point** – Represents a specific location on an existing object. A line has three control points: both end points and the midpoint of the line. The control point for a closed circle is its center, while the control points for an open arc are its end and midpoints. A spline has a control point at each knot point. A control point is a position on existing geometry. Any of the following points: 1. Existing Points 2. Endpoints of conics 3. Endpoints and midpoints of open arcs 4. Center points of circles 5. Midpoints and endpoints of lines 6. Endpoints of splines.

**convert curve** – A method of creating a b-curve in which curves (lines, arcs, conics or splines) may be selected for conversion into a b-curve.



**Coordinate System** – A system of axes used in specifying positions (CSYS).

**counterclockwise** – The right-hand rule determines the counter- clockwise direction. If the thumb is aligned with the ZC axis and pointing in the positive direction, counterclockwise is defined as the direction the fingers would move from the positive XC axis to the positive YC axis.

**current layout** – The layout currently displayed on the screen. Layout data is kept in an intermediate storage area until it is saved.

**curve** – A curve in Unigraphics is any line, arc, conic, spline or b-curve. A geometric object; this may refer to a line, an arc, a conic, or a spline.

**defaults** – Assumed values when they are not specifically defined.

**defining points** – Spline construction points. Splines created using defining points are forced to pass through the points. These points are guaranteed to be on the spline.

**degree-of-freedom arrows** – Arrow-like indicators that show areas that require more information to fully constrain a sketch.

**design in context** – The ability to directly edit component geometry as it is displayed in the assembly. Geometry from other components can be selected to aid in the modeling. Also referred to as edit in place.



**dimensional constraint** – This is a scalar value or expression which limits the measure of some geometric object such as the length of a line, the radius of an arc, or the distance between two points.

**directory** – A hierarchical file organization structure which contains a list of filenames together with information for locating those files.

**displayed part** – The part currently displayed in the graphics window.

**edit in place** – See design in context.

**emphasize work part** – A color coding option which helps distinguish geometry in the work part from geometry in other parts within the same assembly.

**endpoint** – An endpoint of a curve or an existing point.

**expression** – An arithmetic or conditional statement that has a value. Expressions are used to control dimensions and the relationships between dimensions of a model.

**face** – A region on the outside of a body enclosed by edges.

**feature** – An all-encompassing term which refers to all solids, bodies, and primitives.

**file** – A group or unit of logically related data which is labeled or “named” and associated with a specified space. In Unigraphics, parts, and patterns are a few types of files.

**filtering** – See object filtering.

**font box** – A rectangle or “box” composed of dashed line objects. The font box defines the size, width and spacing of characters belonging to a particular font.

**font, character** – A set of characters designed at a certain size, width and spacing.

**font, line** – Various styles of lines and curves, such as solid, dashed, etc.

**free form feature** – A body of zero thickness. (see body and sheet body)

**generator curve** – A contiguous set of curves, either open or closed, that can be swept or revolved to create a body.

**geometric constraint** – A relationship between one or more geometric objects that forces a limitation. For example, two lines that are perpendicular or parallel specifies a geometric constraint.

**grid** – A rectangular array of implied points used to accurately align locations which are entered by using the “screen position” option.

**guide curve** – A set of contiguous curves that define a path for a sweep operation.

**virtual intersection** – Intersection formed by extending two line segments that do not touch to the position that they cross. The line segments must be non-parallel and coplanar.

**inflection** – A point on a spline where the curve changes from concave to convex, or vice versa.

**interactive step** – An individual menu in a sequence of menus used in performing a Unigraphics function.

**isometric view (Tfr-ISO)** – Isometric view orientation – one where equal distances along the coordinate axes are also equal to the view plane. One of the axes is vertical.

**knot points** – The defining points of a spline. Points along a B-spline, representing the endpoints of each spline segment.



**layer** – A layer is a partition of a part. Layers are analogous to the transparent material used by conventional designers. For example, the user may create all geometry on one layer, all text and dimensions on a second, and tool paths on a third.

**layout** – A collection of viewports or window areas, in which views are displayed. The standard layouts in Unigraphics include one, two, four or six viewports.

**layouts** – Standard layouts are available to the user. These include:

- L1 – Single View,
- L2 – Two Views,
- L3 – Two Views,
- L4 – Four Views,
- L6 – Six Views.

**Information window** – The window used in listing operations, such as **Info**.

**loaded part** – Any part currently opened and in memory. Parts are loaded explicitly using the *File→Open* option and implicitly when they are used in an assembly being opened.

**menu** – A list of options from which the user makes a selection.

**model space** – The coordinate system of a newly created part. This is also referred to as the “absolute coordinate system.” Any other coordinate system may be thought of as a rotation and/or translation of the absolute coordinate system.



**name, expression** – – The name of an expression is the single variable on the left hand side of the expression. All expression names must be unique in a part file. Each expression can have only one name. See expression.

**objects** – All geometry within the Unigraphics environment.

**offset face** – A Unigraphics surface type created by projecting (offsetting) points along all the normals of a selected surface at a specified distance to create a “true” offset.

**options** – A number of various alternatives (functions, modes, parameters, etc.) from among which the user can choose.

**origin** – The point  $X = 0$ ,  $Y = 0$ ,  $Z = 0$  for any particular coordinate system.

**parametric design** – Concept used to define and control the relationships between the features of a model. Concept where the features of the model are defined by parameters.

**part** – A Unigraphics file containing a .prt extension. It may be a piece part containing model geometry, a sub-assembly, or a top-level assembly.

**part or model** – A collection of Unigraphics objects which together may represent some object or structure.

**partially loaded part** – A component part which, for performance reasons, has not been fully loaded. Only those portions of the component part necessary to render the higher level assembly are initially loaded (the reference set).

**point set** – A distribution of points on a curve between two bounding points on that curve.

**Point Subfunction Menu** – A list of options (methods) by which positions can be specified in Unigraphics.

**read-only part** – A part for which the user does not have write access privilege.

**real time dynamics** – Produces smooth pan, zoom, and rotation of a part, though placing great demand on the CPU.

**Refresh** – A function which causes the system to refresh the display list on the viewing screen. This removes temporary display items and fills in holes left by *Blank* or *Delete*.

**right-hand rule, conventional** – The right-hand rule is used to determine the orientation of a coordinate system. If the origin of the coordinate system is in the palm of the right fist, with the back of the hand lying on a table, the outward extension of the index finger corresponds to the positive Y axis, the upward extension of the middle finger corresponds to the positive Z axis, and the outward extension of the thumb corresponds to the positive X axis.

**right-hand rule for rotation** – The right-hand rule for rotation is used to associate vectors with directions of rotation. When the thumb is extended and aligned with a given vector, the curled fingers determine the associated direction of rotation. Conversely, when the curled fingers are held so as to indicate a given direction of rotation, the extended thumb determines the associated vector.

**screen cursor (cursor)** – A marker on the screen which the user moves around using some position indicator device. Used for indicating positions, selecting objects, etc. Takes the form of a full-screen cross.

**sheet** – A object consisting of one or more faces not enclosing a volume. A body of zero-thickness. Also called sheet body.)

**sketch** – A collection of geometric objects that closely approximates the outline of a particular design. You refine your sketch with dimensional and geometric constraints until you achieve a precise representation of your design. The sketch can then be extruded or revolved to obtain a 3D object or feature.



**Sketch Coordinate System (SCS)** – The SCS is a coordinate system which corresponds to the plane of the sketch. When a sketch is created the WCS is changed to the SCS of the new sketch.

**solid body** – An enclosed volume. A type of body (see Body).

**spline** – A smooth free-form curve.

**stored layout** – The last saved version of a layout.

**stored view** – The last saved version of a view.

**string** – A contiguous series of lines and/or arcs connected at their end points.

**sub-assembly** – A part which both contains components and is itself used as a component in higher-level assemblies.

**surface** – The underlying geometry used to define a face on a sheet body. A surface is always a sheet but a sheet is not necessarily a surface (see sheet body). The underlying geometry used to define the shape of a face on a sheet.

**system** – The Unigraphics System.

**temporary part** – An empty part which is optionally created for any component parts which cannot be found in the process of opening an assembly.

**top-down modeling** – Modeling technique where component parts can be created and edited while working at the assembly level. Geometric changes made at the assembly level are automatically reflected in the individual component part when saved.

**trim** – To shorten or extend a curve.

**trimetric view (Tfr-Tri)** – A viewing orientation which provides you with an excellent view of the principal axes. In Unigraphics II, the trimetric view has the Z-axis vertical. The measure along the X-axis is  $\frac{7}{8}$  of the measure along Z, and the measure along the Y-axis is  $\frac{3}{4}$  of the measure along Z.

**Unigraphics** – A computer based turnkey graphics system for computer-aided design, drafting, and manufacturing, produced by UGS.

**units** – The unit of measure in which you may work when constructing in Unigraphics. Upon log on, you may define the unit of measure as inches or millimeters.

**upgraded component** – A component which was originally created pre-V10 but has been opened in V10 and upgraded to remove the duplicate geometry.



**version** – A term which identifies the state of a part with respect to a series of modifications that have been made to the part since its creation.

**view** – A particular display of the model. View parameters include view orientation matrix; center; scale; X,Y and Z clipping bounds; perspective vector; drawing reference point and scale. Eight standard views are available to the user: Top, Front, Right, Left, Bottom, Back, Tfr-ISO (top-front-right isometric), and Tfr-Tri (top-front-right trimetric).

**view dependent edit** – A mode in which the user can edit a part in the current work view only.

**view dependent modifications** – Modifications to the display of geometry in a particular view. These include erase from view and modify color, font and width.

**view dependent geometry** – Geometry created within a particular view. It will only be displayed in that view.

**WCS** – Work Coordinate System.

**WCS, work plane** – The WCS (Work Coordinate System) is the coordinate system singled out by the user for use in construction, verification, etc. The coordinates of the WCS are called work coordinates and are denoted by XC, YC, ZC. The XC-YC plane is called the work plane.

**Work Coordinate System** – See WCS.

**work layer** – The layer on which geometry is being constructed. You may create objects on only one layer at a time.

**work part** – The part in which you create and edit geometry. The work part can be your displayed part or any component part which is contained in your displayed assembly part. When displaying a piece part, the work part is always the same as the displayed part.

**work view** – The view in which work is being performed. When the creation mode is view dependent, any construction and view dependent editing that is performed will occur only in the current work view.

**XC axis** – X-axis of the work coordinate system.

**YC axis** – Y-axis of the work coordinate system.

**ZC axis** – Z-axis of the work coordinate system.



# Index

## A

ABS, GL-1  
 Absolute Coordinate System, GL-1  
 Active View, GL-1  
 Add Attribute, 2-15  
     Basic Procedure, 2-15  
 Add Child Rule, 2-23  
 Adopt Existing Object, 4-3  
 Adoption, 4-2  
     Procedure, 4-5  
 Angle, GL-1  
 Arc, GL-1  
 ASCII, GL-1  
 Aspect Ratio, GL-1  
 Assemblies, 6-2, GL-1  
 Associativity, GL-1  
 Attribute, GL-1  
 Attribute  
     Behavior Options, 2-17  
     Types, 2-16

## B

Body, GL-1  
 Bottom-Up Modeling, GL-1  
 Boundary, GL-2

## C

Category, Layer, GL-2  
 Chaining, GL-2  
 Circle, GL-2  
 Class, 1-9  
 Component, GL-2  
     Part, GL-2  
 Cone  
     Direction, GL-2  
     Origin, GL-2  
 Constraints, GL-2

Construction Points, GL-2  
 Control Point, GL-2  
 Controlling  
     Color, 3-7  
     Interactive Edit, 3-11  
     Layer, 3-7  
     Mass Properties, 3-8  
     Suppression Status, 3-7  
     Topology, 3-2  
 Convert, Curves to B-Curves, GL-2  
 Coordinate Systems, GL-3  
     Sketch, GL-7  
 Counterclockwise, GL-3  
 Current Layout, GL-3  
 Cursor, GL-6  
 Curve, GL-3

## D

Defaults, GL-3  
 Defining Points, GL-3  
 Degree-of-freedom Arrows, GL-3  
 Design in Context, GL-3  
 DFA, 1-7  
 Dimension Constraints, GL-3  
 Direction, Cone, GL-2  
 Directory, GL-3  
 Displayed Part, GL-3

## E

Edit in Place, GL-3  
 Emphasize Work Part, GL-3  
 Endpoint, GL-3  
 Expressions, 3-17, GL-3  
     Names, GL-5

## F

Face, GL-3  
 Features, GL-4





File, GL-4  
Filtering, GL-4  
Font  
    Box, GL-4  
    Character, GL-4  
    Line, GL-4  
Free Form Feature, GL-4

## **G**

Generator Curve, GL-4  
Geometric Constraint, GL-4  
Grid, GL-4  
Guide Curve, GL-4

## **H**

Half Angle, GL-2

## **I**

Inflection, GL-4  
Instantiation, 1-9

## **K**

KBE, 1-2  
KF Class, 2-33  
    UG System Class, 2-33  
    ug\_askKFattrValue, 6-3  
    ug\_block, 2-27  
    ug\_body, 3-8  
    ug\_child\_in\_part, 6-3  
    ug\_component, 6-7  
    ug\_cylinder, 2-38  
    ug\_evaluateInPart, 6-2  
    ug\_expression, 3-17  
    ug\_mass\_properties, 3-8  
    ug\_odbc\_database, B-3  
    ug\_odbc\_recordset, B-5  
    ug\_optimize, 7-4  
    ug\_spreadsheet, A-2, A-4  
    ug\_udfs, 5-2, 5-15  
    User Defined Class, 2-33  
Knot Points, GL-4  
knowledge base, 1-2  
Knowledge Based Engineering, 1-2

Knowledge Fusion, Navigator, 2-2  
    Attribute Node, 2-4  
    Child Node, 2-4  
    Instance Node, 2-4  
    open, 2-3  
    Root Node, 2-4  
Knowledge Fusion Attribute, 1-11  
Knowledge Fusion Class, 1-10  
Knowledge Fusion Object, 1-10  
Knowledge Fusion Rule, 1-11  
Knowledge Fusion, 1-6  
    language, 1-6  
    Navigator, 1-6  
    Toolbar, 1-12

## **L**

Layer, GL-5  
Layout, GL-5  
Listing Window, GL-5  
Loaded Part, GL-5

## **M**

MB3 Pop-up Menus, 2-5  
Menu, GL-5  
Model, GL-6  
Model Space, GL-5

## **O**

Object, 1-9, GL-5  
Object Oriented, 1-9  
ODBC Drivers, B-6  
ODBC Interface, B-2  
Offset Surface, GL-5  
Open Database Connectivity, B-2  
Optimization, 7-2  
    Constraints, 7-3  
    Convergence Criteria, 7-3  
    Design Variables, 7-2  
    General Process, 7-4  
    Objective, 7-2  
Origin, Cone, GL-2

## **P**

Parametric Design, GL-5

Part, GL-5, GL-6

Partially Loaded Part, GL-6

Point Set, GL-6

Point Subfunction, GL-6

## R

Read-Only Part, GL-6

Real Time Dynamics, GL-6

Refresh, GL-6

Right Hand Rule, GL-6

Rotation, GL-6

## S

SCS, GL-7

Sheet, GL-6

Sketch, GL-6

Coordinate System, GL-7

Solid Body, GL-7

Specify Rule for Class, 3-2

Procedure, 3-3

Spline, GL-7

Spreadsheet, A-2

Stored Layout, GL-7

Stored View, GL-7

String, GL-7

Sub-assembly, GL-7

Surface, GL-7

System, GL-7

## T

Temporary Part, GL-7

Tfr-ISO, GL-4

Tfr-Tri, GL-7

Top-Down Modeling, GL-7

Trim, GL-7

## U

UDF, 5-2

Positioning

Reference Frames, 5-8

Reference Geometry, 5-10

UDF Dialog, 5-18

Customizing, 5-18

UDF , Positioning, 5-8

Unigraphics, GL-7

Units, GL-7

Upgrade, Component, GL-7

User Defined Features, 5-2

Exporting, 5-3

Swap, 5-27

## V

Version, GL-8

View, GL-8

Isometric, GL-4

Trimetric, GL-7

Work, GL-8

## W

WCS, GL-8

Work Layer, GL-8

Work Part, GL-8

## X

XC-Axis, GL-8

## Y

YC-Axis, GL-8

## Z

ZC-Axis, GL-8



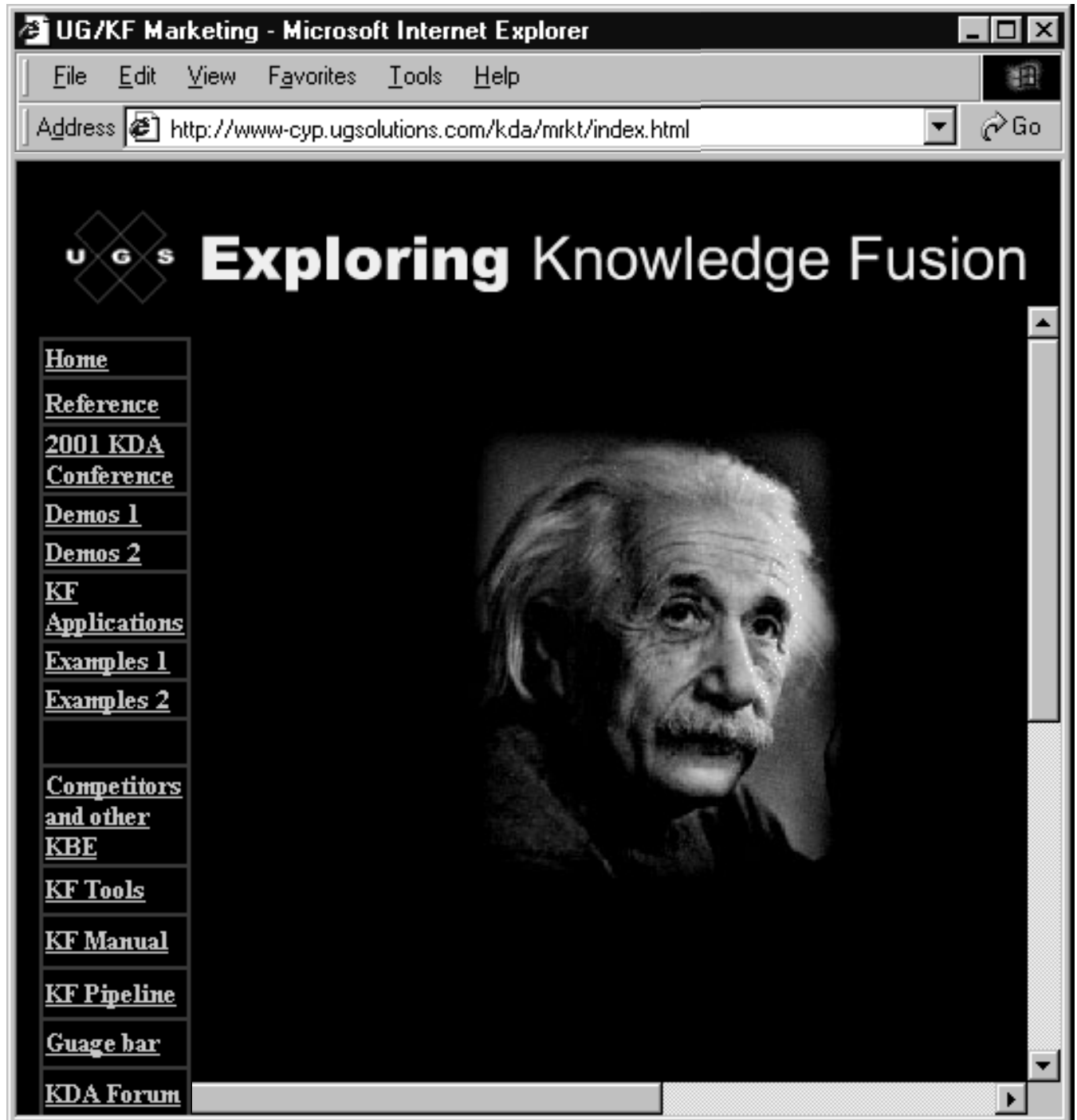
**(This Page Intentionally Left Blank)**



---

## Web Site of Knowledge Fusion

<http://www-cyp.ugsolutions.com/kda/mrkt/index.html>



---

**(This Page Intentionally Left Blank)**

---

## V18 Knowledge Fusion for Designers Course Agenda

### Day 1            Morning



- Welcome
- Lesson 1. Introduction to Knowledge Fusion
- Lesson 2. Knowledge Fusion Navigator (begin)

#### Afternoon

- Lesson 2. Knowledge Fusion Navigator (continued)
- Lesson 3. Design Control

### Day 2            Morning



- Lesson 4. Adoption
- Lesson 5. User Defined Features (begin)

#### Afternoon

- Lesson 5. User Defined Features (continued)

### Day 3            Morning



- Lesson 6. Assembly
- Lesson 7. Optimization
- Brief Overview of the “KF for Programmers” Class

#### Afternoon

- Question and Answer
- Hands-On

---

**(This Page Intentionally Left Blank)**



# Student Profile

## Knowledge Fusion for Designers

Name \_\_\_\_\_ Date \_\_\_\_\_

Employer \_\_\_\_\_

U.S. citizen? Yes / No

When is your planned departure time? \_\_\_\_\_ am/pm

*Please answer the following questions as honestly as you can. We are concerned about providing training that meets your needs. If you have any additional comments please write them on the back of this form.*

1. Job title: \_\_\_\_\_

2. Current responsibilities: \_\_\_\_\_

3. How long have you held these responsibilities? Years \_\_\_\_\_ Months \_\_\_\_\_

4. How long have you been working with CAD/CAM/CAE systems? Years \_\_\_\_\_

5. What other CAD/CAM/CAE systems are you familiar with? \_\_\_\_\_

6. Are you currently using Unigraphics? \_\_\_\_\_ Version \_\_\_\_\_ Hours per week? \_\_\_\_\_

7. What is the function of your CAD/CAM/CAE system (documentation, modeling, analysis, translation interface, etc.)? \_\_\_\_\_

8. What do you model in your Unigraphics part files (castings, assemblies, floor plans, etc.)? \_\_\_\_\_

9. Please list other completed CAD/CAM/CAE courses and the provider including *Unigraphics CBT* and *CAST*:

**Course**

**Provider**

_____	_____
_____	_____
_____	_____
_____	_____

10. Please check the box that best describes your current skill level in the various Unigraphics disciplines listed below.

	none	novice	intermediate	advanced	future use
Wireframe Modeling					
Solid Modeling					
Parametric Modeling					
Drafting					
Assemblies					
Manufacturing					



[illegible]



# Knowledge Fusion for Designers Training Course Evaluation

Name (Optional) \_\_\_\_\_ Date \_\_\_\_\_

Instructor \_\_\_\_\_ Location \_\_\_\_\_ UG Version 18

Please give your **honest** opinion about the training you have received during this class. Provide additional comments on the reverse side of this evaluation form.

☐ Please check the box if you would like your comments, regarding the training you just received, featured in our training publications. We will contact you if more information is needed.

**Hotel Accommodations** (if applicable) Hotel name \_\_\_\_\_

What was your overall impression of this hotel? Poor 2 3 4 5 6 7

**Facilities** – How would you rate the training facilities? Poor 2 3 4 5 6 7

**Instruction** – How would you rate the instruction? Poor 2 3 4 5 6 7

Was the instructor knowledgeable of the subject? Poor 2 3 4 5 6 7

Comments \_\_\_\_\_

1. Were the course objectives clearly defined and were they met? Yes No

Please explain: \_\_\_\_\_

2. Were concepts effectively communicated so that you understand how to apply the software? Yes

No Please explain: \_\_\_\_\_

3. How well prepared do you now feel to use the functions covered in this course in your day to day activities? Please explain: \_\_\_\_\_

4. Were the student activities effective in learning Knowledge Fusion? Yes No

Please explain: \_\_\_\_\_

over

5. What additional topics related to Knowledge Fusion would you like to see covered in this course?  
Please explain: \_\_\_\_\_  
\_\_\_\_\_  
\_\_\_\_\_
6. Do you have any other suggestions on how the course could be improved?    Yes    No  
Please explain: \_\_\_\_\_  
\_\_\_\_\_  
\_\_\_\_\_
7. In order to continually improve our courseware a post class survey is conducted; would you be willing to participate in this survey. ☐ (If you checked this box, make sure that your name is on this sheet.)

**Course –** What was your overall impression?

Poor 2 3 4 5 6 7

Additional Comments

## Hot Key Chart

Hot Key	Function	Hot Key	Function
Ctrl-A	Assembly Navigator	Ctrl-N	File, New
Ctrl-B	Edit, Blank	Ctrl-O	File, Open
Ctrl-C	Copy	Ctrl-P	File, Plot
Ctrl-D	Delete	Ctrl-Q	
Ctrl-E	Tools, Expression	Ctrl-R	View, Operation, Rotate (full menu)
Ctrl-F	Fit View	Ctrl-S	File, Save
Ctrl-G	Grip Execute	Ctrl-T	Edit, Transform
Ctrl-H		Ctrl-U	Execute User Function
Ctrl-I	Information, Object	Ctrl-V	Paste
Ctrl-J	Edit, Object Display	Ctrl-W	Application, Gateway
Ctrl-K		Ctrl-X	Cut
Ctrl-L	Format, Layer Settings	Ctrl-Y	
Ctrl-M	Application, Modeling	Ctrl-Z	Edit, Undo

Ctrl-Shift-A	File, Save As	Ctrl-Shift-N	Format, Layout, New
Ctrl-Shift-B	Edit, Blank, Reverse Blank All	Ctrl-Shift-O	Format, Layout, Open
Ctrl-Shift-C	View, Curvature Graph	Ctrl-Shift-P	Tools, Macro, Playback
Ctrl-Shift-D		Ctrl-Shift-Q	Quick Shaded Image
Ctrl-Shift-E		Ctrl-Shift-R	Tools, Macro, Record
Ctrl-Shift-F	Format, Layout, Fit All Views	Ctrl-Shift-S	
Ctrl-Shift-G	Debug Grip	Ctrl-Shift-T	Preferences, Selection
Ctrl-Shift-H	High Quality Image	Ctrl-Shift-U	Edit, Blank, Unblank All Of Part
Ctrl-Shift-I		Ctrl-Shift-V	Format, Visible In View
Ctrl-Shift-J	Preferences, Object	Ctrl-Shift-W	
Ctrl-Shift-K	Edit, Blank, Unblank Selected	Ctrl-Shift-X	
Ctrl-Shift-L		Ctrl-Shift-Y	
Ctrl-Shift-M	Model Navigator	Ctrl-Shift-Z	View, Operation, Zoom (full menu)

---

Hot Key	Function	Hot Key	Function
Alt-Tab	Toggles Application	Ctrl-Alt-W	Application Assemblies
Alt-F4	Closes Active Window	Ctrl-Alt-	
F1	Help on Context	Ctrl-Alt-M	Application Manufac- turing
F3	View Current Dialog	Ctrl-Alt-O	Operation Navigator
F4	Information Window	Ctrl-Alt-X	Tools, Lathe Cross – Section
F5	Refresh	Ctrl-Alt-C	Tools, CLSF
F6	Quick Zoom	Ctrl-Alt-B	Tools, Boundary
F7	Quick Rotate	Ctrl-Alt-N	Tools, Unisim



