STUDENT GUIDE

Student Notes:

CATIA V5 Training

Foils

Knowledge Based Engineering

Version 5 Release 18
December 2007

EDU_CAT_EN_KBE_FF_V5R18

Student Notes:

About this course

Objectives of the course

Upon completion of this course you will be able to:

- Become familiar with the Knowledgeware working environment, how it can be accessed, the terminology that will be used and the Settings.

- Create a parametric parts and assemblies.

- Embed knowledge in your designs by controlling it using parameters, formulae, rules , checks and reactions.

- Create and reuse Power Copies and User Defined Features.

- Create and store Knowledge Driven design templates so as to instantiate them in a new context.

- Create and reuse advanced instantiation features like Knowledge Pattern.

Targeted audience

CATIA V5 users

Prerequisites

Students attending this course should have knowlegde of CATIA V5 Fundamentals.

3 days

Student Notes:

Table of Contents (1/4)

Student Notes:

Table of Contents (2/4)

Student Notes:

Table of Contents (3/4)

Student Notes:

Table of Contents (4/4)

STUDENT GUIDE

Student Notes:

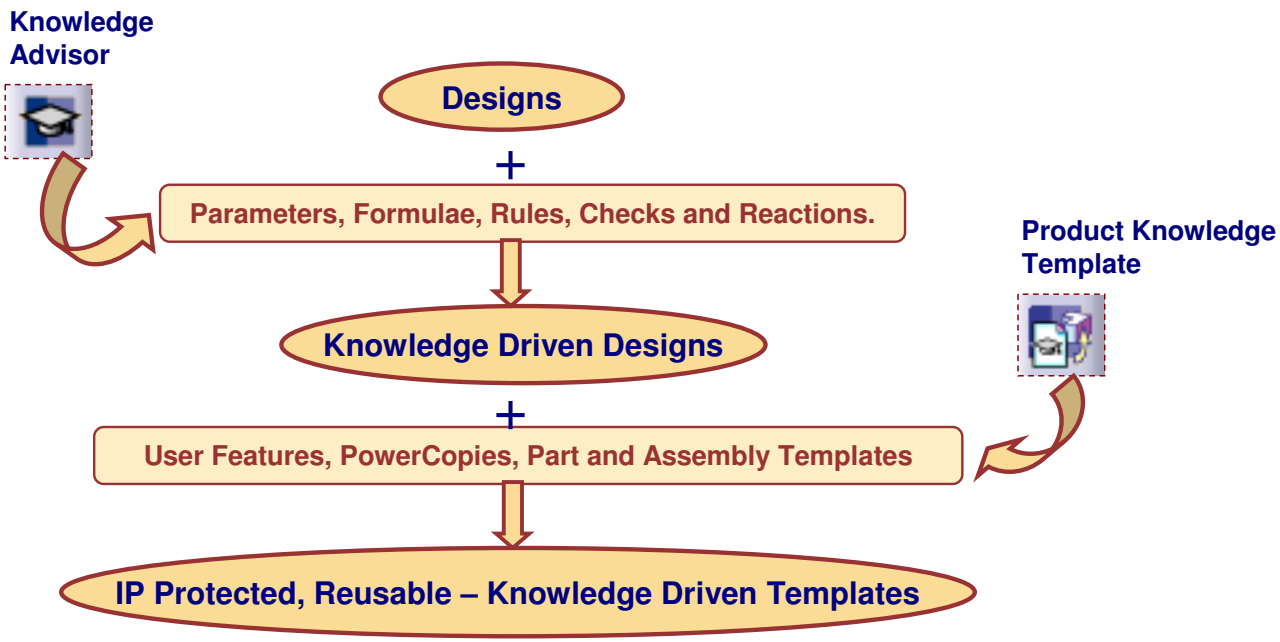
Introduction to KBE – Basic Course

This skillet will give you an overview of the Knowledge Based Engineering - Basic course and about the user settings which are to be made for the course.

Student Notes:

Overview of 'KBE – Basic' course

The Knowledge Based Engineering – Basic course teaches you how the Knowledge Advisor and Product Knowledge Template workbench's functionalities can be used to create Knowledge Driven designs and templates.



Student Notes:

Knowledge User Settings (1/5)

You should make all the settings mentioned in this section while browsing this course or during the exercise replay of the course.

Display and update General Settings:

Check the corresponding option if you need:

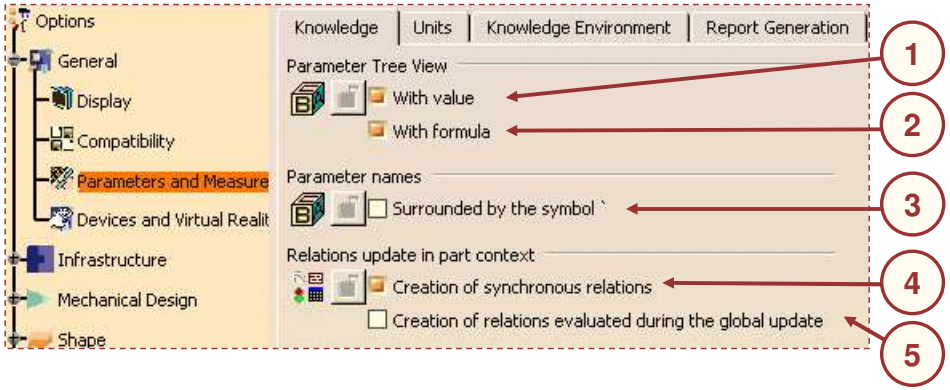
(1) the **value** of the parameter to appear in the tree.

(2) the **formula** driving the parameter to appear in the tree beside the parameter.

(3) to work with non-latin characters. Otherwise, parameter names have to be renamed in latin characters when used.

(4) to **create synchronous relations**, that is to say relations that will be immediately updated if one of their parameters is modified. Relations based on parameters are the only ones that can be synchronous.

(5) to associate the evaluations of **asynchronous relations with the global update**. The relations can be asynchronous for two reasons: the user wants the relations to be asynchronous or the relation contains measures.



Student Notes:

Knowledge User Settings (2/5)

Design Tables General Settings:

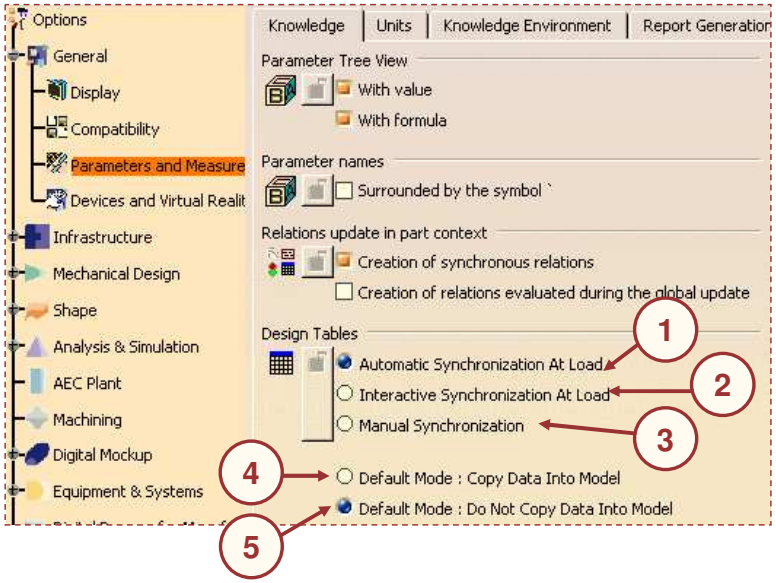
(1) **Automatic Synchronization At Load:** When loading a model containing user design tables, if design table files have been modified and if the external file data is contained in the model, the design table will be synchronized automatically if this button is checked.

(2) **Interactive Synchronization At Load:** When loading a model containing user design tables whose external source file was deleted, this option enables the user to select a new source file or to save the data contained in the design tables in a new file.

(3) **Manual Synchronization:** When loading a model containing user design tables, if the design table files have been modified and the external file data is contained in the model, the design table will be synchronized if this radio button is checked. To synchronize both files, right-click the design table in the specification tree and select the DesignTable object->Synchronize command or the Edit->Links command.

(4) **Default Mode: Copy Data Into Model:** If checked, the data contained in the external source file will be copied into the model.

(5) **Default Mode: Do Not Copy Data Into Model:** If checked, the data contained in the external source file will not be copied into the model.



Student Notes:

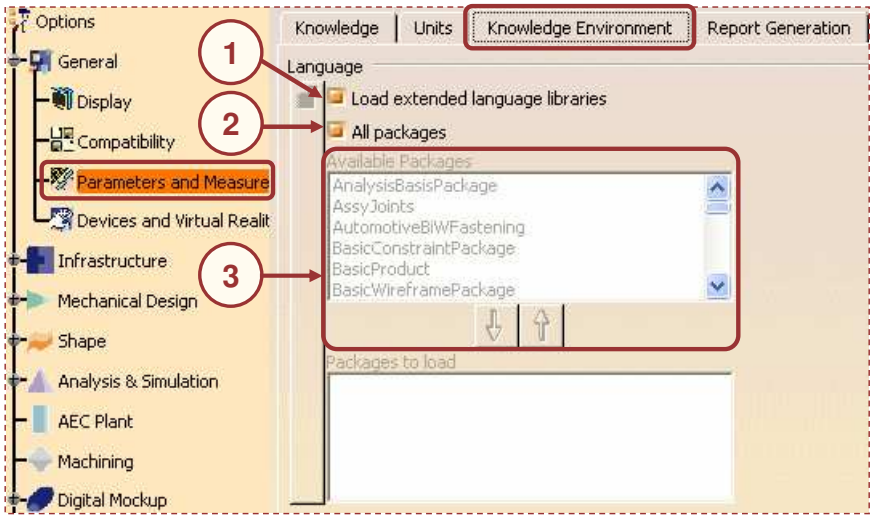
Knowledge User Settings (3/5)

Language Settings:

(1) **Check this option to have access to more language libraries. That means that more functions will be available for the Edition of Relations.**

(2) **Check this button to load ALL the available libraries.**

(3) **Otherwise select libraries packages in the list and use the arrows to add or retrieve them to the list of libraries to be loaded.**



Student Notes:

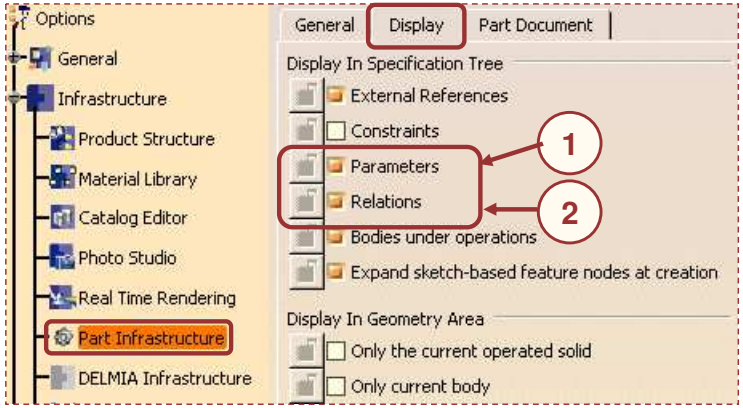
Knowledge User Settings (4/5)

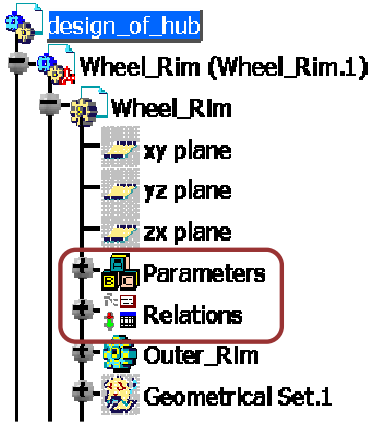
Part Infrastructure Settings:

Check the corresponding options if you need :

(1) the **parameters** of the **part** to be displayed in the specifications tree.

(2) the **relations** of the **part** to be displayed in the specifications tree.





Student Notes:

Knowledge User Settings (5/5)

Product Structure Settings:

Activate the following options if you need :

(1) the **parameters** of the **product** to appear in the specifications tree.

(2) the **relations** of the **product** to appear in the specifications tree.

Knowledge Advisor Workbench Presentation

You will learn what are the main features of the Knowledge Advisor workbench as well as some infrastructure features provided with CATIA V5.

- **Accessing the Workbench**
- **User Interface**
- **Knowledge User Settings**

Student Notes:

Student Notes:

Accessing the Workbench

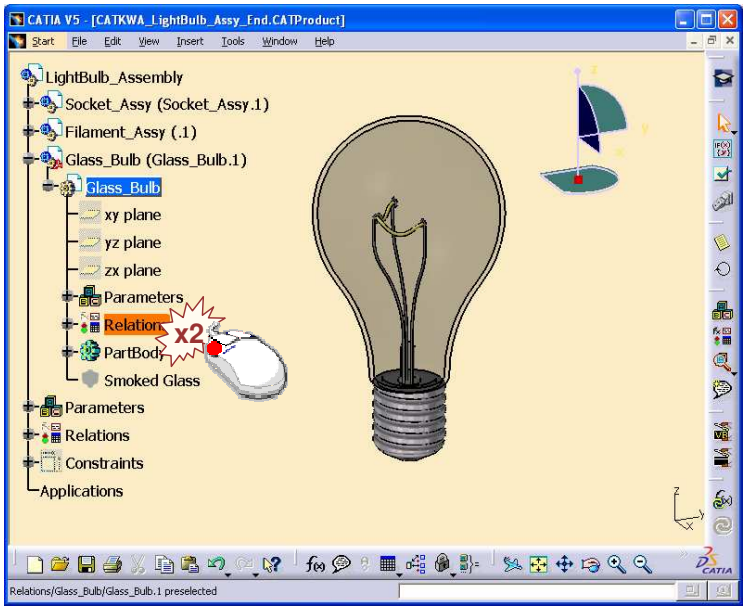
You can access Knowledge Advisor workbench through the usual ways:

A **From the Start Menu**

B **From the workbench icon:**

Go to Tools /Customize /Start Menu to customize the content of this Welcome box

C **From a CATIA Document**



If the Relations node exists in the specification tree, double-click on it to launch Knowledge Advisor workbench.

Student Notes:

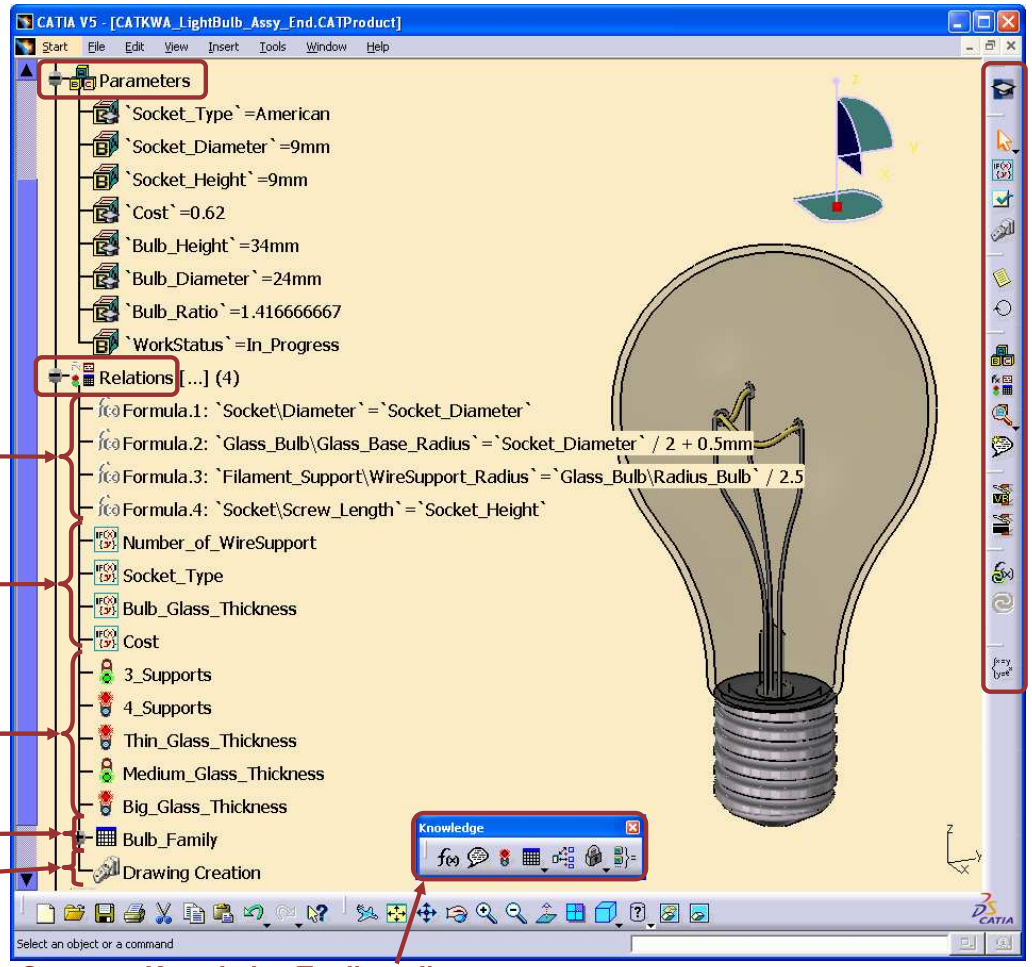
User Interface (1/4)

Parameters node contains User parameters and Lists

Relations node contains:

- Formulas
- Rules
- Checks
- Design Tables
- Reactions

and Macro with arguments



Knowledge Advisor Workbench

Common Knowledge Toolbar allows you to access:
Formulas, Comments & URL, Check Analysis, Design Table creation, Law creation, Knowledge Inspector, Lock/Unlock parameters, Equivalent Dimensions

Student Notes:

User Interface (2/4)

Icon	Name	Definition
	Formula	Simple formulas y=f(x,y,z,…) between any V5 parameters
	Design Table	Tabulated relation of a set of parameters based on an Excel spreadsheet or a text file
	Law	y=f(x) mathematical law that can be used by geometric or analysis operators
	Knowledge Inspector	Allows to evaluate the impact of modifications (what if) and How to modify parameters
	Lock selected parameters	Locks or unlocks selected parameters
	EquivalentDimensions	Enables the user to apply the same value to selected Angle or Length parameters.
	Comment & URLs	Searches for URLs assigned to user parameters or relations

Student Notes:

User Interface (3/4)

Icon	Name	Definition
	Rule	Rule embedded in design that reacts to parameter changes and propagates parameter or geometric modifications
	Check	Check embedded in design that reacts to parameter changes and informs the user in case of violation
	Reaction	Feature embedded in design that reacts to specific events and propagates any kind of modifications
	List	List referencing a set of objects (parameters or geometric features). May compute list size, sum, min, max, etc…
	Loop	Loop similar to loop in languages that manages the creation, destruction or modification of a set of features. Loop is superseded by the powerful 'Knowledge Pattern' function of Product Knowledge Template Workbench.
	Add Set of parameters	Creates a node of parameters
	Add Set of Relations	Creates a node of Relations

Student Notes:

User Interface (4/4)

Icon	Name	Definition
	Parameters Explorer	Creates user parameters stored at feature level
	Add parameters on geometry	Adds parameters to an edge, a face or a vertex
	Comment & URLs	Adds URLs on user parameters or relations and searches for existing URLs
	Macros with arguments	Feature to run VBScript macros with arguments. Can be called from a Rule or a Reaction
	Action	Feature that describes a function that a user can decide to execute
	Measure Update	Updates relations using measures
	Set of Equations	Mathematical set of equations and inequations that drives a set of output parameters according to changes in input parameters

Student Notes:

Knowledge User Settings (1/5)

Display and update General Settings:

Check the corresponding option if you need:

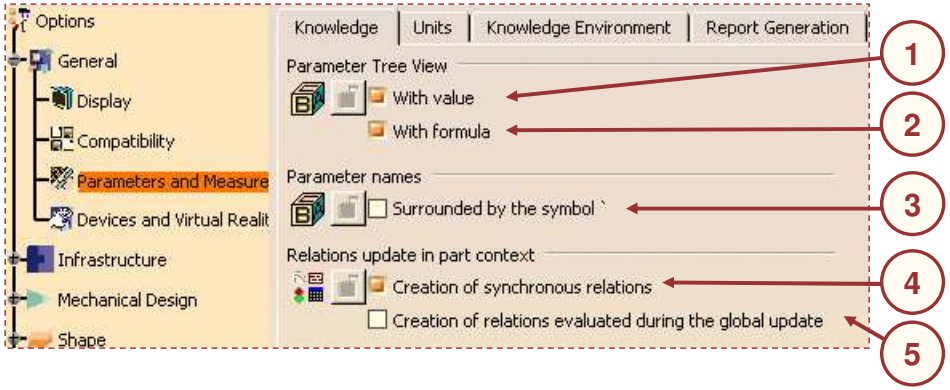
(1) the **value** of the parameter to appear in the tree.

(2) the **formula** driving the parameter to appear in the tree beside the parameter.

(3) to work with non-latin characters. Otherwise, parameter names have to be renamed in latin characters when used.

(4) to **create synchronous relations**, that is to say relations that will be immediately updated if one of their parameters is modified. Relations based on parameters are the only ones that can be synchronous.

(5) to associate the evaluations of **asynchronous relations with the global update**. The relations can be asynchronous for two reasons: the user wants the relations to be asynchronous or the relation contains measures.



Student Notes:

Knowledge User Settings (2/5)

Design Tables General Settings:

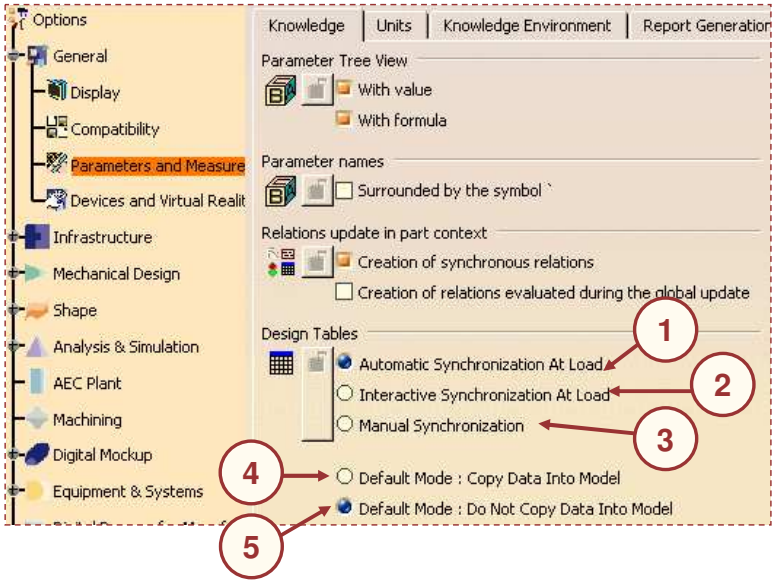
(1) **Automatic Synchronization At Load:** When loading a model containing user design tables, if design table files have been modified and if the external file data is contained in the model, the design table will be synchronized automatically if this button is checked.

(2) **Interactive Synchronization At Load:** When loading a model containing user design tables whose external source file was deleted, this option enables the user to select a new source file or to save the data contained in the design tables in a new file.

(3) **Manual Synchronization:** When loading a model containing user design tables, if the design table files have been modified and the external file data is contained in the model, the design table will be synchronized if this radio button is checked. To synchronize both files, right-click the design table in the specification tree and select the DesignTable object->Synchronize command or the Edit->Links command.

(4) **Default Mode: Copy Data Into Model:** If checked, the data contained in the external source file will be copied into the model.

(5) **Default Mode: Do Not Copy Data Into Model:** If checked, the data contained in the external source file will not be copied into the model.



Student Notes:

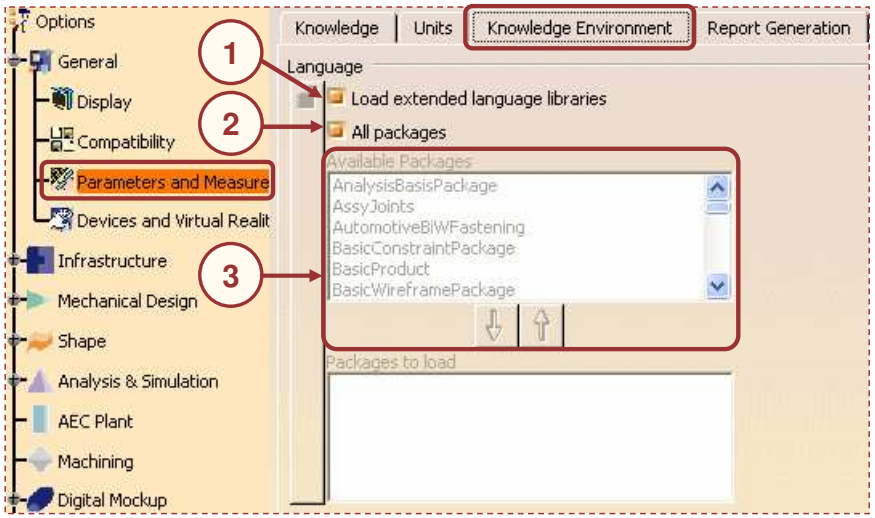
Knowledge User Settings (3/5)

Language Settings:

(1) Check this option to have access to more language libraries. That means that more functions will be available for the Edition of Relations.

(2) Check this button to load ALL the available libraries.

(3) Otherwise select libraries packages in the list and use the arrows to add or retrieve them to the list of libraries to be loaded.



Student Notes:

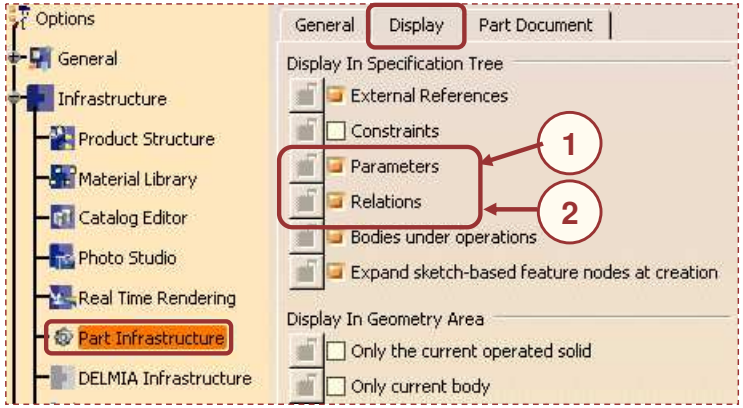
Knowledge User Settings (4/5)

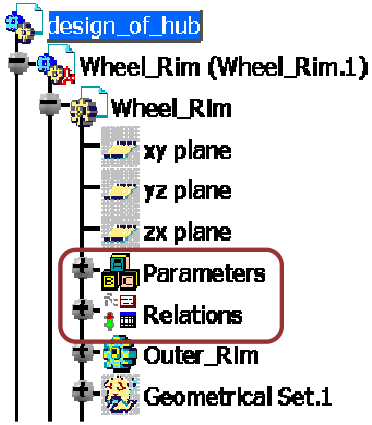
Part Infrastructure Settings:

Check the corresponding options if you need :

(1) the **parameters** of the **part** to be displayed in the specifications tree.

(2) the **relations** of the **part** to be displayed in the specifications tree.





Student Notes:

Knowledge User Settings (5/5)

Product Structure Settings:

Activate the following options if you need :

(1) the **parameters** of the **product** to appear in the specifications tree.

(2) the **relations** of the **product** to appear in the specifications tree.

Student Notes:

Creating Parameters, Formulas and Lists

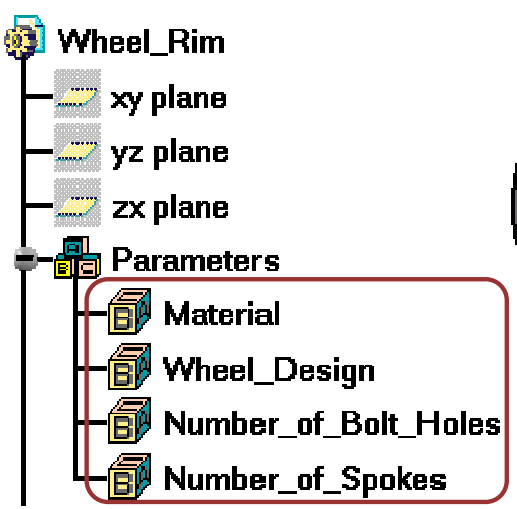
You will learn how to create user parameters and how to reuse them in formulas and equations. You will also learn how to create lists of features.

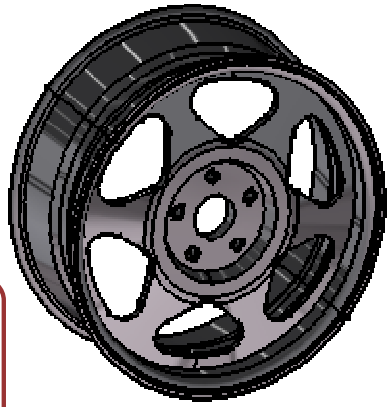
- **Creating User Parameters**
- **Creating and Using Formulas**
- **Creating Lists**
- **Associating URLs to Parameters and Relations**

Student Notes:

Creating User Parameters

You will learn how to create and manage parameters.

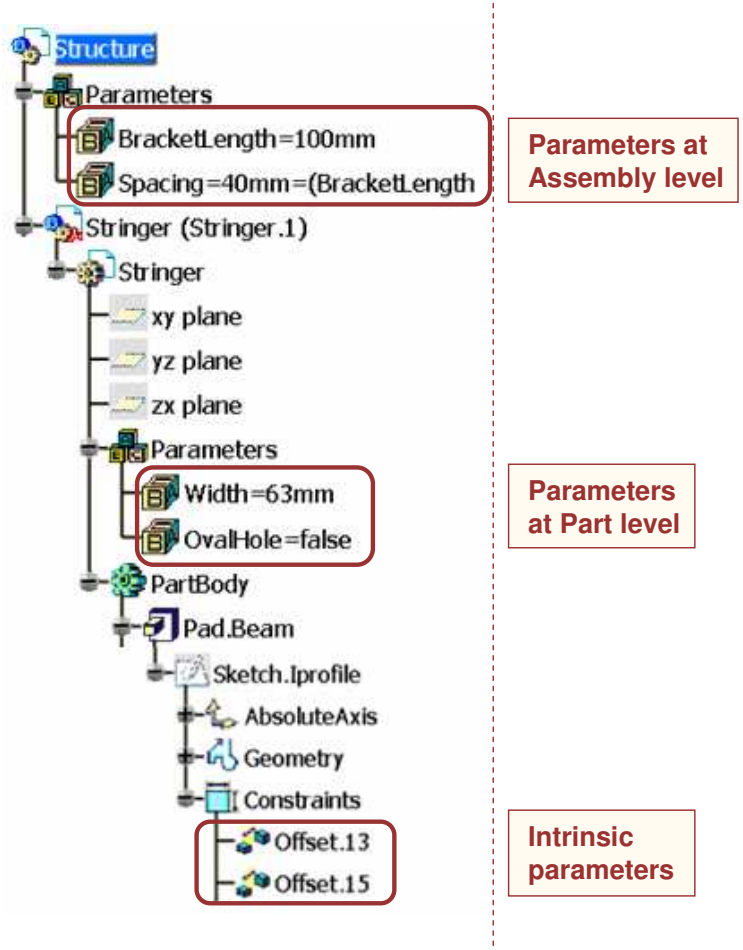




Student Notes:

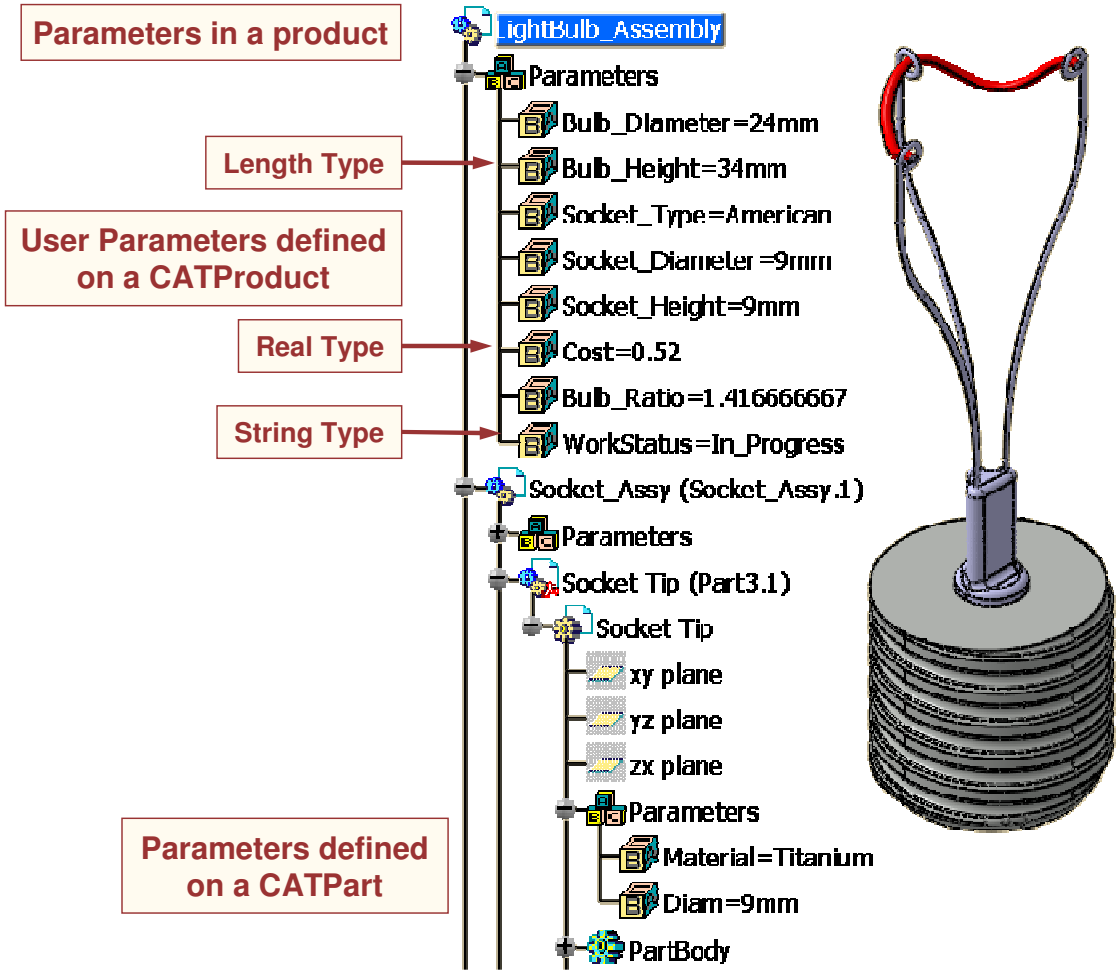
What are Parameters? (1/2)

- There are many types of parameters:
  Real, Integer, String, Boolean, Length, Mass...
- Two kinds of parameters:
  - Intrinsic Pparameters are generated while creating any geometry and features. They define the intrinsic properties of the features (depth, offset, activity, …)
  - User Parameters are especially created by the user. They define the extra pieces of information added to a document. The User Parameters can be defined at different levels:
    - Part level
    - Assembly level
- User Parameters can either be defined:
  - With a single value (continuous). In this case, the parameter can take any value.
  - Or with multiple values (discrete). In this case, the parameter can take only the pre-defined values given at its creation.
- Any parameter can be:
  - Defined or constrained by relations
  - Used as argument of relations



Student Notes:

What are Parameters? (2/2)

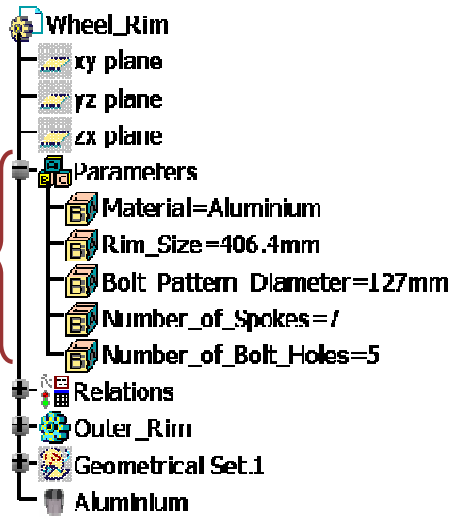


Student Notes:

Why Use User Parameters?

- To have an immediate access to the parameters that pilot the geometry and to change easily their value.
- To centralize key information so that any new user on the model can use it immediately.
- To refer easily to the same parameter when editing relations.
- With User Parameters, you can create generic models that are driven only from the User Parameter node.

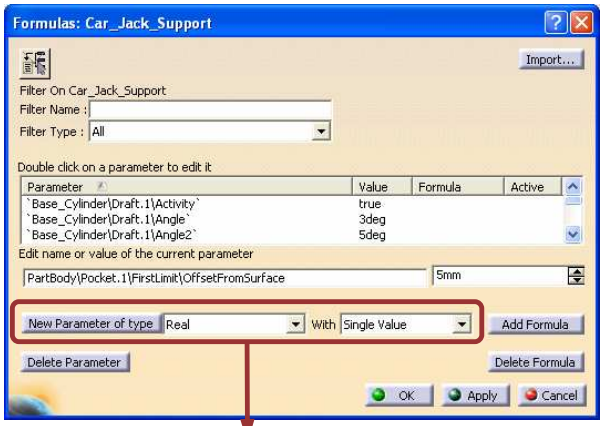
All the key information of the model is accessible from this place of the part, so that you do not need to search in the PartBody to change the number of spokes, for instance.



Student Notes:

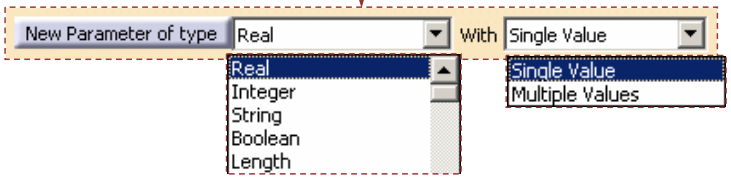
Creating User Parameters (1/2)

1. Click on the *f(x)* icon.
   The Formulas panel is displayed.

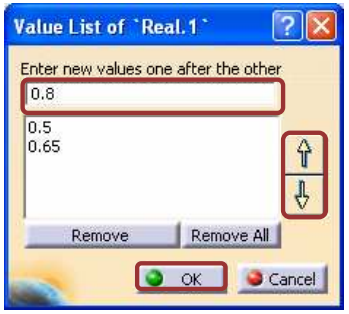


2. Select the desired type of parameter and then specify the Single Value or the Multi Values option.

3. Click the New Parameter of type button to create the parameter.



The **Multiple Values** option allows you to pre-define fixed values for the parameter.
In this case, you are required to enter the values of the parameter as soon as you click the New Parameter of type button. The "Value List" panel appears.



Type here the different values of the parameter. Click the Enter button to validate each value.

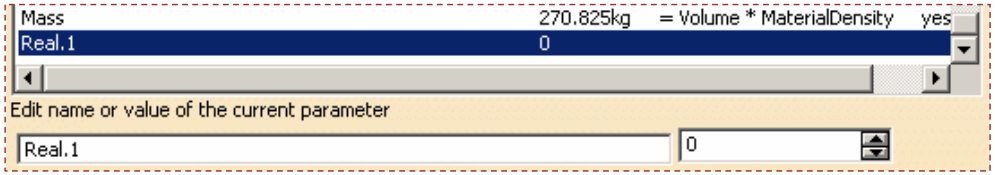
Use these arrows to reorder the values.

Click OK when finished.

Student Notes:

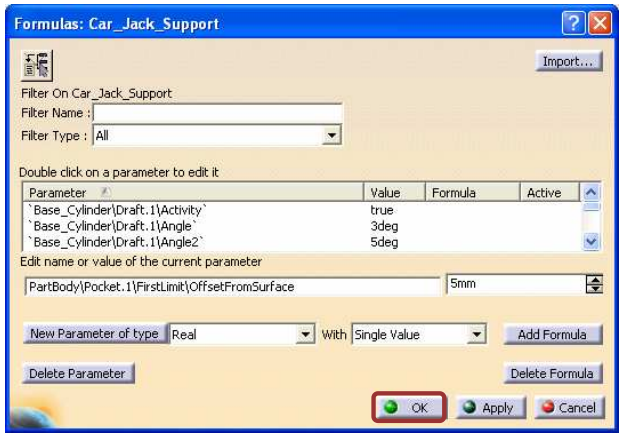
Creating User Parameters (2/2)

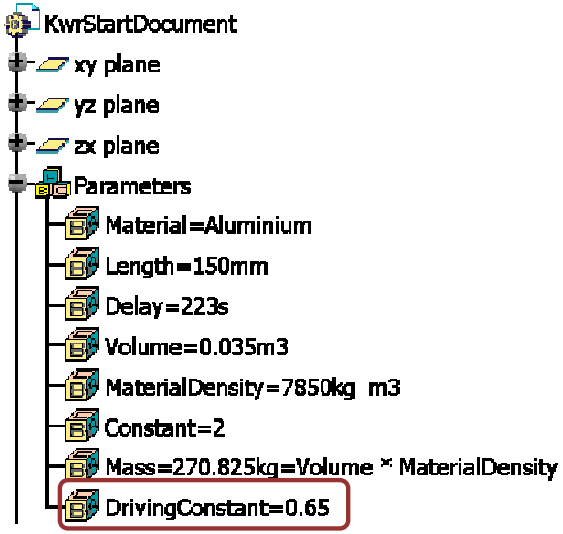
4 **The new parameter appears at the end of the parameters list with a default name (here *Real.1*) and a default value *0*.**



5 **You can rename the parameter by typing a new name in the Edit name field; and attribute it a value by filling the Edit value field.**

6 **The OK button validates the creation of the parameter and closes the Formulas panel.**
The new User Parameter is added to the specification tree.





Student Notes:

Filtering Parameters (1/2)

The Formulas panel as well as many Editor panels, in which you may use the parameters, allow you to filter parameters in order to ease their selection.

1. When the selection panel is opened, first select your selection mode: incremental or not.

2. Then select in the specification tree the feature that contains the parameters that you want to use.

With the incremental mode unchecked, ALL the parameters of the Groove and ALL those of its definition sketch are displayed.

Double click on a parameter to edit it

Parameter	Value	Formula	Active
PartBody\Assemble.10\Body.8\Groove.3\Sketch.7\Parallelism.94\A...	true		
PartBody\Assemble.10\Body.8\Groove.3\Sketch.7\Parallelism.94\m...	Constrained		
PartBody\Assemble.10\Body.8\Groove.3\Sketch.7\Fixed.95\Activity	true		
PartBody\Assemble.10\Body.8\Groove.3\Sketch.7\Fixed.95\mode	Constrained		
PartBody\Assemble.10\Body.8\Groove.3\ThickThin1	1mm		
PartBody\Assemble.10\Body.8\Groove.3\ThickThin2	0mm		
PartBody\Assemble.10\Body.8\Groove.3\Activity	true		

lots of parameters are displayed: activities, modes, etc.

With the incremental mode checked, the parameters of the Groove and ONLY the dimension parameters of its definition sketch are displayed.

Double click on a parameter to edit it

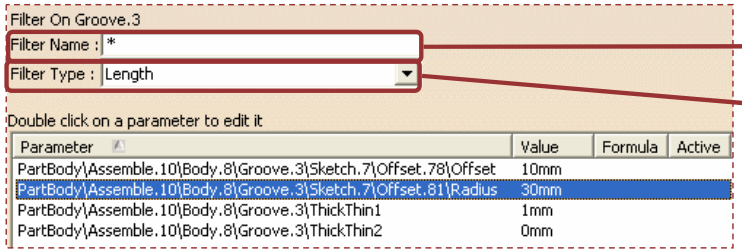
Parameter	Value	Formula	Active
PartBody\Assemble.10\Body.8\Groove.3\FirstAngle	360deg		
PartBody\Assemble.10\Body.8\Groove.3\SecondAngle	0deg		
PartBody\Assemble.10\Body.8\Groove.3\Sketch.7\Offset.78\Offset	10mm		
PartBody\Assemble.10\Body.8\Groove.3\Sketch.7\Offset.81\Radius	30mm		
PartBody\Assemble.10\Body.8\Groove.3\ThickThin1	1mm		
PartBody\Assemble.10\Body.8\Groove.3\ThickThin2	0mm		
PartBody\Assemble.10\Body.8\Groove.3\Activity	true		

fewer parameters are displayed: only 7 where found for Groove.3

Student Notes:

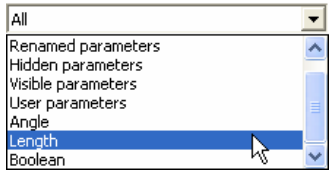
Filtering Parameters (2/2)

3 **If you still have too many parameters listed, you can use filters: you usually have the possibility to filter the parameters by types and by name.**

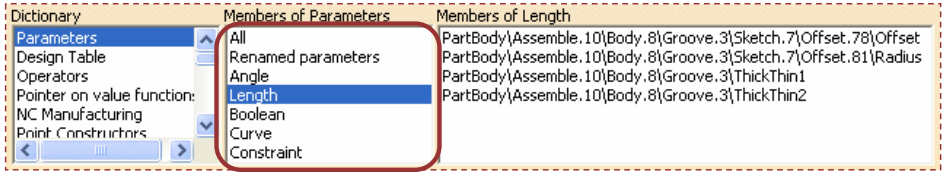


you can make a query per name

...or per type:



or...



select a type in the list above

Types available in the "Filter Type" list are the types of parameters found in the current selection.

4 **You should now be able to select a parameter easily.**

Student Notes:

Adding Sets of Parameters

In the specification tree, you can create sets of parameters under the Parameters node in order to regroup the parameters by categories.

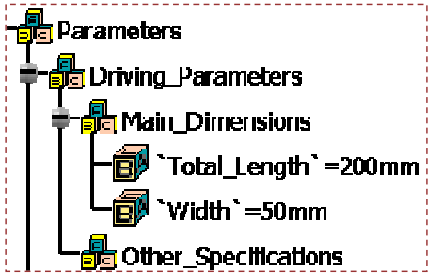
1. In the Knowledge Advisor workbench, click on **Add Set of Parameters** icon.

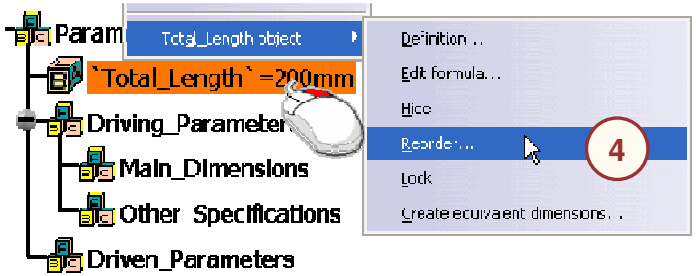
2. Select in the tree the Parameters node or an existing set of parameters under which the new Set of Parameters will be created.

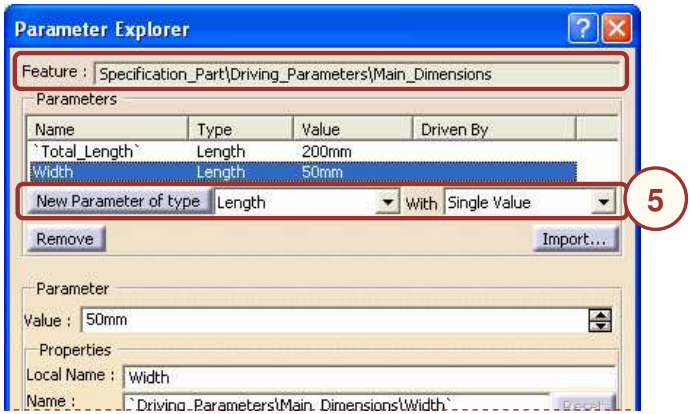
3. You can rename the Set of Parameters by editing its **Properties** (in the contextual menu).

4. You can reorder the already existing user parameters using the **Reorder** option of the parameter contextual menu. Select a Set of Parameters to place the parameter in it.

5. To create a new user parameter directly in a specific Set of Parameters, you have to use the **Parameter Explorer**. Select a Set of Parameters before clicking the New Parameter of type button.





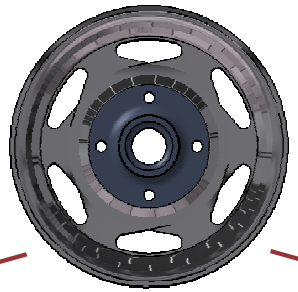


Student Notes:

Why Publish Parameters?

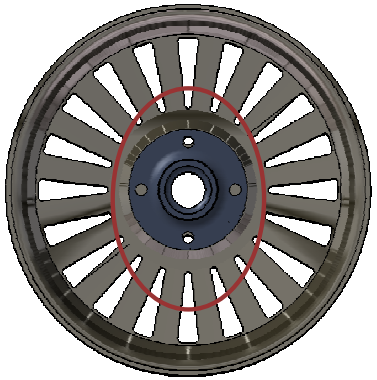
- **Publication of parameters is useful when replacing in an assembly a component which contains parameters that drive other component's external parameters.**
- **If the exported parameters are published and if the parameters of the replacing component are published under the same name, they will inherit the control of the exported parameters.**
- **Otherwise, the parameters of the replaced component will keep the control.**

In this example, the hub is linked to the rim: the hub reuses the number of holes and the pattern diameter of the rim.
Let us see the difference in the behavior of the hub when replacing the rim, with its parameters published or not.



The rim is replaced by a bigger one, the parameters of which are not published.

The rim is replaced by a bigger one, the parameters of which are published under the same names than the first rim.



The external parameters of the hub are still linked to the first rim. They are not updated.

The number of holes of the hub and the diameter of the pattern automatically adapt to the new rim.

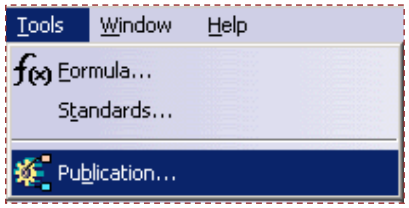
Student Notes:

Publishing a Parameter (1/3)

The Publication command is available in Assembly Design and Part Design.
It publishes the geometry and the parameters as well.

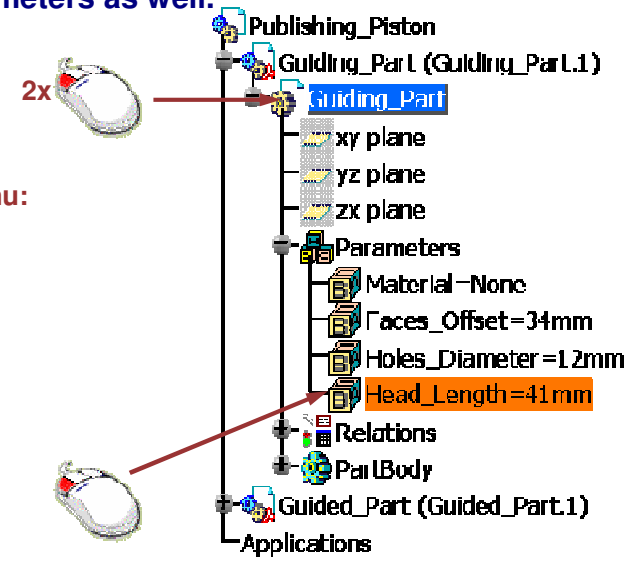
1. Activate the part containing the parameter you want to publish.

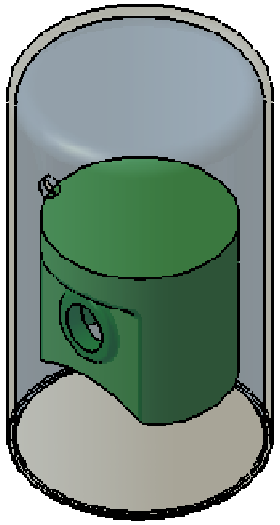
2. Select Publication… in the Tools menu:

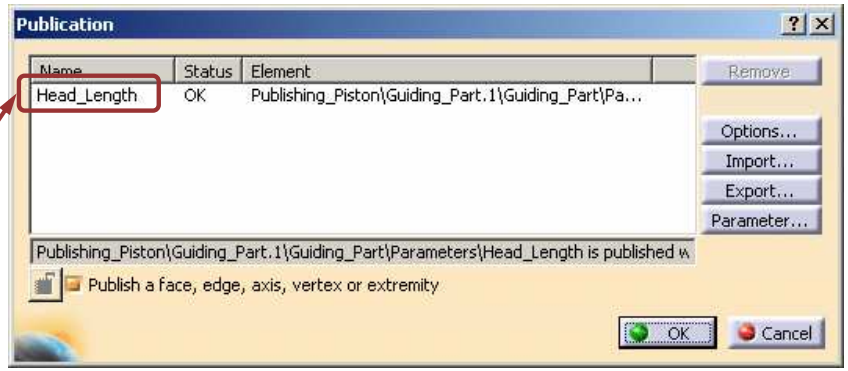


3a. If the parameter you want to publish is a user parameter, click on its icon in the tree.

4a. The user parameter now appears in the list of published elements of the Publication dialog box.

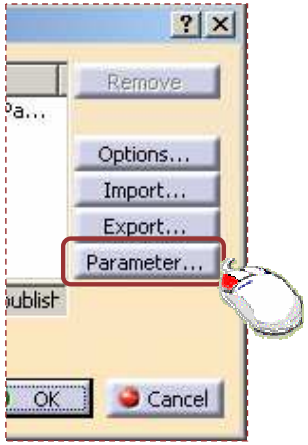






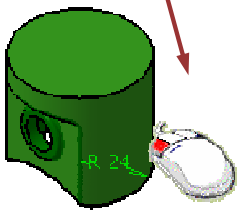
Student Notes:

Publishing a Parameter (2/3)



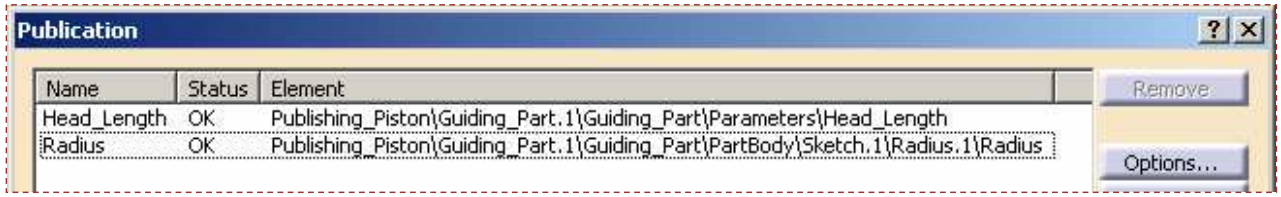
3b If the parameter you want to publish is an intrinsic parameter, click the Parameter button of the dialog box.

4b Select the parameter:
- directly in the dialog box
- or by the intermediate of the geometry



5b Click OK to validate the selection.

6b The intrinsic parameter appears in the list of published parameters:

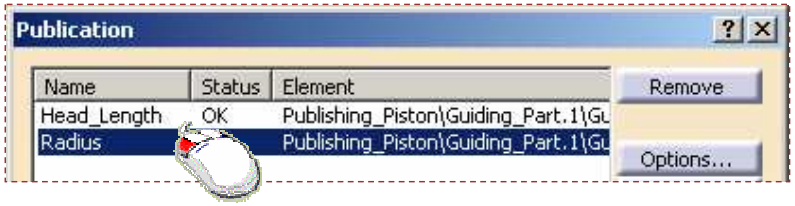


Student Notes:

Publishing a Parameter (3/3)

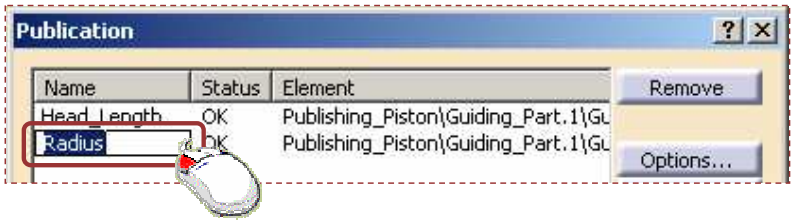
7 Published Parameters appear in the list with a default publication name.

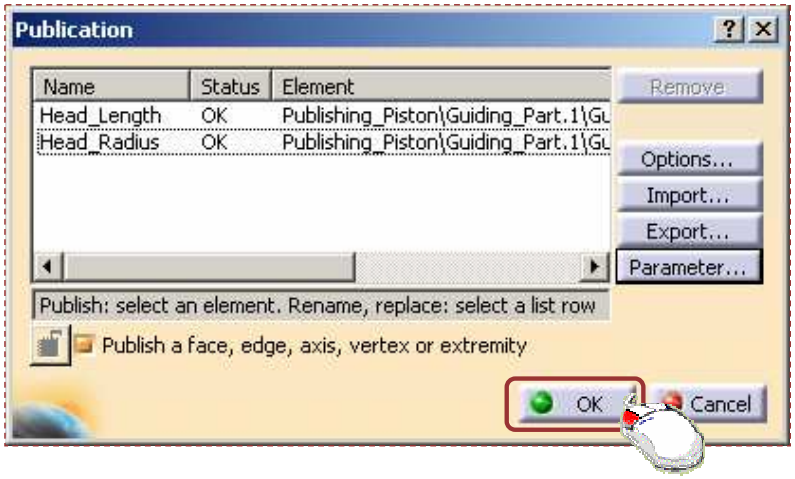
8 To modify the publication name, first select the publication.



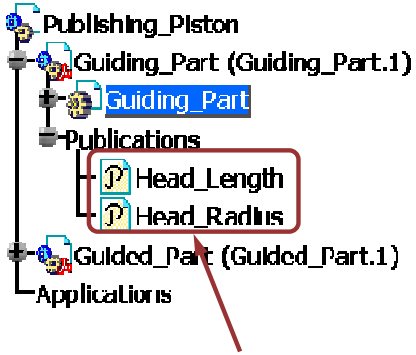
9 Then, select the name field.

10 Edit the name and validate with Enter.





11 Validate the publication by clicking OK.



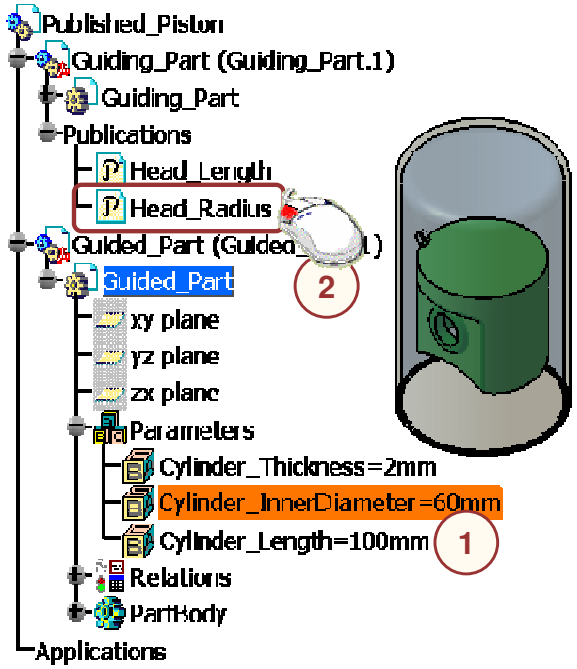
12 Your newly published parameters appear under the publications node of the active part.

Student Notes:

Using Published Parameters (1/5)

Published parameters are called while editing formulas.

In this example, we are going to make equal the inner cylinder diameter to the head diameter.



1. Be activated on Guided_Part level and open the formula editor panel of Cylinder_InnerDiameter parameter.

2. Edit the formula by selecting the Head_Radius parameter:
   - Under the Publications' node of Guiding_Part
   - In the External Parameters of Guided_Part, provided that it has previously been copied with link. The copy with link is already made if you have used this external parameter before, or if you have intentionally copied/pasted it Special as result with link.

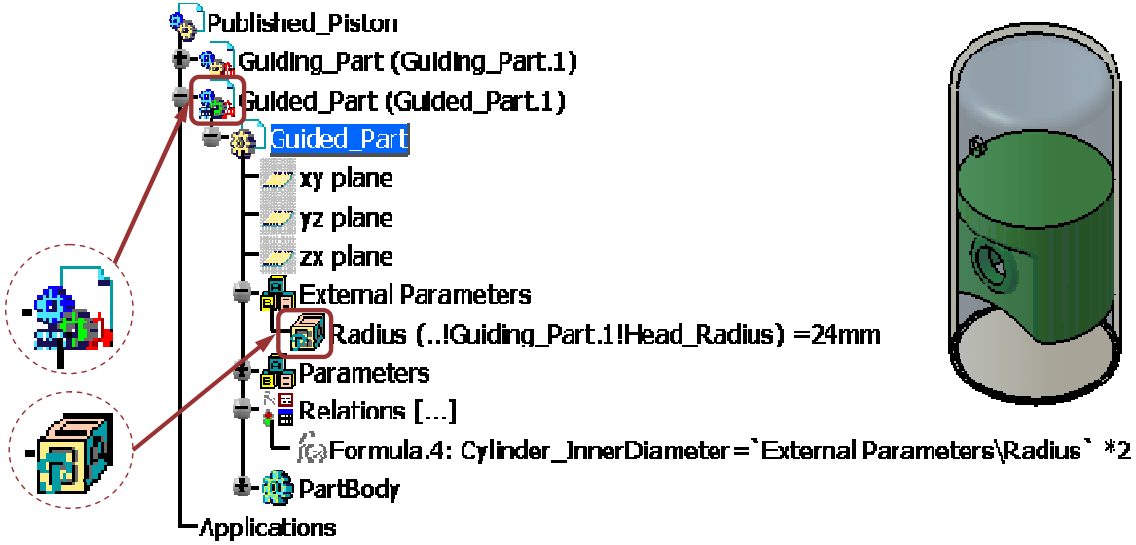
Student Notes:

Using Published Parameters (2/5)

Published parameters are called while editing formulas.

3a The edited part has become contextual.

3b The External Parameters linked to the published parameters appear with a green Capital P on their icon in the tree.

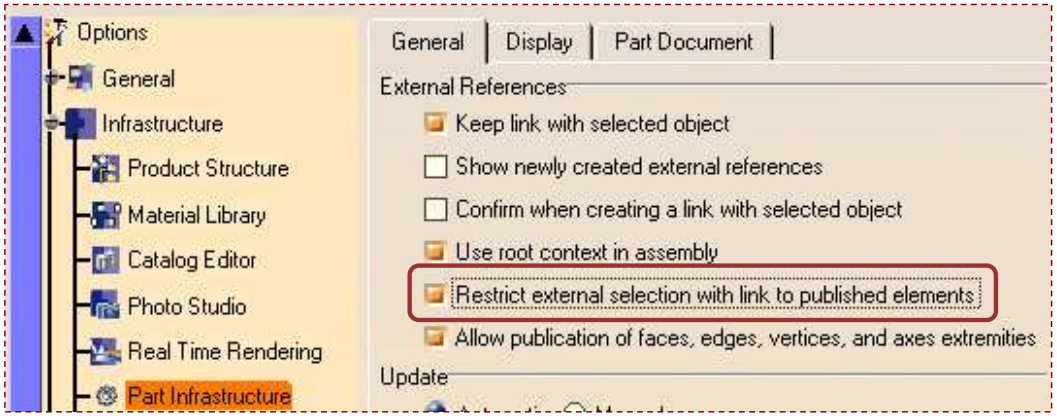


Student Notes:

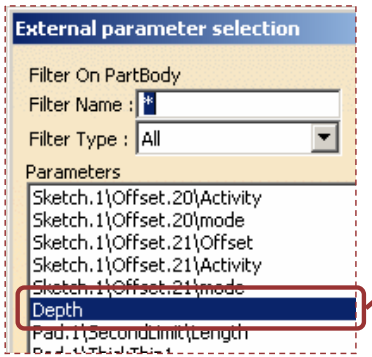
Using Published Parameters (3/5)

Some CATIA options can prevent the user from creating external parameters from the unpublished parameters.

1 **The setting preventing the use of non published geometry also works with parameters.**



2 **When this option "Restrict External selection…" is activated, and when you select an unpublished parameter in an external document, no external parameter is created and no link is kept: only the value of the parameter will be taken (as if the option 'Keep link…' was deactivated).**

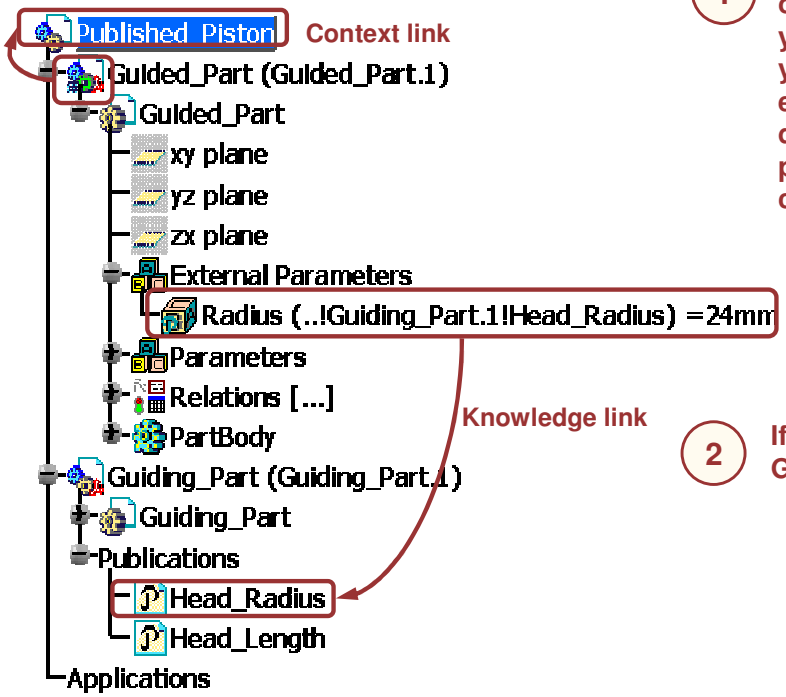


In this case, the depth parameter of GuidingPart was not published and only its value (52mm) has been taken to edit this formula. Neither link nor external parameter are created.

Student Notes:

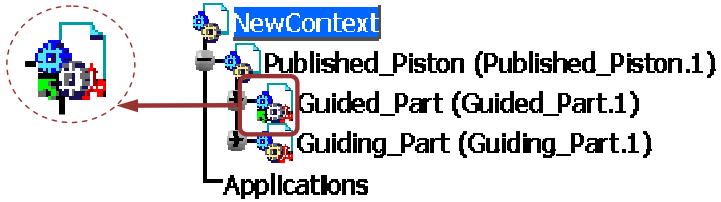
Using Published Parameters (4/5)

While using the published parameters you have to pay attention to the context assembly.



1 The first time you use an external reference or a published external parameter, not only you create links to external information, but you also define a "context" link from the edited part to the root assembly (by default). The context link is unique and the product it is connected to is called the context assembly.

2 If the root product is not anymore the context product of Guided_Part, its icon indicates it is out of context.

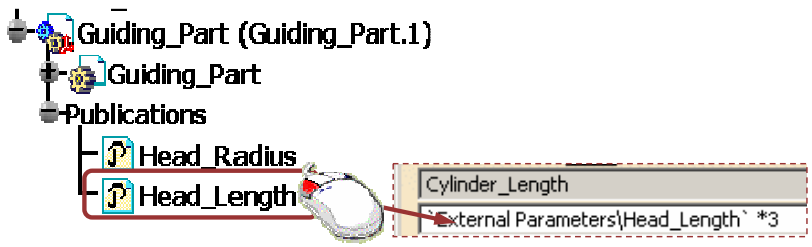


Student Notes:

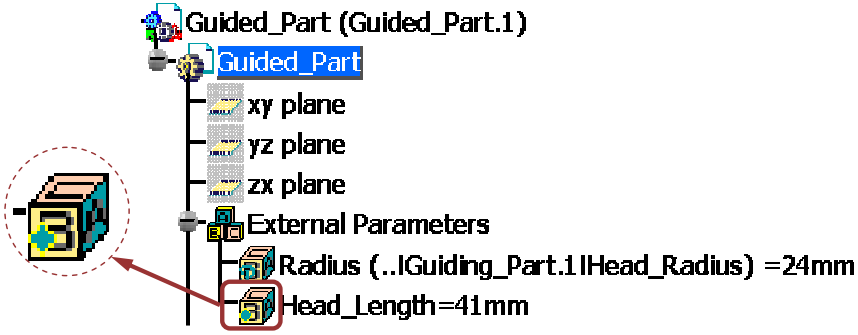
Using Published Parameters (5/5)

While using the published parameters you have to pay attention to the context assembly.

3 In this new context, try to create, in the Guided_Part, a new formula referring to another published parameter of the Guiding_Part.



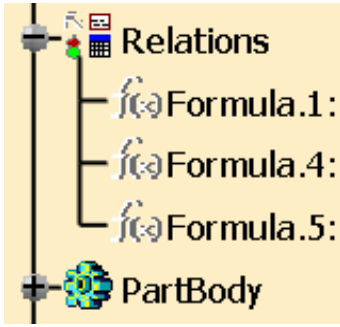
4 An external parameter which is created when the root product is not the context product will never be considered as connected to a published parameter.

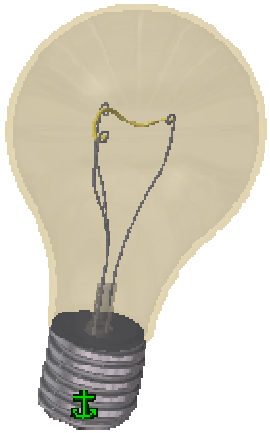


Student Notes:

Creating and Using Formulas

You will learn how to create and use Formulas.





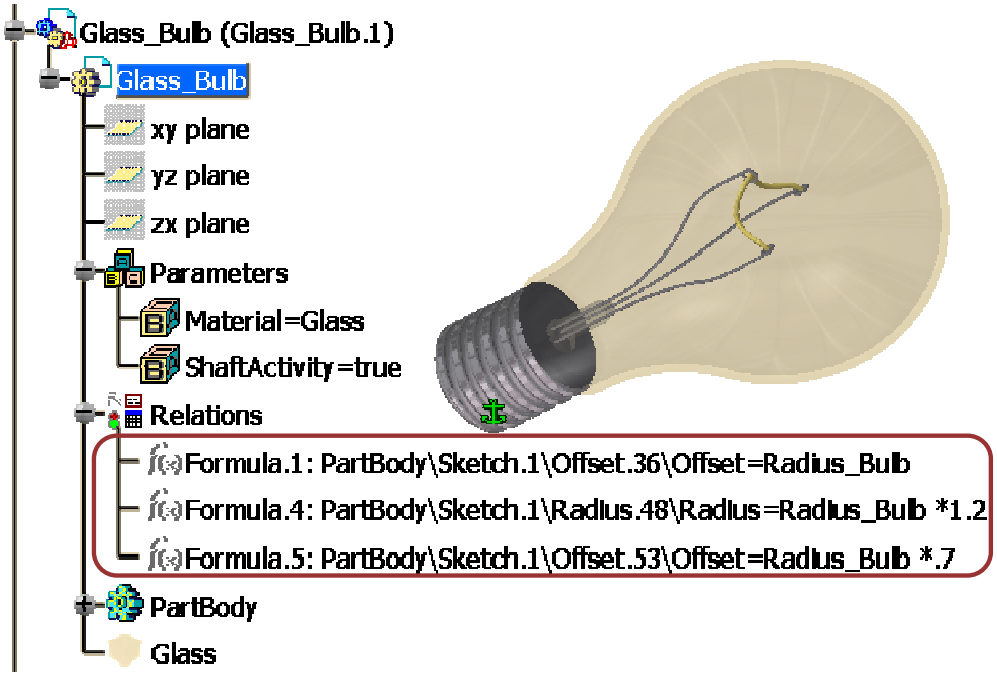
Student Notes:

What are Formulas?

- Formulas are relations used to define or constrain any parameter.
- Formula can be defined with parameters, operators, and functions.
- A Formula is created from the moment you attribute a user parameter to a feature, for example.
- The left part of the relation is the parameter to constrain and the right part is a statement.

Formula.4: PartBody\Sketch.1\Radius.48\Radius=Radius_Bulb *1.2

- Once it has been created, a Formula can be manipulated like any other feature from its contextual menu.



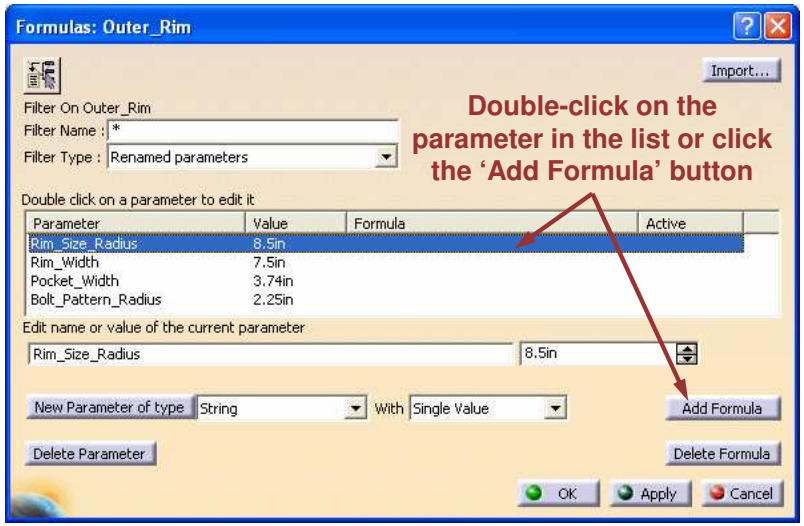
Student Notes:

Creating a Formula (1/2)

You can create Formulas with 'dimensions' or User Parameters.

1 **You can access the Formula Editor through different means:**

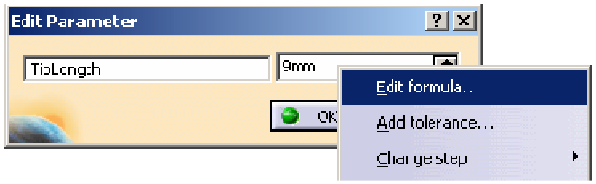
- **Click on the f(x) icon; in the Formulas panel, use the filter to select the parameter you want to edit. Either double-click on this parameter or click the Add Formula button.**



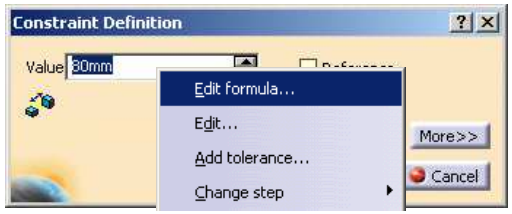
Double-click on the parameter in the list or click the 'Add Formula' button

OR

- **In the specification tree, double-click on the parameter or on the dimension you want to add a formula to. Right-click in the Value field and select 'Edit formula' in the contextual menu.**



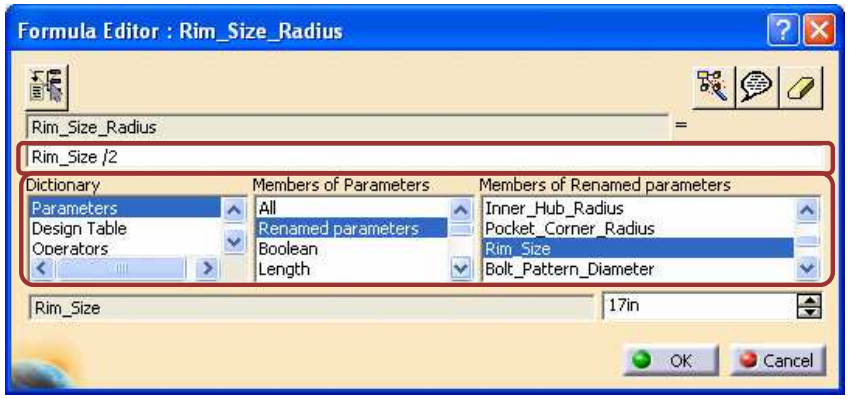
or…



Student Notes:

Creating a Formula (2/2)

2 The Formula Editor panel appears.
Enter the right side of the formula in the formula editor field.



Enter the formula here

Use the dictionary to select a parameter or a function

Check the Incremental mode button in order to display in the dictionary only the parameters of the feature selected in the specifications tree or in the 3D. If this option is not checked, will be displayed not only the parameters of the selected feature but also those of the features under it.

Click to open the language browser panel (see specific slides).

Click to attach a URL or a comment to the formula.

Click on the Eraser to delete all the contents of the formula field.

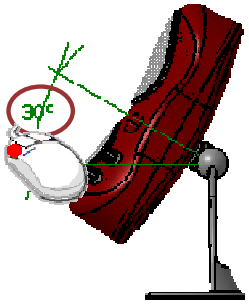
3 Click OK to validate the creation of the formula.
The Formula is added to the Relations node in the specification tree.

Student Notes:

Selecting Parameters in the Formula Editor

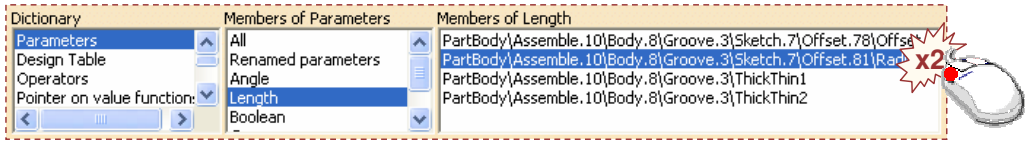
While creating the parametric models you often have to select a parameter to use it in a statement, in a design table, or simply to edit it. Here are different ways of selection.

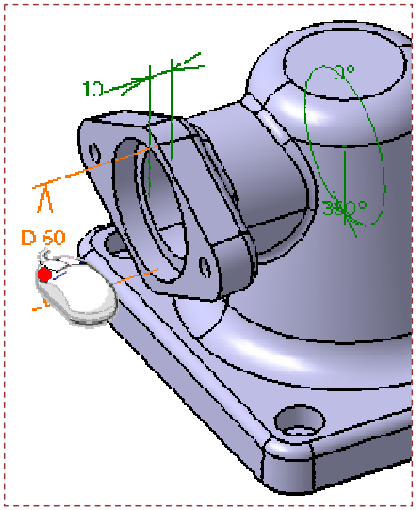
A If the parameter is displayed in the specification tree click on it.



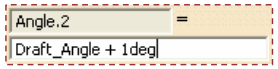
B If the parameter is displayed in the 3D (assembly constraint for instance) you can also click on it in the 3D.

C If you are using the Parameters Dictionary, you can either double-click on it in the list or click once on it in the 3D.





D If you know the exact name of the parameter you can also type it.



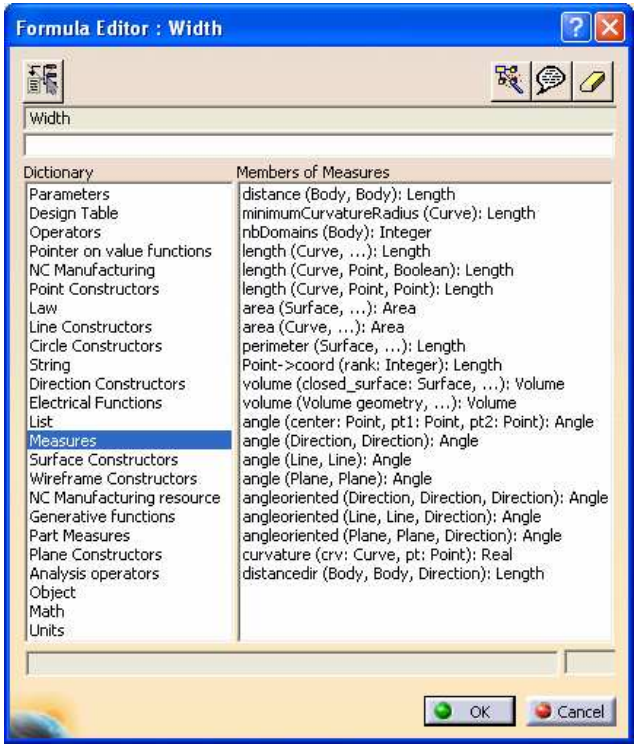
Student Notes:

Using Measure Functions in Formulas (1/3)

When you are editing a formula, you have the possibility to use pre-defined functions, especially measures. The functions allow you to capture values from the geometry.

For instance, the functions of the Measures dictionary allow you to define a parameter as:

- **A distance between two points**
- **The minimum radius of a curve**
- **The total length of a curve**
- **The length of a curve segment**
- **The area of a surface or a sketch**
- **The perimeter of a surface**
- **The volume of a PartBody or a closed surface**
- **An angle, oriented or not, between two lines, directions, or planes**



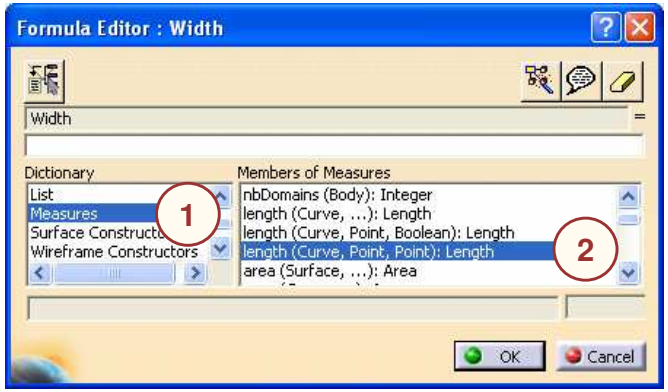
To make sure that you have access to all these functions, check that the Load extended language libraries option is selected in the Knowledge tab of General settings (Tools>Options).

Student Notes:

Using Measure Functions in Formulas (2/3)

1. **In the Formula Editor panel, select the Measures item from the dictionary list.**

2. **The list of measures functions appears. Select for example the *length(Curve,Point,Boolean)* item by double-clicking on it.**



Student Notes:

Using Measure Functions in Formulas (3/3)

3 The length function is added to the Formula Editor.

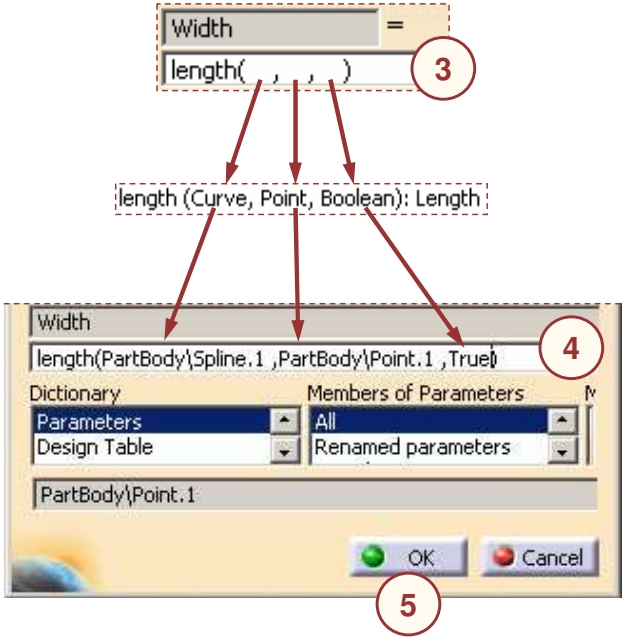
4 You now need to fill the arguments of the function. The function description informs you of the nature of the arguments.

For each argument, check that the cursor is positioned where the argument is intended to be typed, and then select the corresponding feature in the tree.

Of course, when the argument is an Integer or a Boolean, you can just type it. In our example, the third argument is a Boolean: type 'True' if the length is to be calculated from the origin, and 'False' if the length is to be calculated from the curve end.

5 Validate by clicking OK.

6 CATIA may ask you if you want the relation to be updated automatically with global update. We advise you to answer 'Yes'.



Student Notes:

Using the Language Browser

1. Open the Language browser panel by clicking on the following Action button in a Relation Editor.

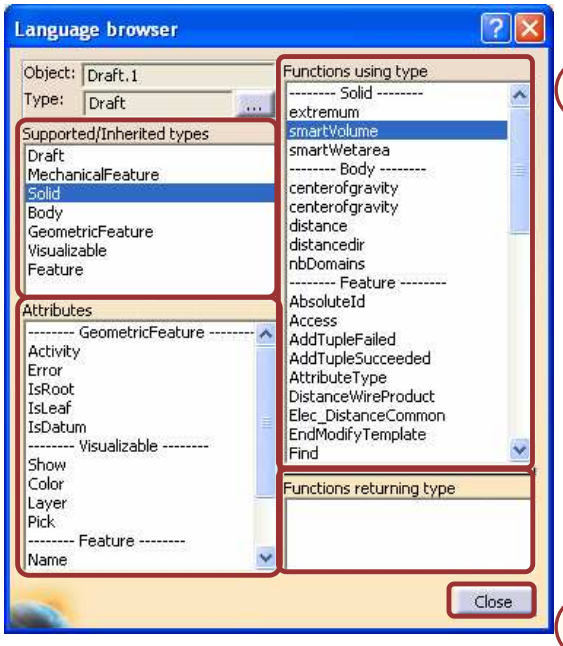
2. Select a feature in the tree or in the geometry. Its type is indicated in the Type field. You can also choose a type manually using the … button.

3. The **Supported/Inherited types** field provides you with a list of the types supported by the selected type, and of the types that the selected type inherits from.
Double-click on the type to have it automatically declared in your relation.

Let Solid1 (Solid)

4. The **Attributes** field lists the possible attributes of the selected type, and of the supported and inherited types.
Double-click on an Attribute to have it filled in your relation.

.Activity



5. The **Functions using type** field lists the functions and methods whose first argument is a type of the Supported/Inherited types list.
The **Functions returning type** field lists the functions and methods returning the selected type.
Double-click on a function to have it added to your relation.

smartVolume()

6. Click Close to close the panel.

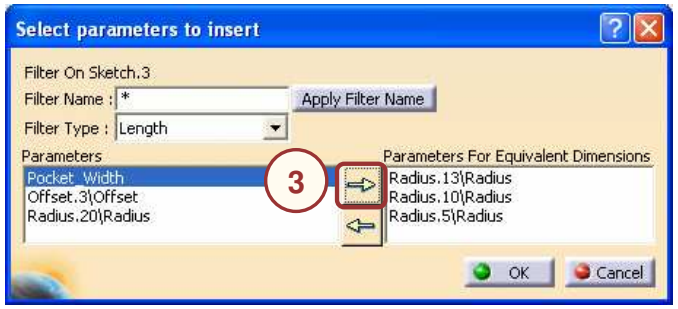
Student Notes:

Equivalent Dimensions Feature

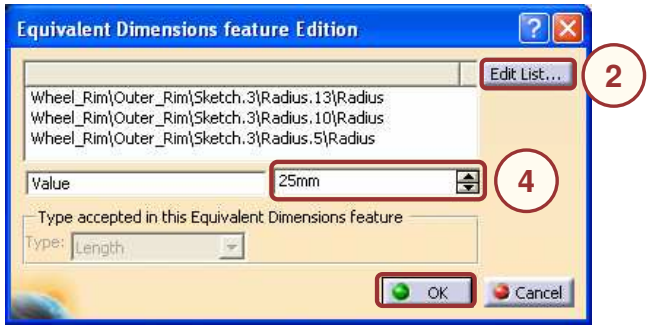
The Equivalent Dimensions feature helps you to define an equality between a set of Angle or Length parameters. Its value can be modified through the editor and is propagated to all the parameters belonging to the equivalence.
This feature increases the designer's productivity and also decreases the model size.

1. Click on the Equivalent Dimensions icon in the common Knowledge Toolbar. The Equivalent Dimensions Edition window displays.

2. Click the Edit List… button. A panel displays for you to select the equivalent parameters.

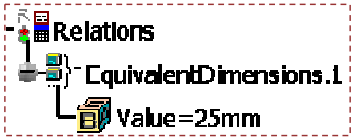


3. Select in the list the parameters that will have the same value and use the right arrow button to add them to the Equivalent Dimension feature. Click OK when all the parameters are selected.



4. Back in the Equivalent Dimensions Edition panel, check the value of the equality before validating by clicking OK.

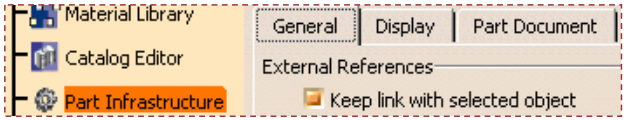
5. The Equivalent Dimensions feature is displayed in the Relations node.
Double-click on it to view the list of parameters, modify it or change the value.



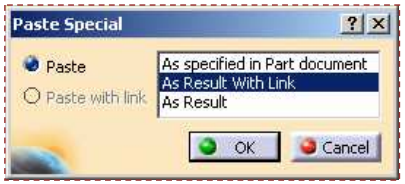
Student Notes:

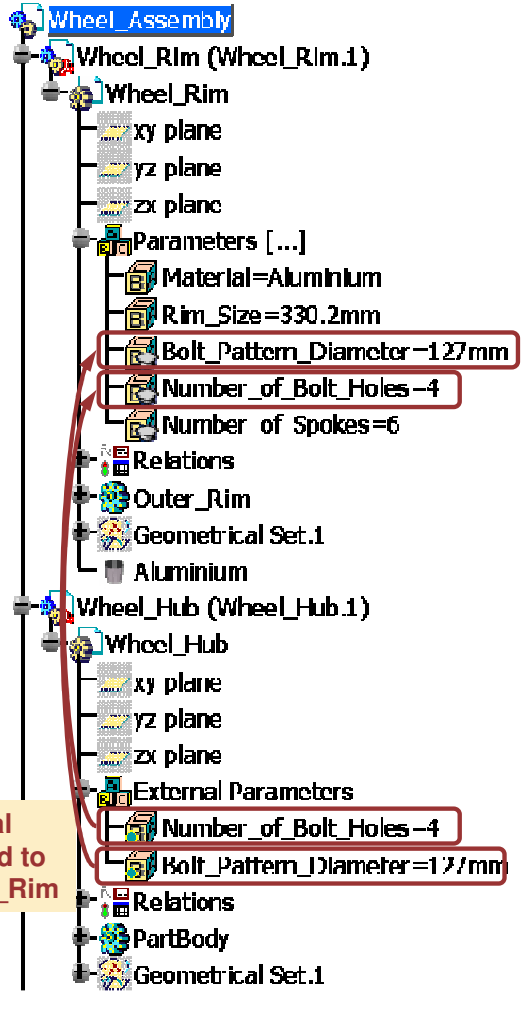
What is an External Parameter?

- External Parameters are linked copies of parameters driven in an external document.
- It is possible to create them provided that the 'Keep Link with selected object' in the Tools > Option menu is activated.



- They can be created:
  - Automatically by referring to another part's parameter in a relation
  - Manually by using the Copy/ Paste Special – As Result With Link command





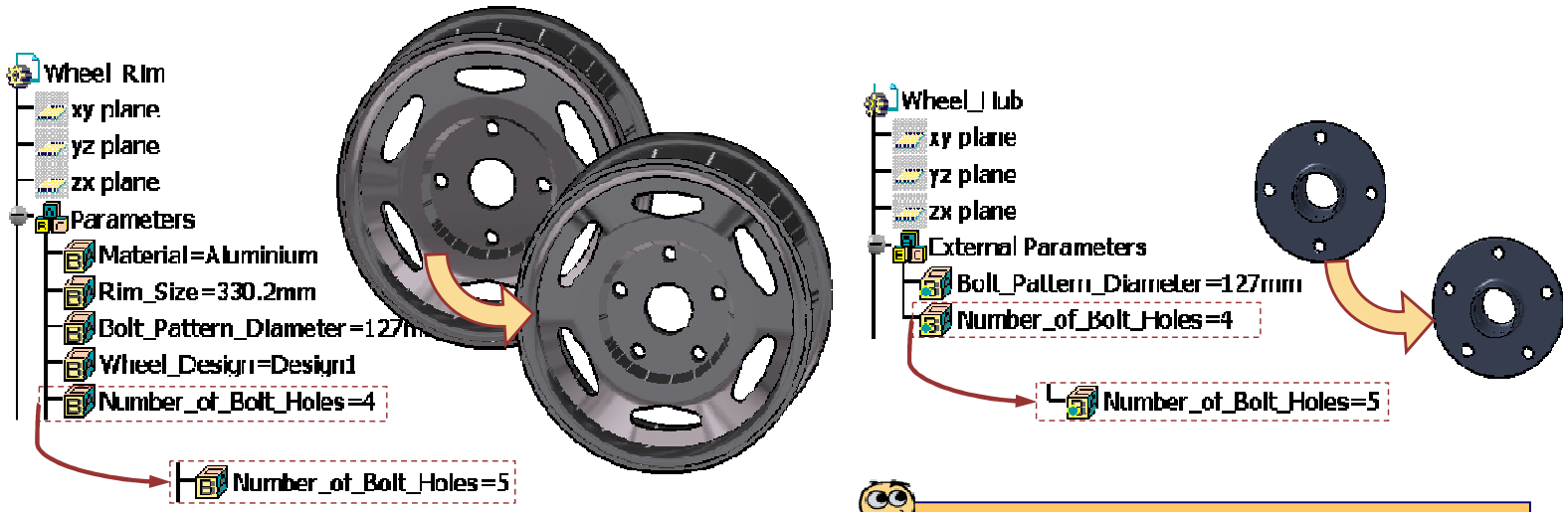
These two External Parameters are linked to their fathers in Wheel_Rim

Student Notes:

Why Use External Parameters?

- To reuse a parameter that drives a Part into another Part in order to link their geometries.
- To be sure that the design of the two linked parts is consistent.
- To avoid manual update of all the parameters that must have the same value in different parts.

In this example, the hub needs to adapt to the holes of the rims. External parameters have been created in order to link the number of holes and the bolt pattern diameter.



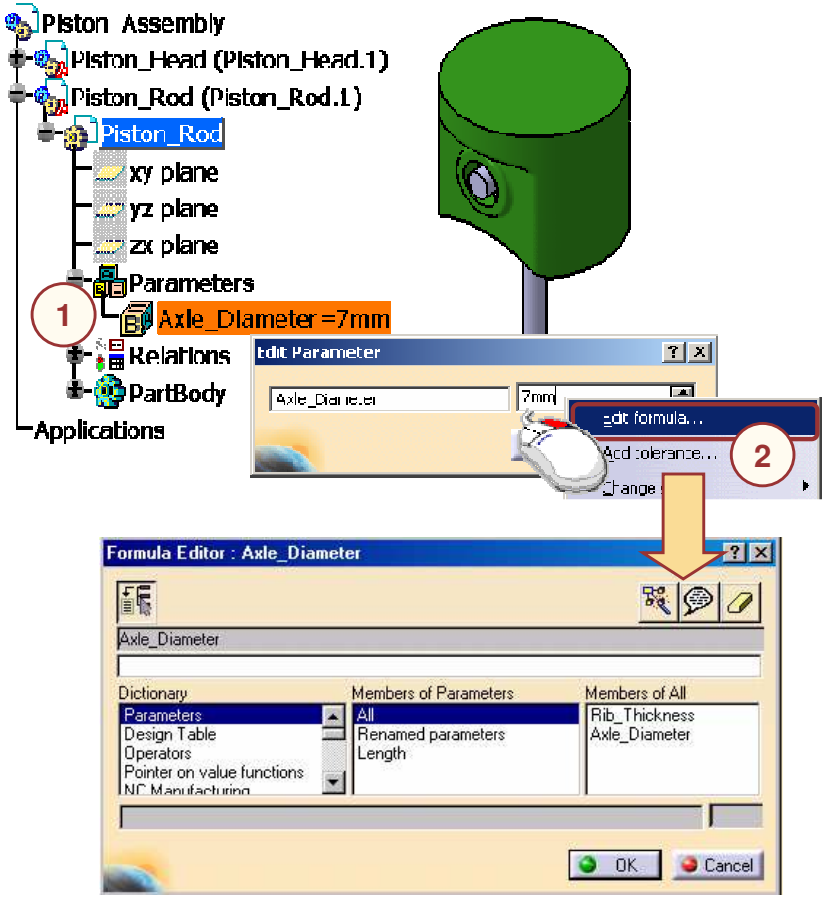
Here the Number_of_Bolt_Holes parameter has been copied with link from Wheel_Rim.CATPart to Wheel_Hub.CATPart.

Student Notes:

Referring to External Parameters in Formulas (1/2)

In a Formula, you can use the parameters defined in the external documents.
This is possible between any type of document.

The following Assembly contains two Parts.



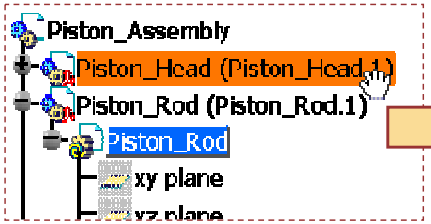
1. In the specification tree, double-click on the user parameter Axle_Diameter in order to edit it.

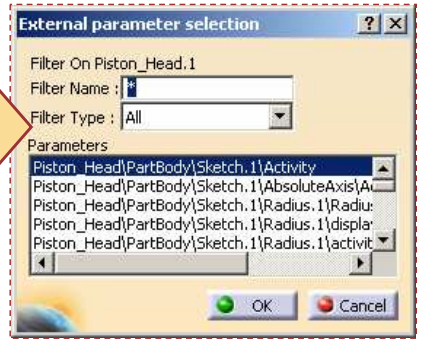
2. In the contextual menu of the parameter's value, select the *Edit formula* option.
The Formula Editor panel is displayed.

Student Notes:

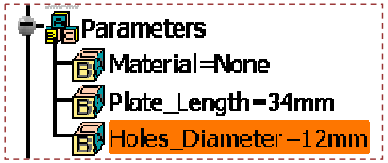
Referring to External Parameters in Formulas (2/2)

3 Select the second instance (Piston_Head).
The *External parameter selection* panel is displayed.



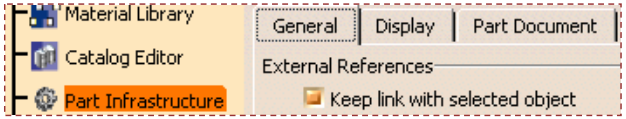


Remark:
The *External parameter selection* panel is mainly used to select intrinsic parameters. In the case of user parameters, it is possible to directly select the parameter in the tree.

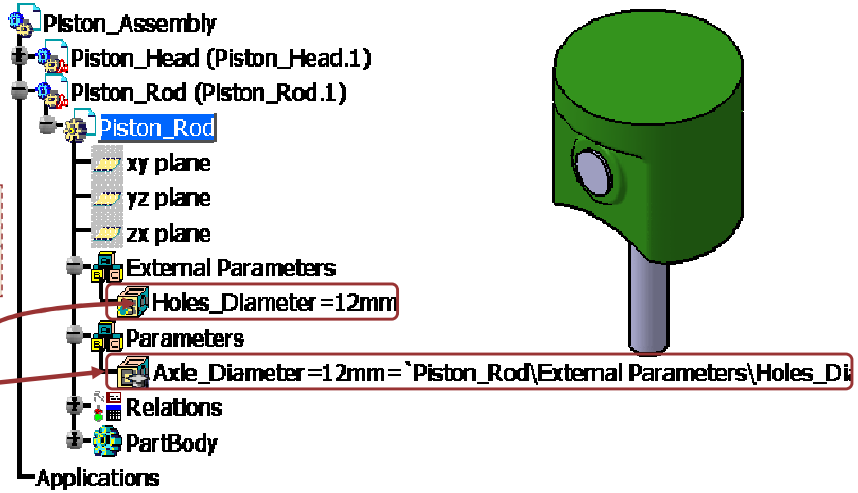


4 Select in the tree the user parameter Holes_Diameter.
Validate by clicking OK in the *External parameter selection*, in *Formula editor*, and in the *Edit Parameter* dialog box.

5 Provided this option was activated,



an external parameter has been created in the Piston_Rod.CATPart and is used in the newly created formula.



Student Notes:

Creating Lists

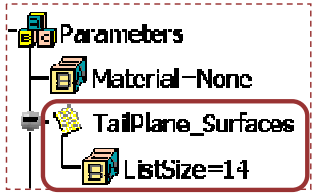
You will learn how to create lists. List features can be used to manage lists of objects or parameters.

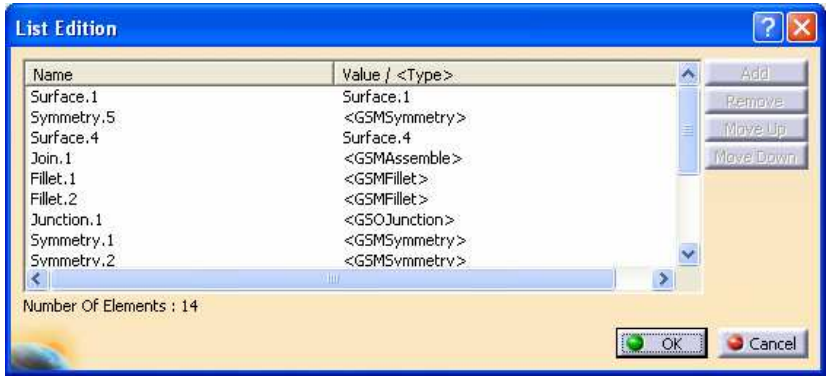
Student Notes:

What are Lists?

- The Knowledgeware List features are lists of ordered features or parameters.
- A list can be populated either automatically or manually.
- The items of a list can be reordered either manually or throughout functions.
- The list features are stored under the Parameters node of the specification tree and are integrated in the update mechanism.
- A ListSize integer parameter indicates the number of items that populate the list. It is computed automatically.

- Lists can be used:
  - To make a sum of parameters easily
  - To count the number of features of a given type in a document and then to calculate a cost
  - To create loops in reactions features or in loops features





Student Notes:

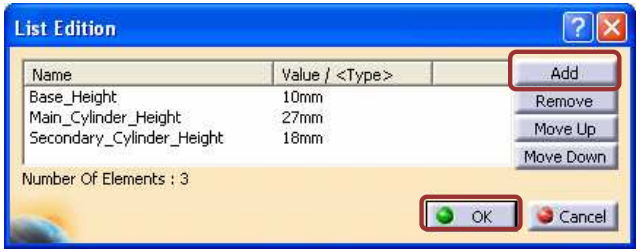
Creating a List

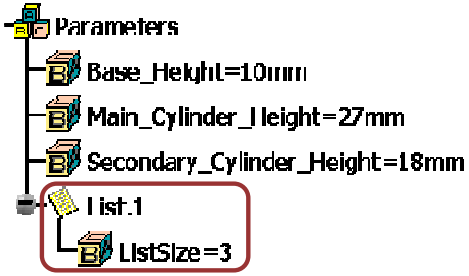
1. In the Knowledge Advisor Workbench, select the List icon.
   The List Edition panel appears.

2. Select some parameters or features in the tree and click the Add button to add them to the list.

3. Validate List creation by clicking OK.

4. The List appears under the Parameters node in the tree, and a ListSize parameter is automatically created and indicates the number of items in the list.
   You can rename the List using its Properties.





The list feature can be manipulated through specific functions to:
- **Add and remove elements to the list**
- **Get an element**
- **Retrieve values from the list**
- **Move elements of the list to another position**
- **Copy the content of a list into another one**

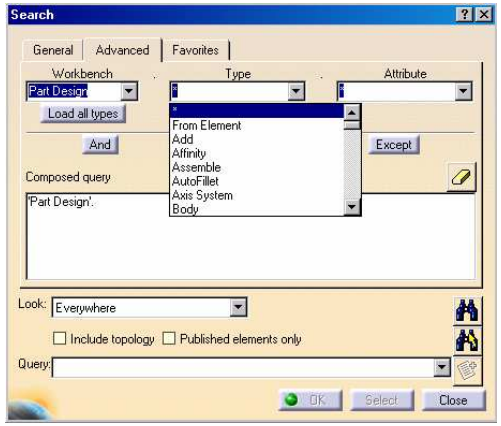
Student Notes:

What is Populating a List Using a Query?

- **Using the Query function you can automatically populate a List whith features that verify a specified expression.**
- **In the example below, the result of the search will return the holes of the PartBody whose diameters are greater than 10mm:**

  Example: List.1=PartBody.Query("Hole","x.Diameter>10mm")
  Where:
  - **List.1 is the name of the list on which the calculation will be performed**
  - **PartBody is the body on which the search will be carried out**
  - **Hole is the Type of the searched feature**
  - **x.Diameter>50mm is the expression (optional). If no expression is to be verified, just write PartBody.Query("Hole","")**

- **To know the possible feature types and attributes that you can use in the Query function, use the Edit/Search command.**



Student Notes:

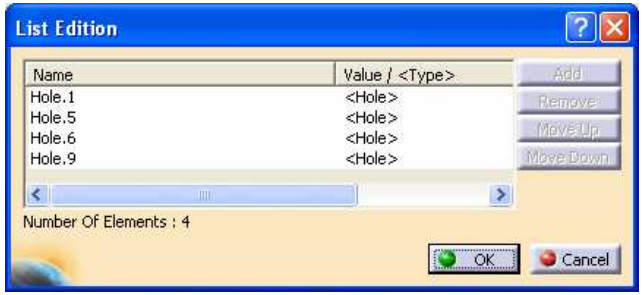
Populating a List Using a Query

1. Create an empty List: click on the List icon and click OK without adding any item to the list.

2. Open the Formula Editor. Select the new List in the tree and click the Add Formula button.
The formula editor panel is displayed.

3. Enter the right side of the formula in the formula editor field. For instance:
PartBody.Query("Hole","x.Diameter>5mm")
In this case, the List will be populated by all the holes of diameter greater than 5mm.

4. Click OK to validate the formula creation and close Formula panel.



5. The List is automatically populated with the holes of diameter greater than 5mm.

Student Notes:

Associating URLs to Parameters and Relations

You will learn how to create and find URLs attached to parameters and relations.

Student Notes:

Adding URLs

You can associate one or more URLs with user parameters and relations. This task is only meaningful when the active document contains user parameters and/or relations.

1. Select the Comment & URLs icon in the Knowledge Advisor workbench.

2. In the specification tree, select any parameter or relation (formula, rule, check, etc) to which the URL will be added. Then click the Add button.
The Add URL dialog box is displayed.

3. Enter a name for the URL and the link to it. It may be, for instance, an Internet address or a path to a document. Click OK to validate the creation of the URL.

4. Back in the main edition window, you can also add a comment to the parameter or relation.
Click OK to exit the panel.
The URL and the comment are added to the selected feature.

URLs can also be added to relations at their creation or edition.
To each parameter or relation can be added several URLs but only one comment.

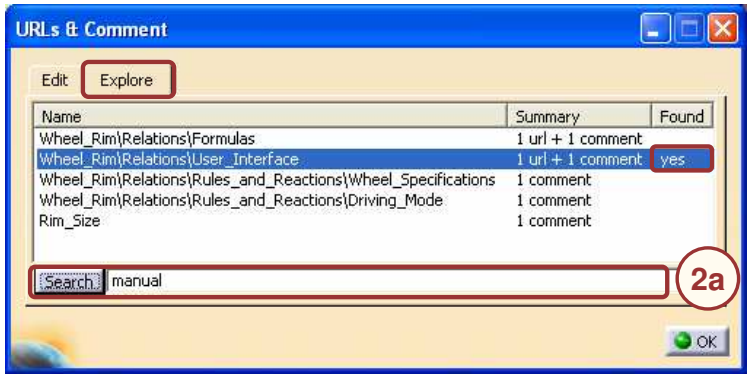
Student Notes:

Searching for URLs

1 Click on the Comment &URLs icon.
The URLs & Comment dialog box opens.

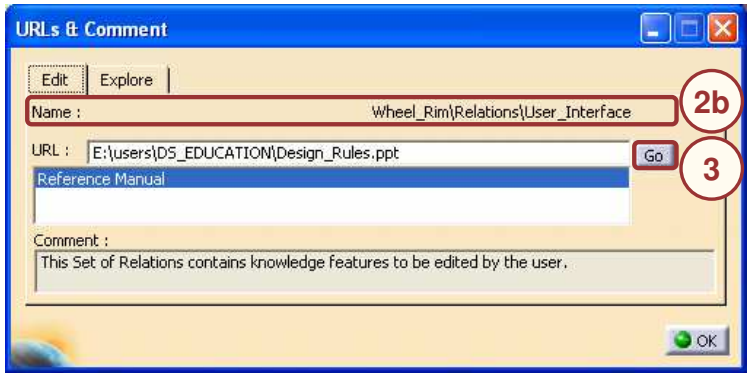
2a **From the Explore tab panel:**
Enter the name of the searched URL and click the Search button.
If the specified URL is found, "yes" is displayed in the Found column. Then return to the Edit Tab.

OR:



2b **From the Edit tab panel:**
Select a parameter or a relation in the specification tree : the URLs and the comments of the object are displayed.

3 In the Edit tab, the URL which has been found is highlighted.
Click the Go button to display the page or document related to this URL.



Student Notes:

Creating Adaptive Behaviors

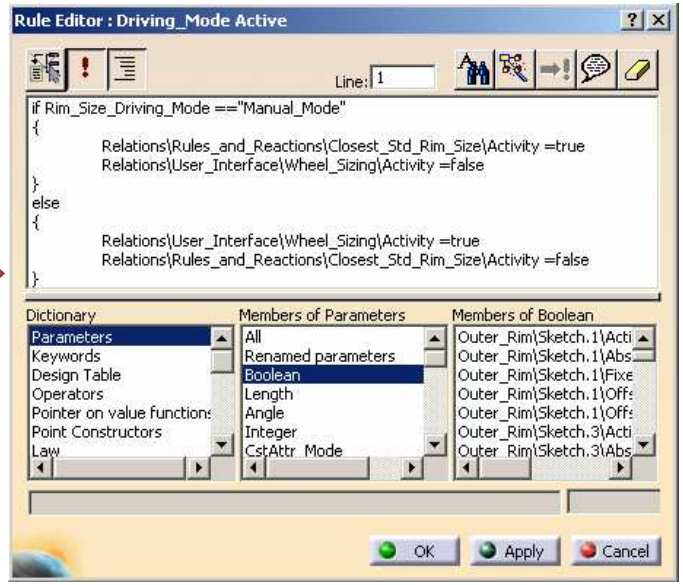
In this lesson you will learn how to create design alternatives and conditional geometries. Besides you will learn how to create self reactive designs using the Reaction feature.

- **Creating Rules**
- **Creating Checks**
- **Creating Reactions**

Student Notes:

Creating Rules

You will learn how to create and use the Rules feature.

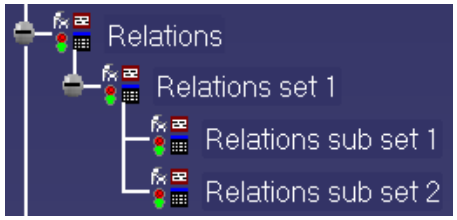


Student Notes:

Adding Sets of Relations

You can create sets of relations below the Relations node of the specification tree. Using this capability enables you to regroup the relations into categories. Formulas, design tables, rules and checks can all be created into relation sets. When no relation set has been created, the destination field of the relation editor is by default the main Relations node.

1. **To create sets and sub-sets of relations, click on the « Add Set of Relations » icon and select the Relations node in which the new set will be created. Eventually, rename the Relations sets using their Properties command (MB3).**



2. **While creating a new Relation (Check, Rule, etc), select the desired Relation set to store your new Relation.**

Student Notes:

What is a Rule?

- **A Rule is a set of instructions, generally based on conditional statements, whereby the relationship between the parameters is controlled.**
- **A Rule appears in the Relations node of the current document:**

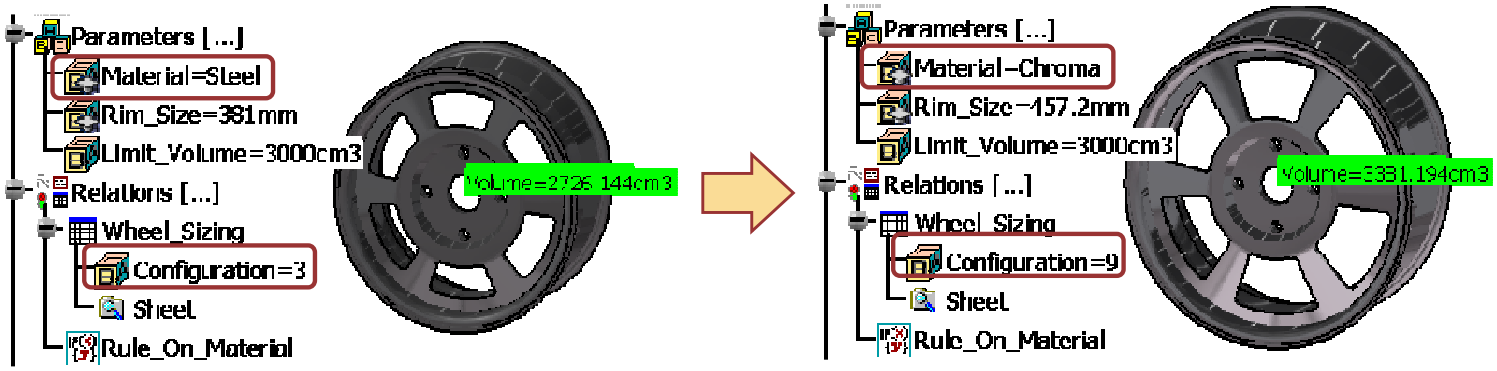
In the example below, the rule calculates the volume of the PartBody and sets the Material parameter in consequence with the result:

```
if smartVolume(PartBody)< Limit_Volume
{Material="Steel"}
```

if the volume of the PartBody is less than a limit value (here 3000cm3), the Material is set to Steel

```
else
Material="Chroma"
```

Otherwise, it is set it to Chroma



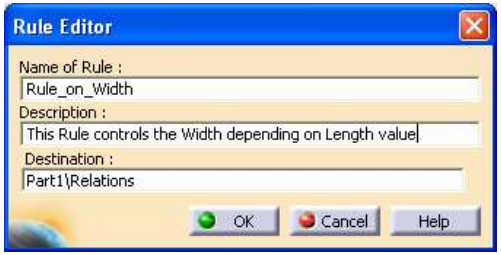
Here, we have changed the wheel's size by changing the configuration of the design table: the volume of the wheel has changed and its material has been updated automatically.

Student Notes:

Creating a Rule

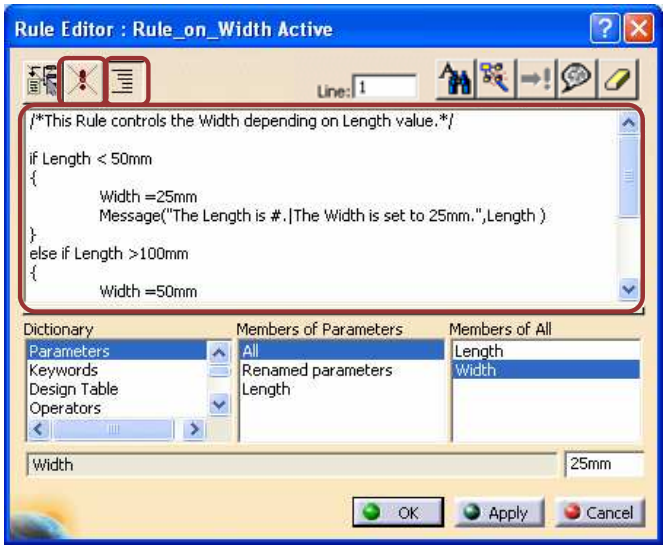
1. Open the Knowledge Advisor workbench and click on the Rule icon.

2. Enter the rule name and comments. You can also choose the relation set to which the Rule will be added.
Click OK.

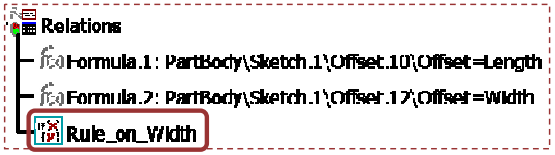


3. The Rule Editor panel is displayed.
Enter the body of the Rule:
- Check the Alignment button to have an automatic text formatting.
- Write your comments between the "/*" and "*/" signs.
- Use the Dictionary to help you select the parameters and the functions.

4. Check the (!) button to have the syntax of your rule verified interactively.
You can also click the Apply button when you have finished scripting the rule to check its syntax.
Click OK to validate the Rule creation.



5. Rule feature is displayed in the tree under the selected Relations node/set.



Student Notes:

Using the Rule/Check/Reaction Editor Interface

The edition panel of the Knowledge Advisor Reactive Features present a few buttons intended to help the user to write the body of the relation.

Type here the feature body.

Use the Dictionary to select the parameters and the functions.

Here is a preview of the latest selected parameter and of its actual value.

Check this button to activate the incremental mode: when you select a feature in the specification tree or in the geometry area, only the first level of features right below the selected feature will be displayed in the editor, which is very useful while working with large models.

Check this button to have a dynamic verification of the body syntax.

Check this button to have the text automatically formatted and indented.

Click this button to open the Language Browser panel.

In case of syntax errors, click this button to highlight the errors.

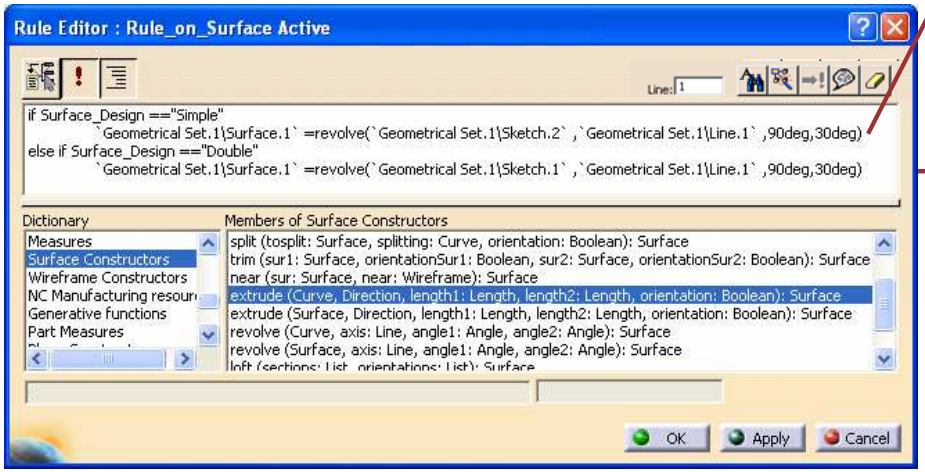
Click this button to add a URL to the relation or to change its comment.

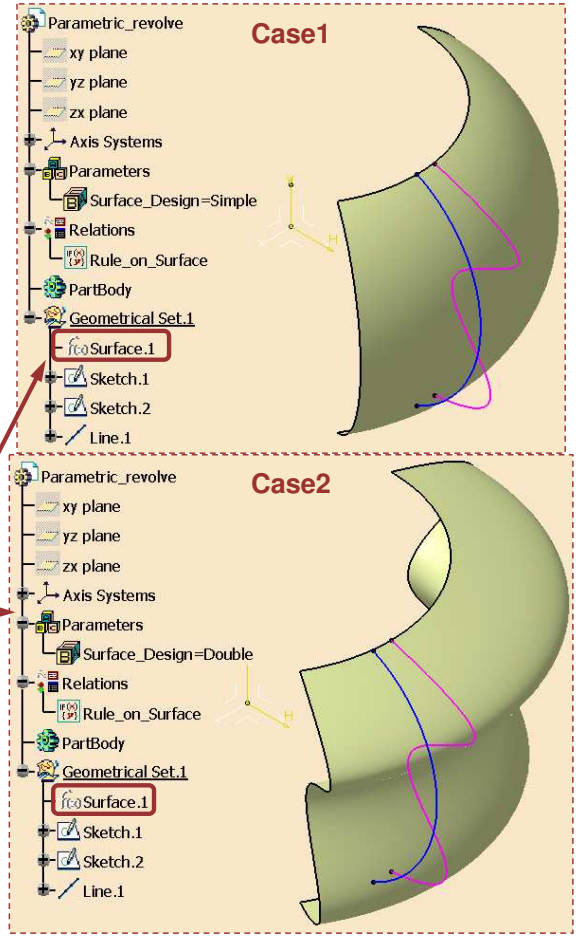
Click the Eraser button to clear the contents of the body field.

Student Notes:

What is Creating Geometry from Rules?

- **In order to create more adaptative designs, it is sometimes useful to create geometric elements from Rules. To do so, you will use the geometrical operators available in the functions dictionary.**
- **The following geometric elements can be created:**
  - **Point**
  - **Plane**
  - **Surface**
  - **Line**
  - **Curve**
  - **Circle**



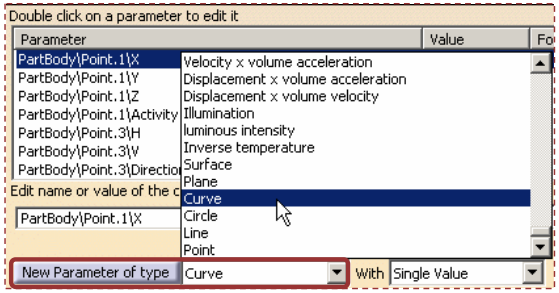


Student Notes:

Creating Geometry from Rules

1. **Click on the F(x) icon to open the formula editor.**

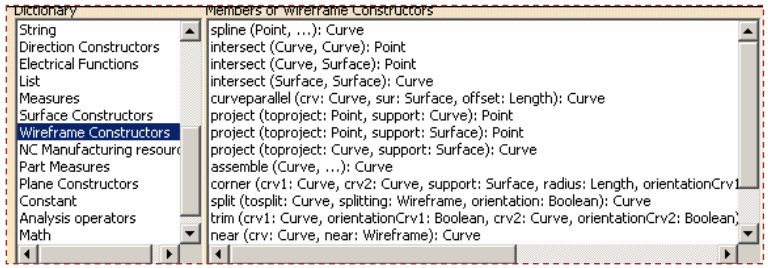
2. **Select the geometric type of element you want to create (Curve for example) and click the New parameter of type button. Close the formula editor by clicking OK.**



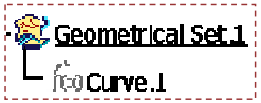
3. **The new parametric feature has been added to the tree as a geometrical element.**
You can rename it by using its properties (MB3).

4. **Create a new Rule in order to valuate the geometric parameter created previously. Use the geometrical operators from the Dictionary.**

```
if Case=="1"
{
PartBody\Curve.1 =intersect(PartBody\Extrude.1 ,PartBody\Extrude.2 )
}
else
{
PartBody\Curve.1  =intersect(PartBody\Extrude.1 ,PartBody\Offset.1  )
}
```



5. **Once the Rule is created, the geometric element is displayed in the tree with the F(x) icon meaning that it is driven by a formula or a Rule.**

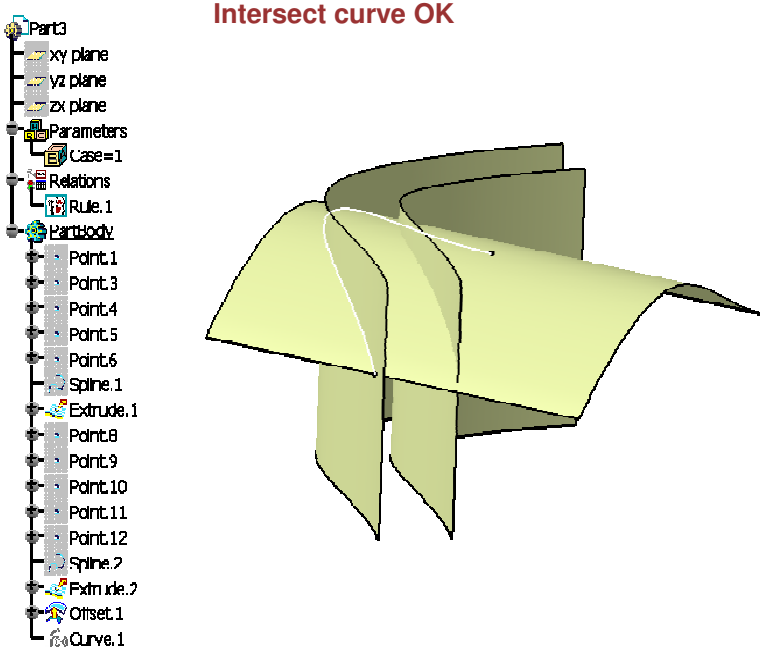


Student Notes:

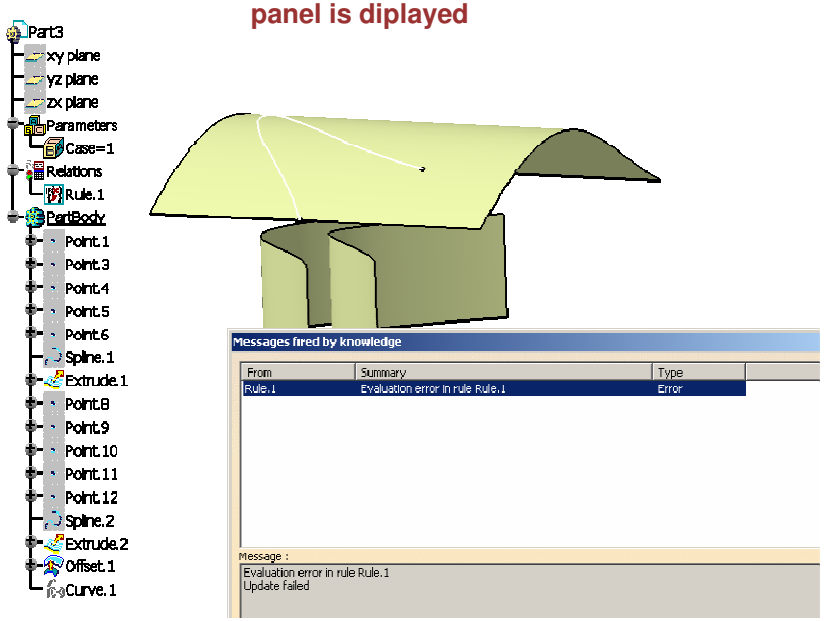
Handling Errors in Rules (1/2)

- **It is possible to test a geometric feature in error while creating rules. Indeed, the use of geometrical operators to value the geometry in relations may lead to update errors in the created features.**
- **For example, if the user values a datum curve with the result of the intersection of two surfaces, these two surfaces may not intersect and the intersection curve is therefore in error.**

Intersect curve OK



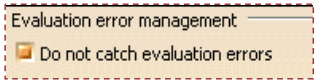
Intersect curve in error => an error panel is diplayed



Student Notes:

Handling Errors in Rules (2/2)

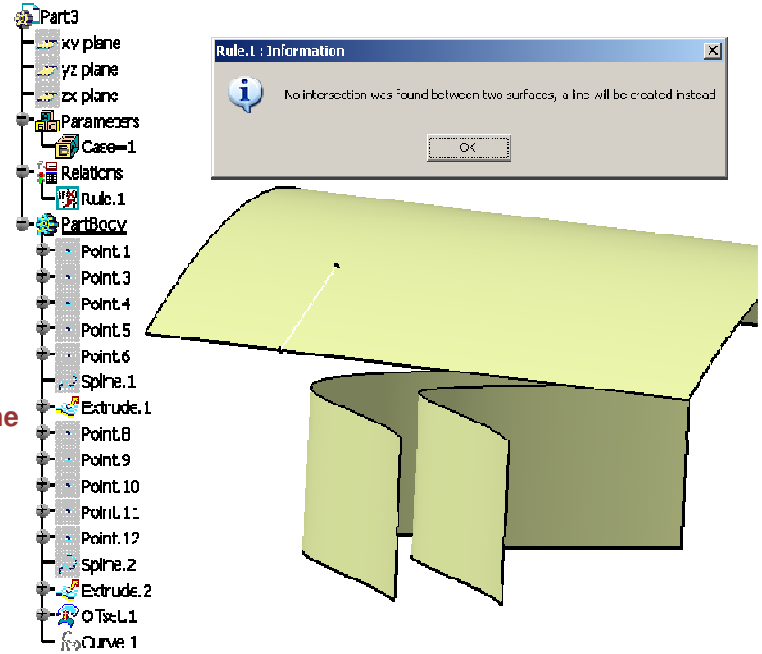
1. **In the properties of the Rule (MB3), check the "Do not catch evaluation errors" option.**



2. **To test if a feature is in error, first create a local variable using let keyword and use the error keyword as shown in the example below:**

```
let x(Curve)

if Case=="1"
{
x =intersect(PartBody\Extrude.1 ,PartBody\Extrude.2 )

 if (x.Error==true)
{
Message ("No intersection was found between two surfaces, a line will be created instead")
x=line( PartBody\Point.1 ,PartBody\Point.3 )
 }

PartBody\Curve.1=x

}
else
{
PartBody\Curve.1  =intersect(PartBody\Extrude.1 ,PartBody\Offset.1  )
}
```

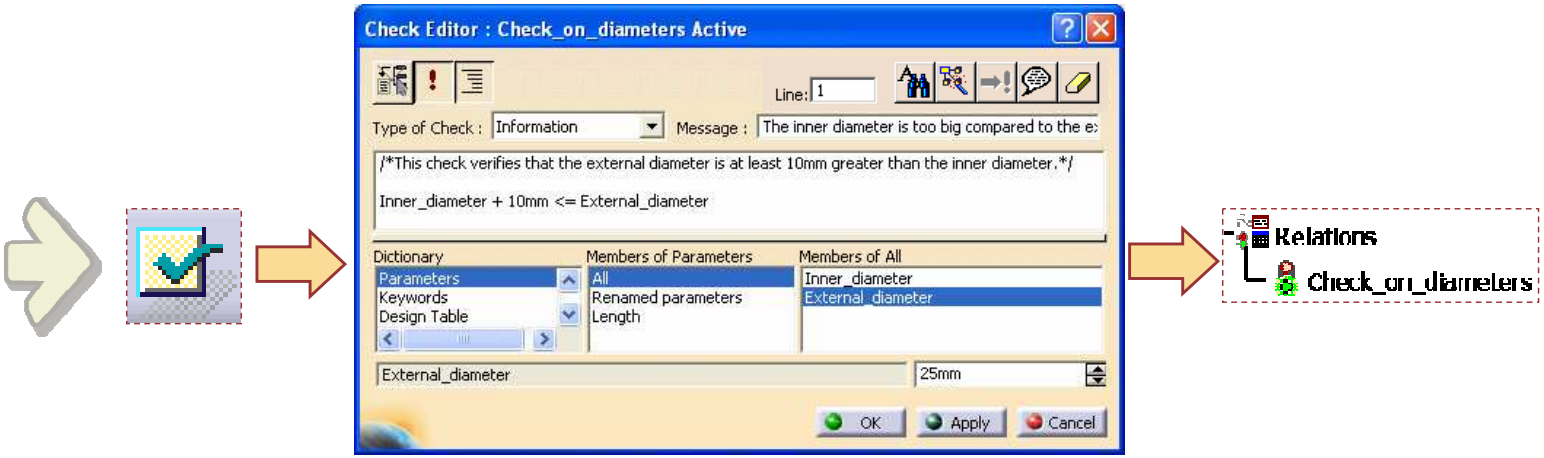
3. **Now, in case there is no intersection between the surfaces, an information panel will be displayed and the intersection result will be a line.**



Student Notes:

Creating Checks

You will learn how to create and analyze the Checks feature.



Student Notes:

What is a Check?

- A Check is a set of statements intended to let the user know whether certain conditions are fulfilled or not.
- A check does not modify the document. It is applied to and just gives a design indication.
- A check usually appears in the Relations node of the specification tree with a traffic light icon, switching to red or green according to the check's status.

Check status is OK.

Check status is not OK.

- There are three types of checks:
  - Silent – the status of the check is only indicated by the feature's icon.
  - Information - the status of the check is indicated by the icon, and an Information message occurs when the check is wrong.
  - Warning - the status of the check is indicated by the icon, and a Warning message occurs when the check is wrong.

Information message

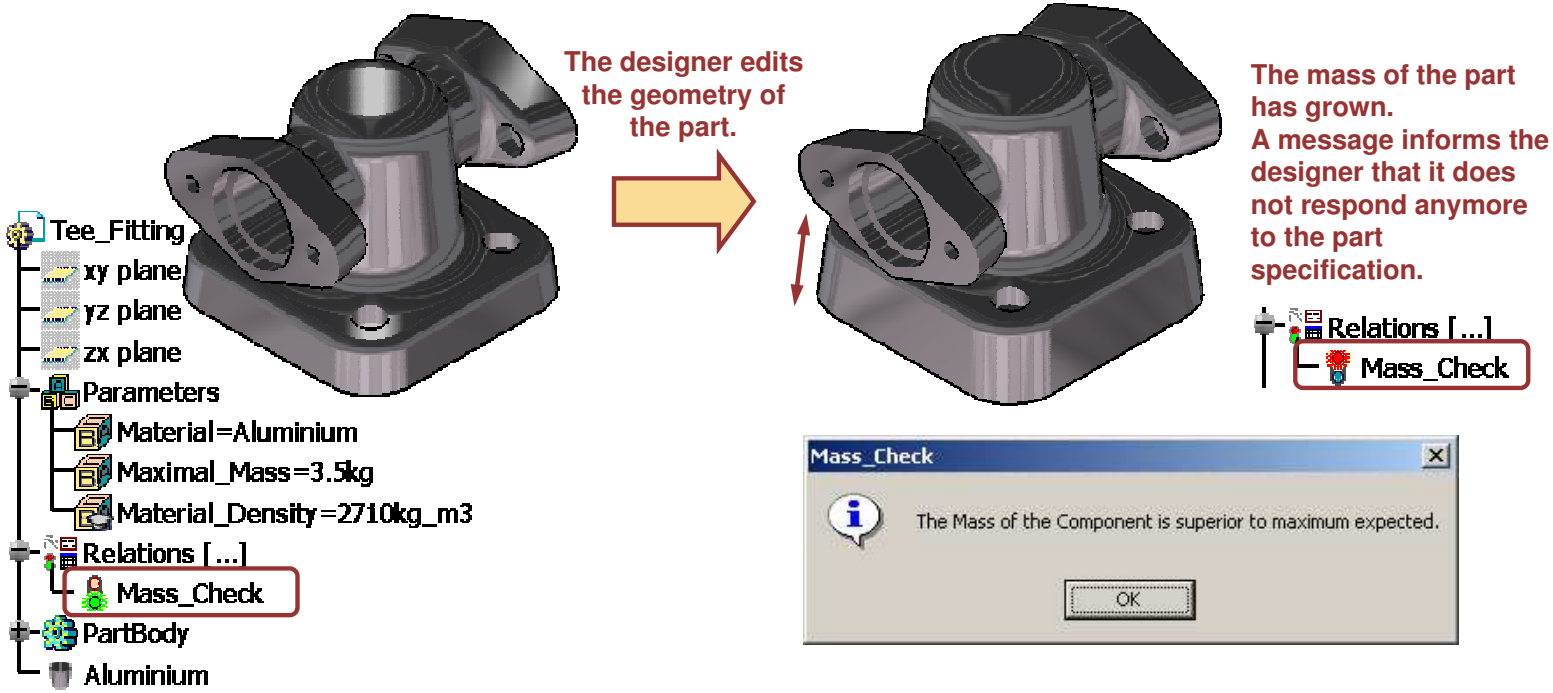
Warning message

Student Notes:

Why Use Checks?

- To check that a parameter or a component property responds to a technical limitation or to a set of conditions.
- To ensure compliance with the corporate design rules.
- To avoid update errors that are foreseeable. The check sends a warning message while editing the feature so that the unsuitable value can be changed before an update.

For instance, this check verifies that this mechanical part respects a maximum mass:



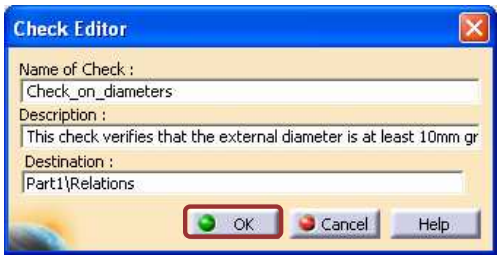
Student Notes:

Creating Checks

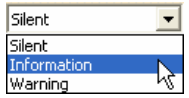
A check is a relationship between the parameters. A direct feedback on the status of the check is given in the tree, thanks to a red or a green light. In case of violation, the user can also be informed by a message panel.

1. In the Knowledge Advisor workbench, click on the Check icon.

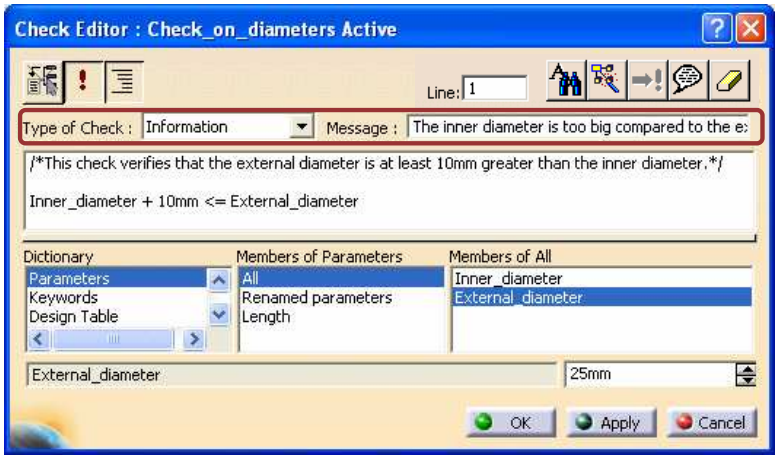
2. Enter the check name and a comment. You can also select the set of relations in which the check will be placed. Click OK.



3. The Check Editor panel has opened.
Select the type of check in the list and enter a message that will appear in case of failure.



4. Type the body of the check in the main field.
A check is a statement generally based on comparison operators:
"<", "<=", "==", ">=", ">", "<>".
You can use the Dictionary to help you select the parameters.
Click OK to validate the creation of the check.

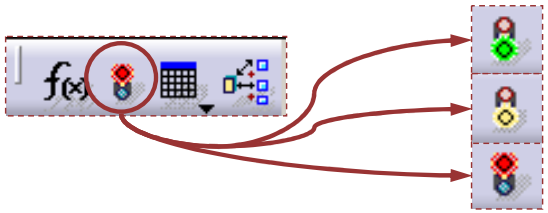


5. The Check feature is displayed in the tree under the selected Relations node/set.

Student Notes:

Analyzing Checks

- **The Global Analysis Tool is designed to manage the Knowledge Expert and the Knowledge Advisor Checks wherever they may be located in the specification tree. It helps to understand the validation status of the designs and allows navigation by checks or violations, and highlights failed components.**
- **In the Knowledge toolbar, the « Check analysis toolbox » icon light indicates the active document Checks status:**



All the checks are updated and could be fired successfully

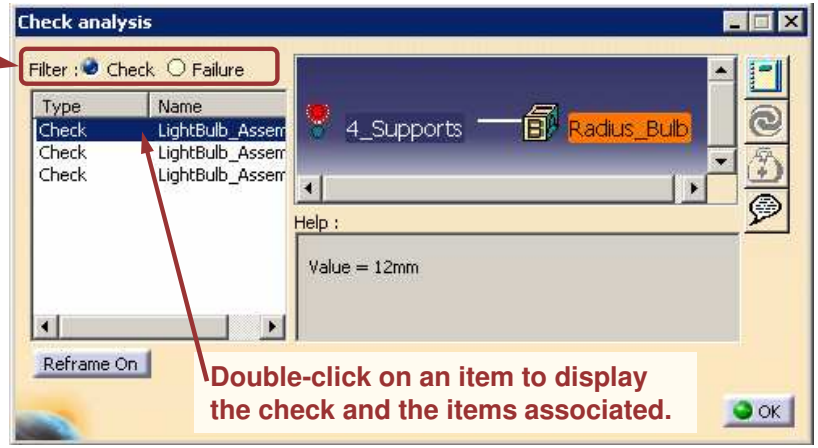
The checks need to be updated

All the checks are updated and at least one of them is incorrect

- **Click on the icon in the toolbar to accede to the Check analysis window:**

The Check mode displays only the Check features that failed while updating the check report.

The Failure mode displays all the items that failed while updating the check report.



Double-click on an item to display the check and the items associated.

Click here to generate the customizable report

Click here to solve the checks created

Click here to launch correction (only available for the Knowledge Expert Checks)

Click here to display or associate a URL

Student Notes:

Simple Example: Rule and Check

Part used: Rule_And_Check.CATPart

The attached part contains a simple example of Rule and Check.

The Rule is based on the 'Length' parameter. According to the value of the 'Length' parameter, the Rule changes the thickness of the part and the diameter of the hole.

The Rule also activates / deactivates the hole if the value of the 'Length' parameter is below 50mm.

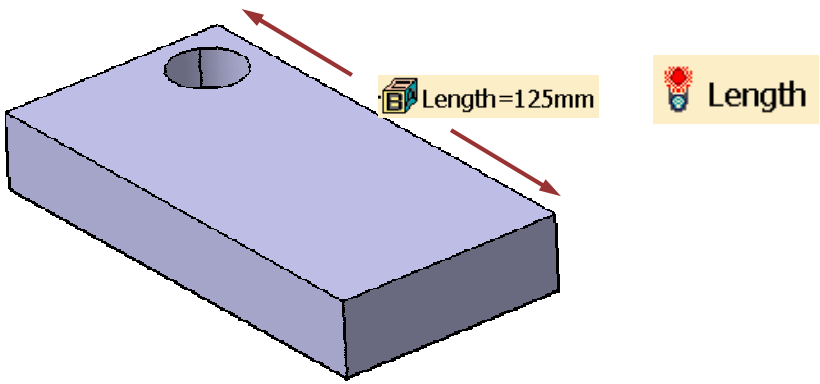
Script of the Rule

Rule Editor : Rule.1 Active

Rule.1

```
if Length <=50mm
{
Thickness=20mm
PartBody\Hole.1\Activity =false
Message("Current Length is: # | Thickness is set to: #|
}
if Length >50mm and Length <100mm
{
Thickness =45mm
PartBody\Hole.1\Activity =true
PartBody\Hole.1\Diameter =10mm
Message("Current Length is: # | Thickness is set to: # |
}
Else if Length >100mm
{
Thickness=80mm
PartBody\Hole.1\Activity =true
PartBody\Hole.1\Diameter =20mm
Message("Current Length is: # | Thickness is set to: #|
}
```

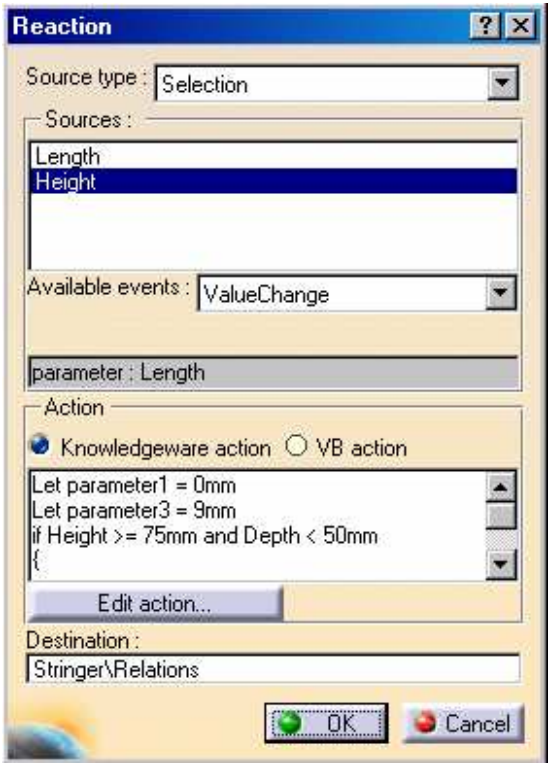
The attached part also contains a 'Check' to check if the length is above 120mm.



Student Notes:

Creating Reactions

You will become familiar with the Reaction feature.



Student Notes:

Why Use Reactions? (1/3)

- The Knowledge Advisor rules have their own limit.
  - They react to parameter changes or feature updates
    - You cannot control exactly when they are fired
    - They may be fired several times when you would not like to
  - They are integrated to the update mechanism
    - Parameters cannot be in input and in output. For example, it is not possible to write: if x>18mm {x=18mm}

The attached part 'ForceValue.CATPart' contains a reaction which forces the value of the length.1 parameter to 50mm if it is increased above 50mm.

    - Loops and conflicts are forbidden
  - Their language is simple
    - And limited too

Student Notes:

Why Use Reactions? (2/3)

- To cope with those limitations and to create more associative and reactive designs use the Reaction feature.
  - A reaction is similar to a rule in the fact that:
    - It is stored in the model
    - It reacts to changes and triggers modifications
    - It also references other objects and parameters in the document, and supports replace mechanism
    - It can be used for the definition of PowerCopies and user defined features
  - But:
    - It can react to a larger amount of changes
    - It can drive very complex modifications

Student Notes:

Why Use Reactions? (3/3)

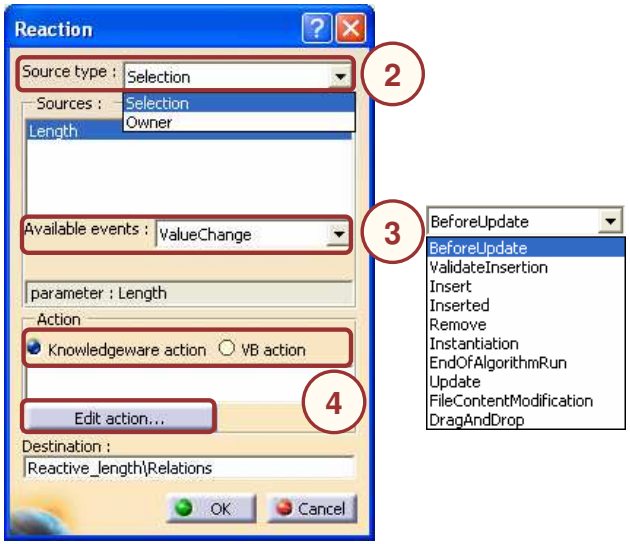
- A reaction is a feature that reacts to events on its source(s) by triggering an action
  - The source can be:
    - A selected feature (or a list)
    - A parameter (result of a test)
  - Events can be:
    - General events on objects (creation, destruction, update, attribute changes) and parameters (value change)
    - Specific events such as instantiation and update for a user defined feature
  - Action can be :
    - Written in Knowledge language to access the existing objects in the document or in the Visual Basic Script to extend the action scope
    - It can access the source object and its arguments

Student Notes:

Creating Reactions (1/3)

1. In the Knowledge Advisor workbench, click on the Reaction icon. The Reaction dialog box opens.

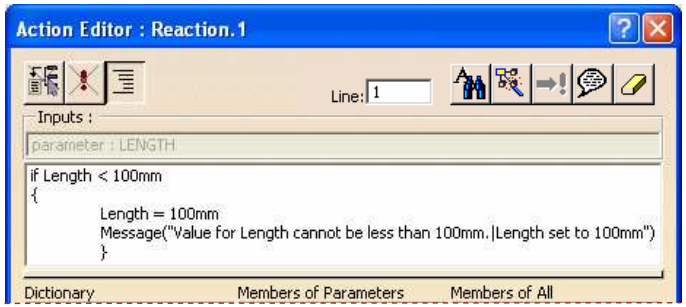
2. Select the Source type:
   - **Selection** enables you to manually select one or more items in the specification tree or in the geometrical area. These items will be displayed in the Sources field.
   - **Owner** enables you to link the action with a feature of the geometry or of the specification tree. To link the reaction with an object of the geometry, click the Destination field and select an object in the specification tree or in the geometry.



3. In the proposed list, select the **Event** which will trigger the Reaction.

4. Select the language (Knowledgeware or VBScript) in which you want to write the action triggered by the reaction. Click the Edit Action button.

> **VBScript offers some additional functions and facilities. So, in such cases you can use VBScript.**



5. The Action Editor dialog box has opened.
   Type the body of the Reaction in the main field.
   If you have chosen Knowledgeware language, use the Dictionary to select the parameters and the functions.

6. Reaction feature is displayed in the tree:
   - Under the Relations node in the Selection mode,
   - Under the source in the Owner mode.
   You can rename the Reaction using its Properties (MB3).

Selection mode

Owner mode

Student Notes:

Creating Reactions (2/3)

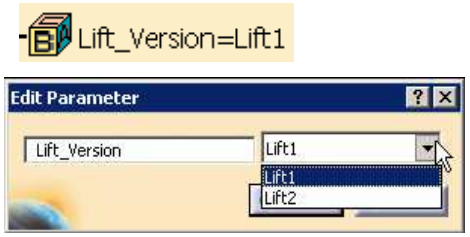
Part used: Lift.CATPart

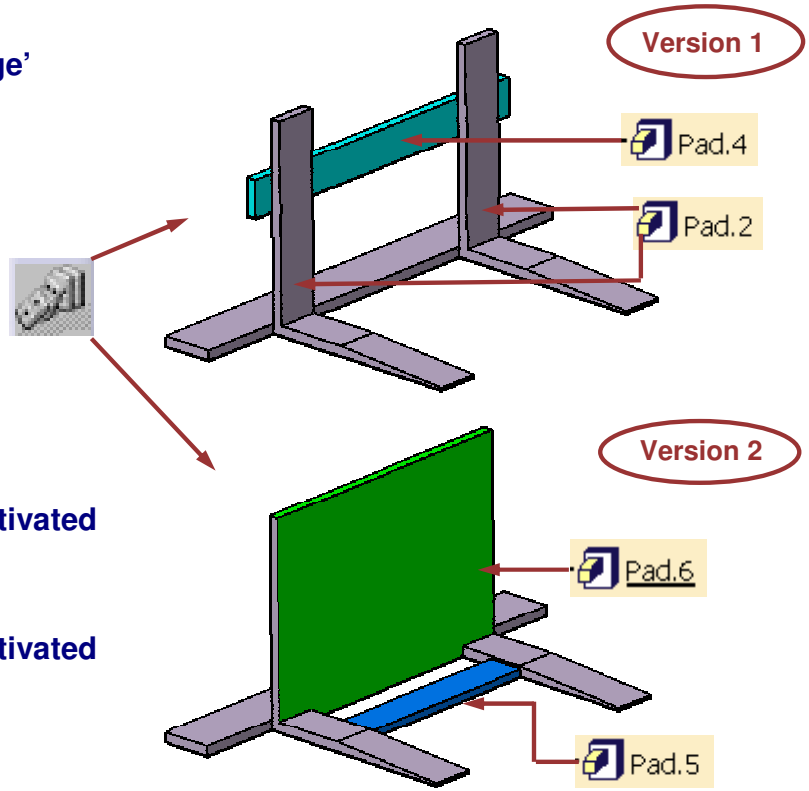
Example:

This sample illustrates how to use the Reaction feature to get two configurations of the design by activating and deactivating features.

This Reaction is written in Knowledgeware script.

This reaction functions for the 'ValueChange' event of the 'Lift_Version' parameter.





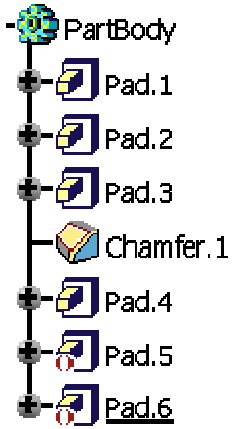
For Version 1, Pad.4 and Pad.2 are to be activated and Pad.6 and Pad.5 are to be deactivated.

For Version 2, Pad.6 and Pad.5 are to be activated and Pad.4 and Pad.2 are to be deactivated.

Student Notes:

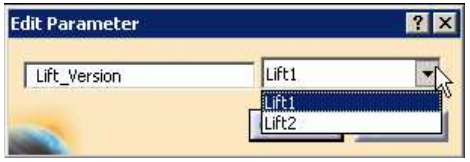
Creating Reactions (3/3)

The activation and deactivation of the features can be done by changing the 'Activity' attribute of the features as shown in the Knowledgeware script below.




```
if Lift_Version == "Lift1"
{Message("Configuration will be changed to Lift1")
        PartBody\Pad.5 .Activity  = False
        PartBody\Pad.6 .Activity = False

        PartBody\Pad.2 .Activity = True
        PartBody\Pad.4 .Activity = True}

if Lift_Version == "Lift2"
{Message("Configuration will be changed to Lift1")
        PartBody\Pad.2 .Activity = False
        PartBody\Pad.4 .Activity = False

        PartBody\Pad.5 .Activity  = True
        PartBody\Pad.6 .Activity = True}
```

You can change the value of the 'Lift_Version' parameter and see the effect.



Student Notes:

Creating a Loop in a Reaction (1/4)

- Using For statement
  - The first type of loop is a loop based on the element of a list. See syntax below:

    For x inside List
    {Body}

  - X is a variable name of a given type. It may represent an object or a value.
  - List is a variable name of type List or an expression returning a list.
  - X (like any other variable of the language) can be used in the body. It contains the Nth element of the list.
  - The body is executed Nth times, where N is the number of elements of the list.

Student Notes:

Creating a Loop in a Reaction (2/4)

- Using For statement
  - The second type of loop executes until an expression becomes false. See syntax below:

    For x while predicate
    {Body}

  - X is a variable name of the integer type. It is incremented at the end of each execution of the body.
  - Predicate is a Boolean expression. The body is executed as long as this expression is true. This expression is evaluated before the body.
  - Note that the second for operator can lead to infinite loops.

Student Notes:

Creating a Loop in a Reaction (3/4)

- Using While statement
  - This loop executes until an expression becomes false. See syntax below:

```
let i = 1
let x(Point)

for i while i<=parameter.Size()
{x = parameter.GetItem(i)
if (x.GetAttributeReal("Y") < 0.04)
x.SetAttributeReal("Y",0.04)}
```

  - i is a variable name of the integer type. It is incremented at the end of each execution of the body.
  - X is a variable for points.

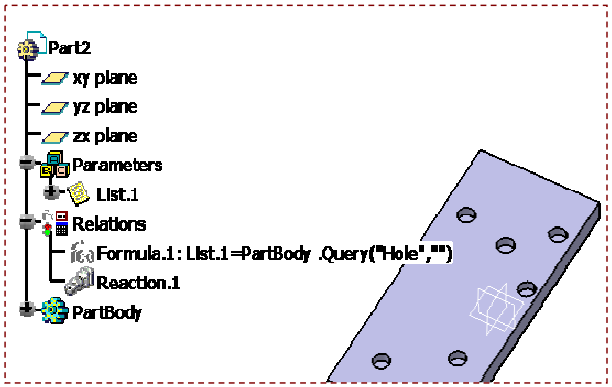
Student Notes:

Creating a Loop in a Reaction (4/4)

Part used: Loop_Reaction.CATPart

- **Example**
  - **The part contains a List feature which is automatically populated with a Query function to add in the list all the holes of the part.**
  - **There is also a Reaction using a loop to set the holes diameter to 15mm, in case their current diameter is lower than 15mm:**

```
let x(Hole)
for x inside parameter
{if (x.Diameter < 15mm)
        {Message("One hole diameter will be set to 15mm to respect company standard")
        x.Diameter=15mm}}
```



Each time a new hole is created in this part, we ensure that it will have a minimal diameter.

Student Notes:

Creating Design Tables and Part Families

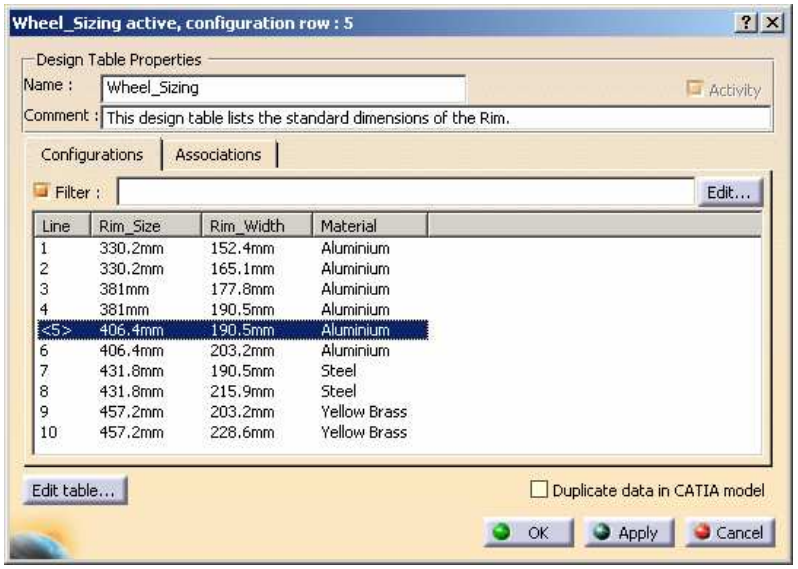
You will learn how to create Design Tables and then how to use them to create Part Families.

- **Creating Design Tables**
- **Creating a Part Family Catalog**

Student Notes:

Creating and Using Design Tables

You will learn how to create Design Tables using the document parameters.



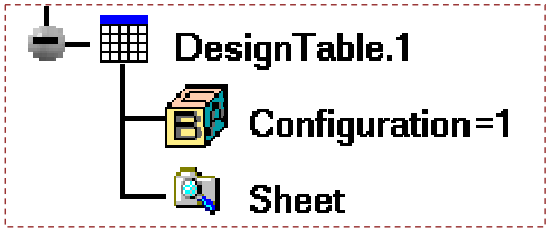
Student Notes:

What is a Design Table?

- The purpose of the Design Table is to drive the parameters of a CATIA document from external values.
- The Design Table allows to create and manage component families. These components can, for example, be mechanical parts just differing in their parameter values.
- A configuration is a set of parameter value and corresponds to a row.

- A Design Table can be created:
  - From the CATIA document parameters
  - From an external file
- The values are stored either in a Microsoft ® Excel file on Windows™ or in a tabulated text file.

Design Table icon in the Knowledge Toolbar



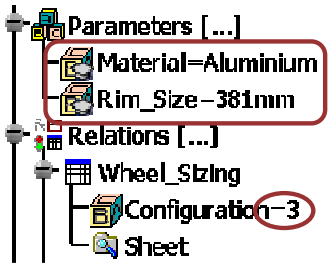
If you create the design from an existing file, it is possible to indicate the sheet number where the table is found.

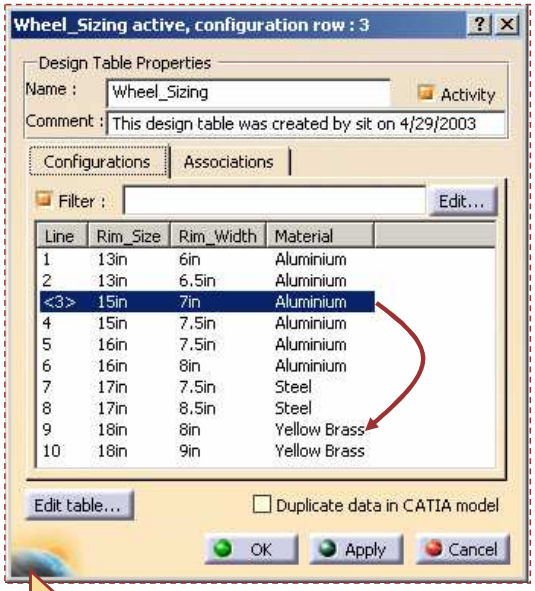
Student Notes:

Why Use Design Tables?

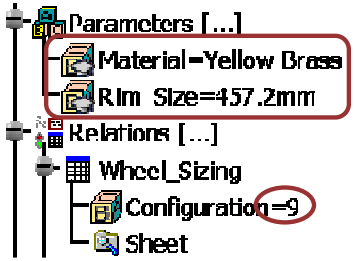
- To pre-define possible configurations of the model and to ease the modifications of the dimensions.
- To select only the realistic configurations of the component.
- To link the parameter values that cannot be expressed with a mathematical relation.
- To create part families.

Here is a part whose main dimensions are driven by a design table.





When you change its configuration, three parameters are updated at a time, including an intrinsic parameter (the access of which is not easy).



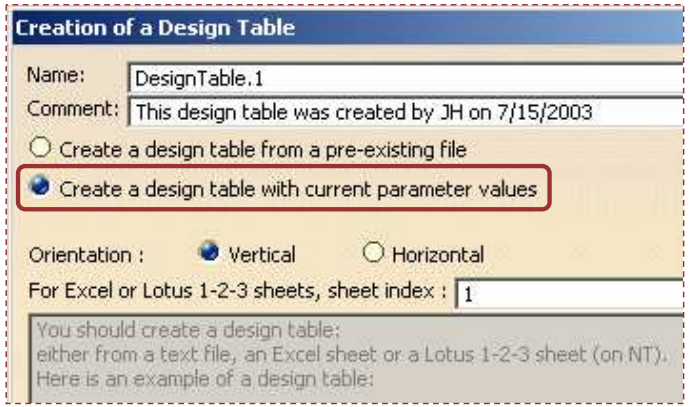
Student Notes:

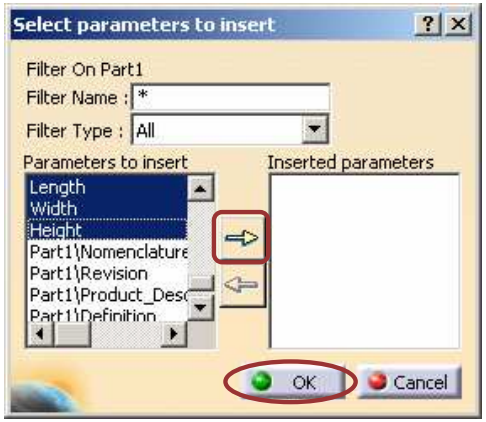
Creating a Design Table from Document Parameters (1/2)

1. 

   Click on the Design Table icon.

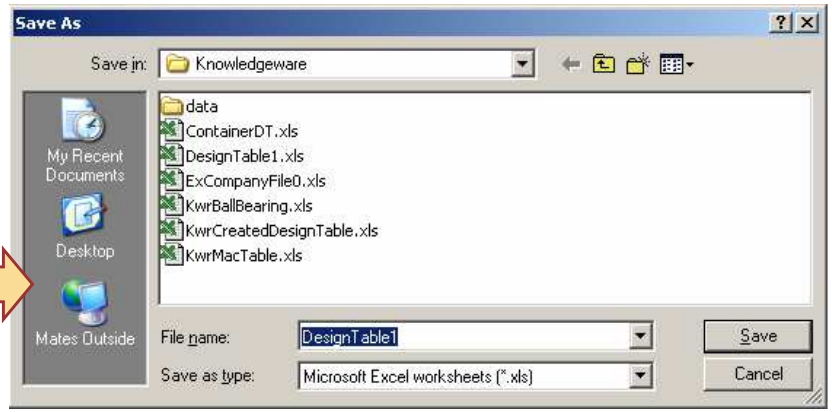
2. **The Design Table creation panel is opened. Select the option Create a design table with current parameter values. Click OK.**



3. 

   Select the parameters to add to the design table and use the arrows to add them to the list. Click OK.

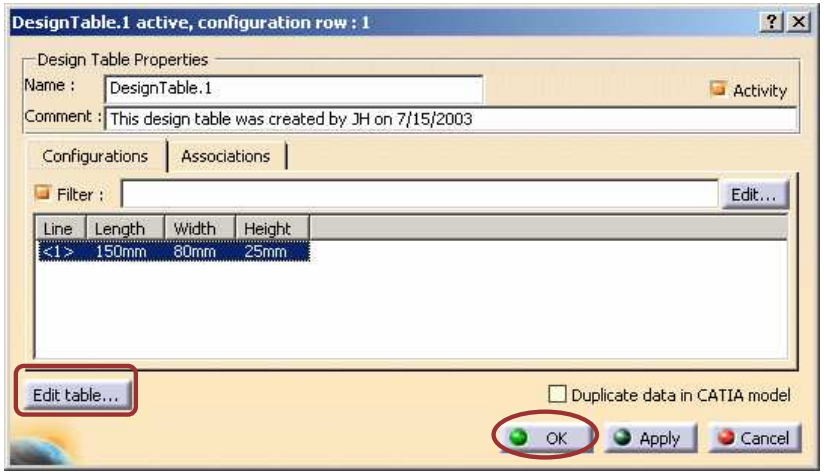
4. **Specify the folder and the file name where the data are stored. Click the Save button.**



Student Notes:

Creating a Design Table from Document Parameters (2/2)

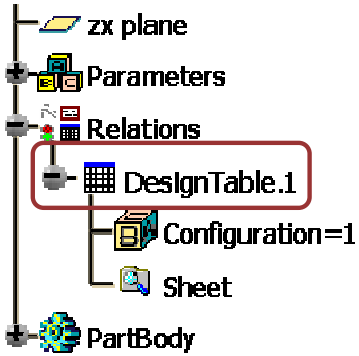
5



The Design Table dialog box has appeared. The Design Table contains only one configuration: the current one. If you want to add more configurations, click the Edit table button.
Click OK to confirm the Table creation.

6

The Design Table feature appears in the specification tree within the Relations node.



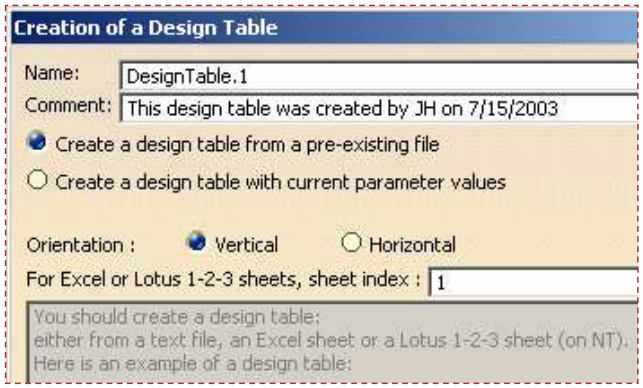
Student Notes:

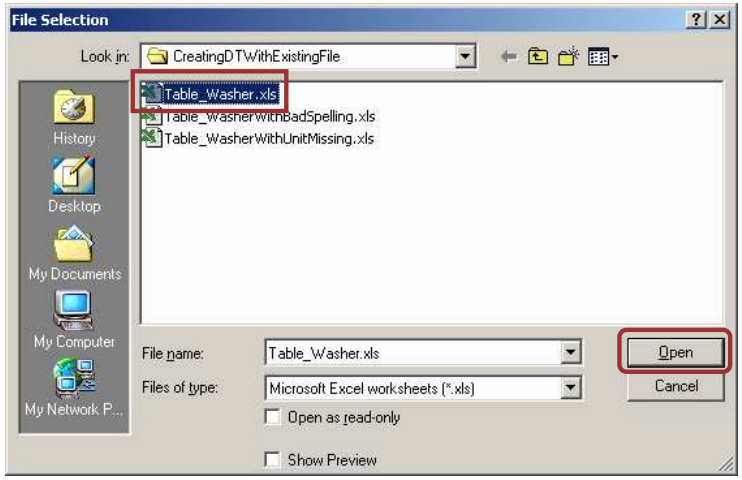
Creating a Design Table with an Existing File (1/2)

You can also create a design table from an already existing file.

1. Select the Design Table icon.

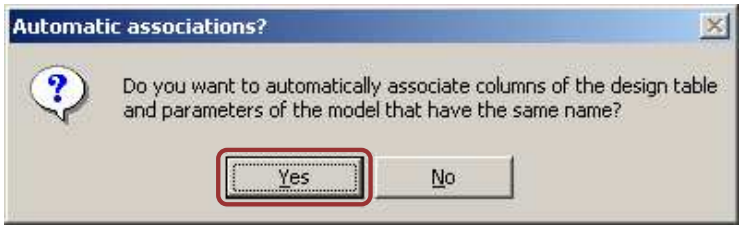
2. The Design Table creation panel is open. Select the Create a design table from a pre-existing file option; Click OK.





3. Specify the external file containing data of your design table; Click the Open button.

4. Click yes if you want an automatic association between the columns of the external file and the parameters of the CATIA document.

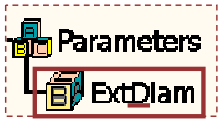


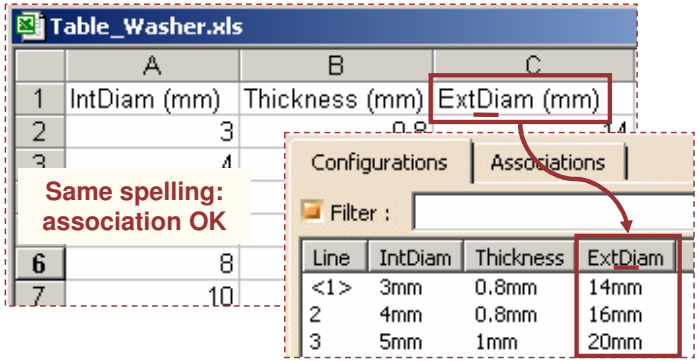
Student Notes:

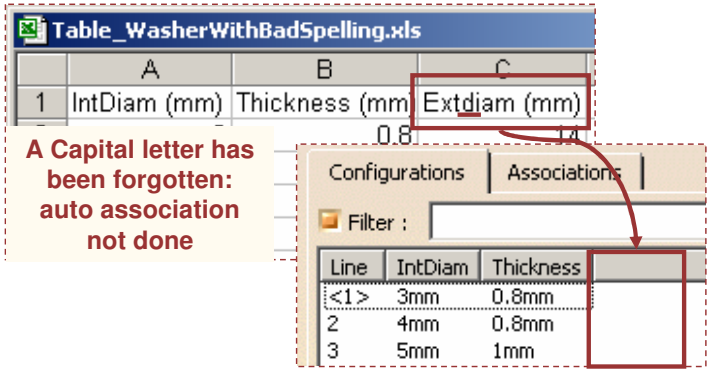
Creating a Design Table with an Existing File (2/2)

When using an existing file, you have to manage the associations between the columns and the parameters. Here are a few pieces of advice to have them automatically made.

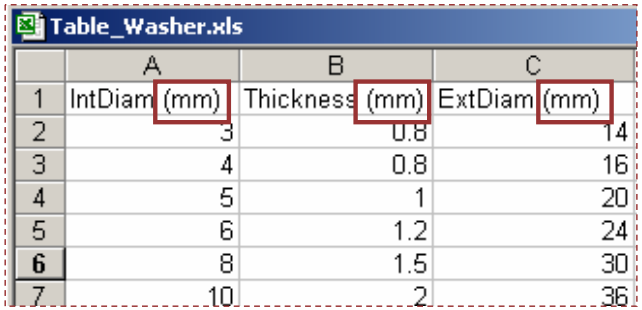
1. **Automatic association occurs between the parameters and the columns having exactly the same spelling (take care of blank space and capital letters).**



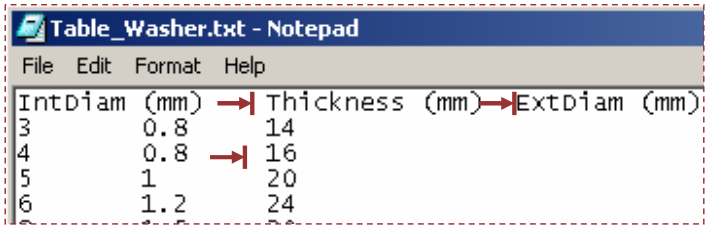




2. **In the external file, be careful to specify the units of the values in the top case of the column. If not done, CATIA considers they have the international system (meter for length etc…).**



3. **If the external file is a text file, take care of having only one tab space between the titles and between the values.**

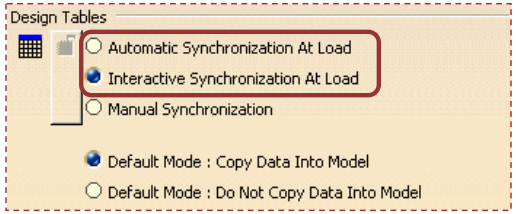


Student Notes:

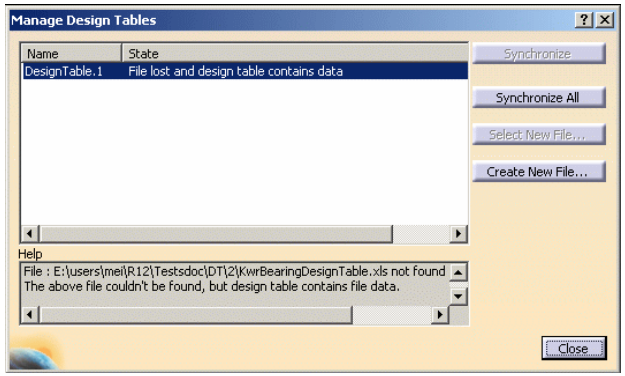
Generating a File From a Design Table

It is possible to regenerate an external file (.XLS or .txt format) using the data contained in the model. The data contained in the model comes from an external file that was previously deleted.
The design Table has to be created with the Duplicate data in the CATIA model option.

1. From the Tools->Options...->Parameters and Measure command, access the Knowledge tab and make sure the Interactive Synchronization At Load is checked.



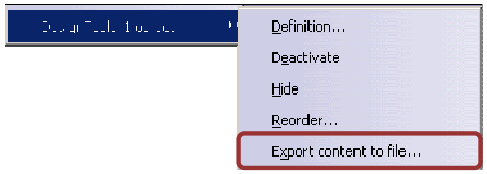
2. Open the CATPart document of which the Design Table file has been deleted or renamed without CATIA. The Manage Design Tables window displays indicating that the external file has been deleted.



3. Click the Create New File... button to generate a file from the data contained in the .CATPart document. The Save As dialog box displays

OR...

If you are working with the option Automatic Synchronization At Load, right-click the DesignTable in the specification tree and select the DesignTable.x object->Export content to file... command.



4. Enter the name of the file that you want to create: .XLS is the default file type. The text format is also available. Click Save and Close when done. The file containing the design table data is created.

Student Notes:

Design Table Functions (1/3)

Various Design Table methods are available to find / set values and configurations in the design tables. These functions can be used in Rules and Reactions. The explanation for a few functions is given below.

CloserSupConfig()

This function applies to a design table sheet. It returns the configuration which contains the values closest to those given in the arguments.

When several configurations meet this condition, the method sorts out the possible configurations with respect to the column order as it is specified in the argument list.

Syntax of the function is given below:
sheet.CloserSupConfig(columnName: String, minValue: Literal, ...): Integer

No.	SketchRadius(mm)	Pad_Limit_1(mm)	Pad_Limit_2(mm)
1	120	60	10
2	130	50	30
3	120	60	25
4	140	50	40

For the design table shown above, an example of the use of 'CloserSupConfig' is given below.

Relations\DesignTable1\sheet_name.CloserSupConfig("SketchRadius", 120mm, "PadLim1", 60mm, "PadLim2", 20mm)

The above function will return configuration number '3' ('third' configuration).

Student Notes:

Design Table Functions (2/3)

CellAsReal()

This function applies to a design table sheet. It returns the contents of a cell (intended for real values). Returns zero if the cell does not contain a real value or if the method arguments are not properly specified.

Syntax

sheet.CellAsReal(rowIndex: Integer, columnIndex: Integer): Real

In the above syntax, the rowIndex is the configuration number (integer from 1 to n) and columnIndex is the column number.

No.	SketchRadius(mm)	Pad_Limit_1(mm)	Pad_Limit_2(mm)
1	120	60	10
2	130	50	30
3	120	60	25
4	140	50	40

Relations\DesignTable1\sheet_name.CellAsReal(3, 2)

The above function will return 60.

Student Notes:

Design Table Functions (3/3)

SetCell()
Enables you to fill in a cell at a given position in an Excel file or a tab file.
Note: the index must start at 1 for the (1,1) cell to be located at the left top corner.

Syntax:
sheet.SetCell(IndexRow:Integer, IndexColumn:Integer, CellValue:Literal): Void
Example:
Sheet.SetCell(2, 2, 45)

No.	Radius(mm)	Pad_Limit_1(mm)
1	120	60
2	130	45
3	120	60
4	140	50

Student Notes:

Creating a Part Family Catalog

You will learn how to create a Part Family Catalog from a Part containing a Design Table.

Student Notes:

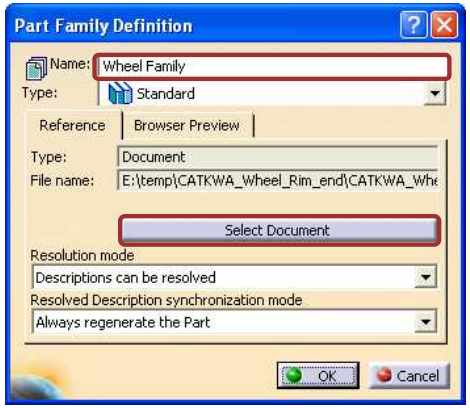
Creating a Part Family Catalog

1. Edit the Part's Design Table and insert a column called "PartNumber". Fill in this column with the names that will be given to the parts that are going to be generated.

	A	B	C	D
1	PartNumber	Rim_Size (in)	Rim_Width (in)	Material
2	Rim_13_6	13	6	Aluminium
3	Rim_13_6.5	13	6.5	Aluminium
4	Rim_15_7	15	7	Aluminium
5	Rim_15_7.5	15	7.5	Aluminium
6	Rim_16_7.5	16	7.5	Aluminium
7	Rim_16_8	16	8	Aluminium

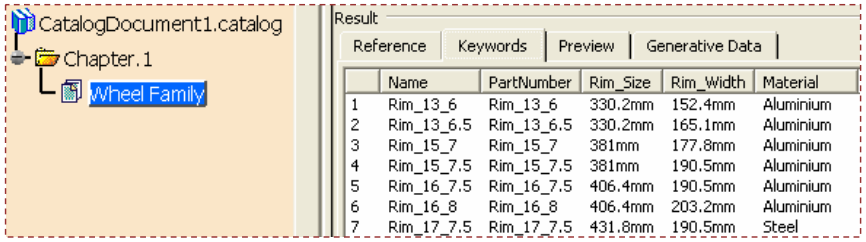
2. Create a new CatalogDocument (File>New). Activate a chapter and click on the Add Part Family icon.

3. Click the Select Document button to browse the CATPart definition document. The CATPart must contain at least one Design Table with a PartNumber column.
Enter a name for the Family in the top field.



4. In the Browser preview tab, click the Select an external preview file button to preview an external file in the .jpg, .bmp., etc. format (optional).

5. The part family is created and displayed in the specification tree. It contains a component per line of the design table.
Save the new Catalog document.

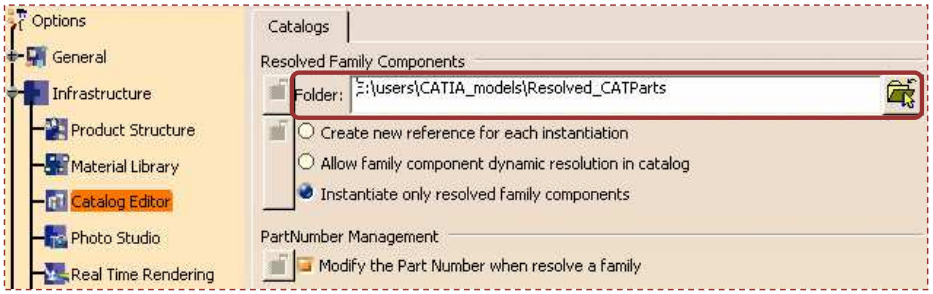


Student Notes:

Part Family Resolution (1/2)

Resolving a Part Family means that you generate the .CATPart documents referred to by the Part Family.
These documents are generated in a specific place, and each generated document is a copy of the generative part configured with the matching row in the design table.

1 **In Tools>Options indicate the folder where the CATParts associated to the resolved components will be generated.**

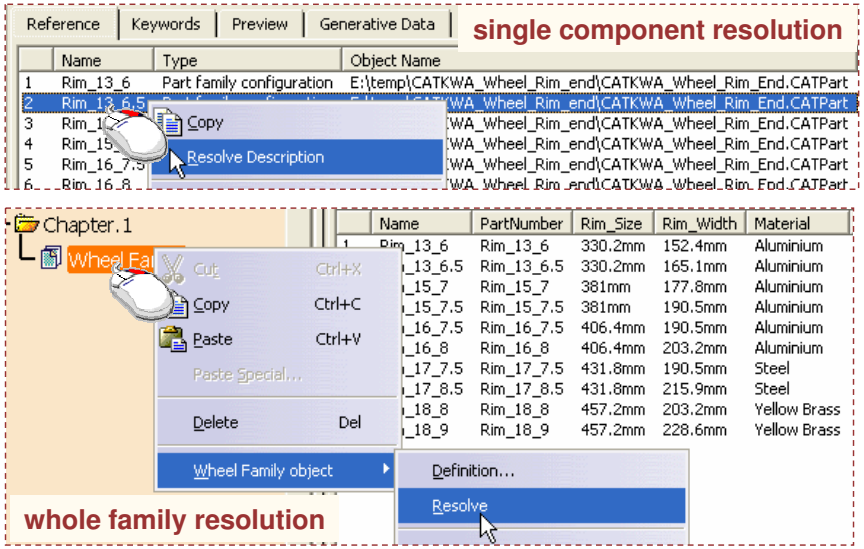


2 **If not already opened, open the catalog containing the Part Family. Activate the Part Family.**

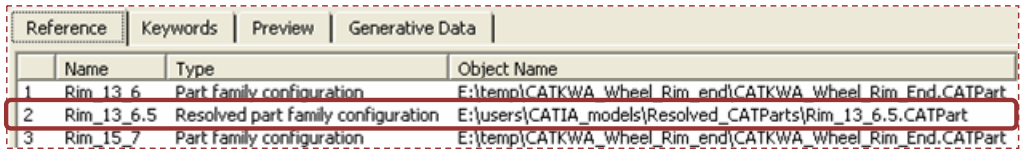
Student Notes:

Part Family Resolution (2/2)

3 **You can resolve either the entire Part Family or a single Part Family component. In both cases, use the Resolve option in the contextual menu.**



4 **The resolved component(s) can be identified in the Part Family description.**



Student Notes:

Using Knowledge Advisor Tools

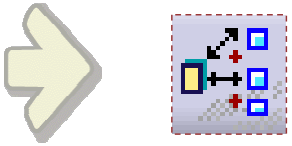
You will learn how to use Knowledge Advisor Tools.

- **Using the Knowledge Inspector Tool**
- **Using the Set of Equations Tool**
- **Creating and Using Laws**

Student Notes:

Knowledge Inspector Tool

You will learn how to use the Knowledge Inspector tool in order to analyze modifications, impacts, and dependencies.



Student Notes:

Using Knowledge Inspector: “What if” Mode (Impacts)

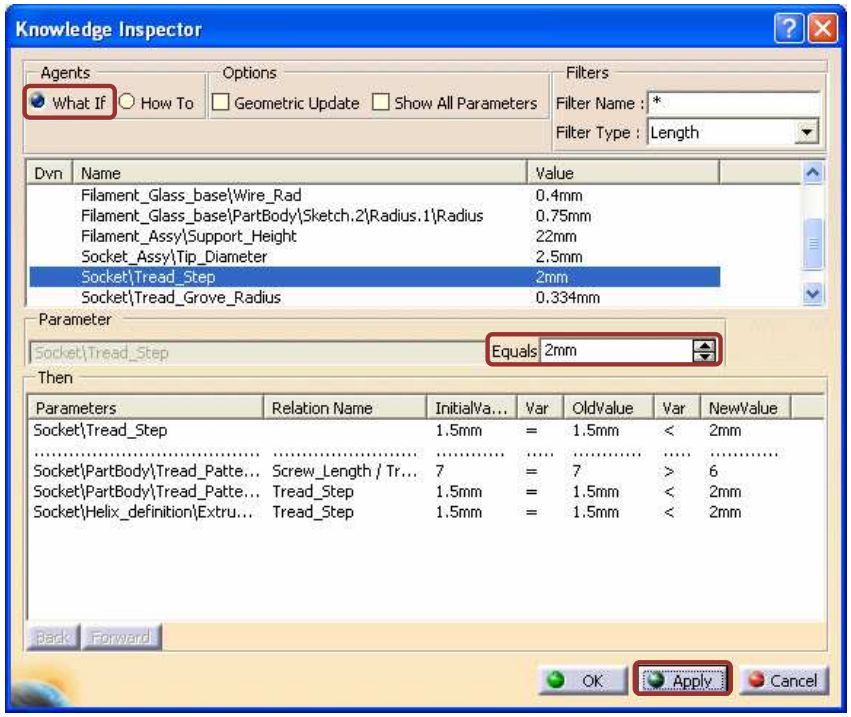
This mode helps you to understand to what extent changing any parameter of your design (such as a dimensional parameter or a material) changes the operation or design of the product on which you are working. It can be used to examine interactions of parameters with each other, and with the rules that make up the product's specifications.

1. Click on the Knowledge Inspector icon in the common knowledge toolbar.

2. Check the “What If” option. All the driving parameters are displayed in the top parameters list.
   Check the “Show All Parameters” option to display all the parameters of the document.
   Check the “Geometric Update” if you want to visualize the result of your modification in the geometry area.

3. Select in the list the parameter whose impacts are to be analyzed.

4. Use the Equals field to modify the selected parameter value.
   Click on Apply or Enter to display the values of the impacted elements in the “Then” area.



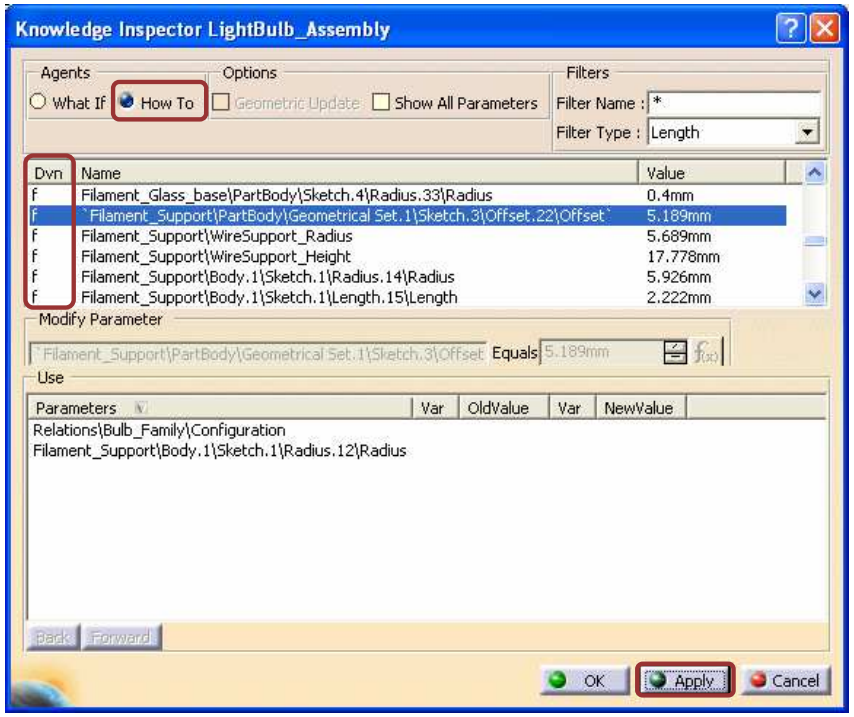
Student Notes:

Using Knowledge Inspector: “How to” Mode (Dependencies)

Helps you to determine how your design can be changed to achieve a desired result.

1. Click on the Knowledge Inspector icon in the Common Knowledge toolbar.

2. Check the “How to” option. The list of all the parameters of the document that are driven by a relation is displayed.
Check “Show all Parameters” to have a list of all the parameters of the document. The driven parameters are identified by an “f” in the left column.



3. Select the parameters whose dependencies are to be analyzed.

4. Click on Apply or Enter.
The list of impacting parameters is displayed in the use area.

Student Notes:

Using the Set of Equations tool

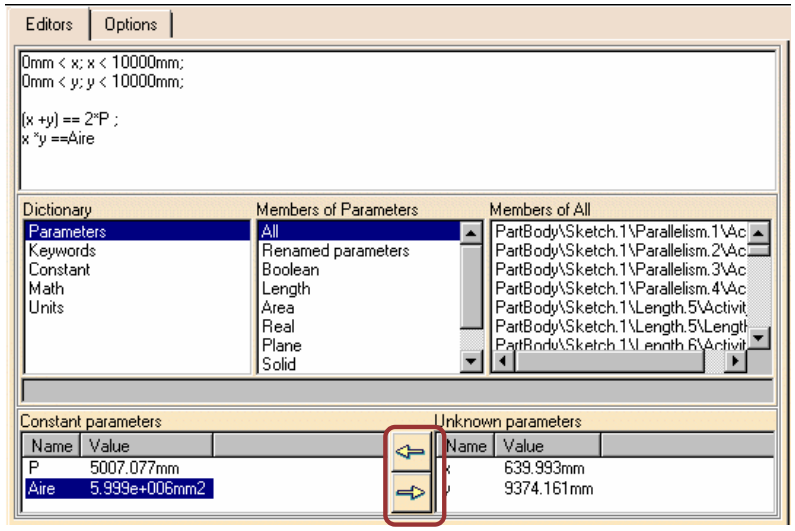
You will learn how to use the Set of Equations tool to solve the engineering problems.

Student Notes:

Using the Set of Equations Tool

1. Click on the « Set of Equations » icon.

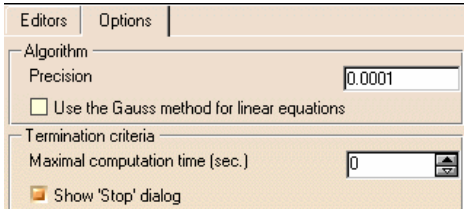
2. Define your set of equations in the editor, using the existing parameters.



3. Use the arrow button to define which paramaters are Constant parameters or Unknown parameters (to be solved).

   Constant parameters can be modified by using the formula editor.

4. Select the solve options.



   Precision option defines the precision of the result.

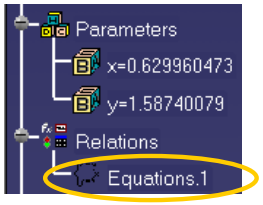
   The Gauss method accelerates the solve operation while working with the linear equations.

   Maximal computation time enables you to indicate the computation time (if 0, the computation will last until a solution is found)

   The Show 'Stop' dialog option displays a 'Stop' dialog box that will enable you to interrupt the computation.

5. Click « Apply » to check the syntax.

6. Click « OK » to exit the editor and solve the equation system.



Student Notes:

Creating and Using Laws

You will learn how to create and use the Knowledge Advisor Laws and how to combine the Knowledge Advisor (KWA) and the Generative Shape Design (GSD) Laws.

Knowledge Advisor Law

Generative Shape Design Law

Student Notes:

Creating a Knowledge Advisor Law

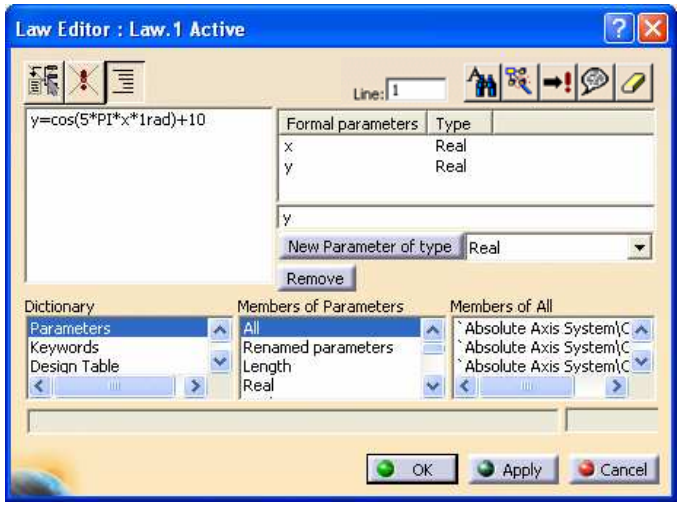
A Knowledge Advisor law is a relation whereby a parameter is defined with respect to another single parameter. Both the parameters involved in a law are called formal parameters. The formal parameters and laws are specifically designed to be used in the creation of shape design parallel curves.

1. Click on the Law icon.

2. Select a destination and give a name to the law.

3. Use the New Parameter of type button to create the formal parameters that will be used to define the law.

4. Enter the law definition, for example: y=cos(5*PI*x*1rad)+10



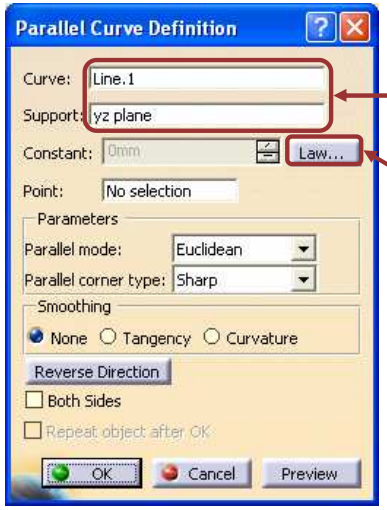
5. The Law feature is created under the Relations node.

Student Notes:

Using a Knowledge Advisor Law for Parallel Curves Definition

1. Create a Line as the reference curve.

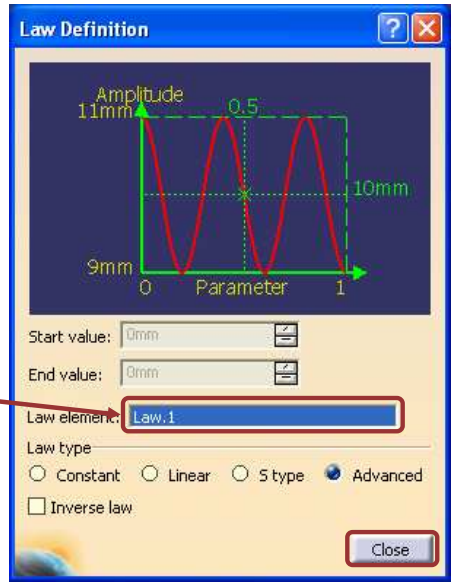
2. Click on the Parallel curve icon to create a curve parallel to the previous line:



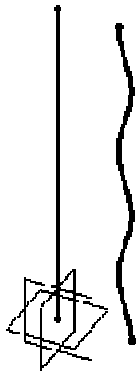
Select the reference line and the support plane.

Click the Law button:

The Law Definition panel appears, select a Knowledge Advisor law in the tree and click Close.



3. The parallel curve is created according to the law definition:

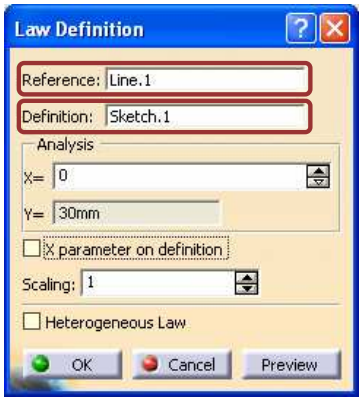


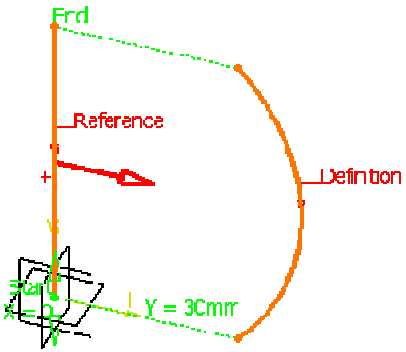
Student Notes:

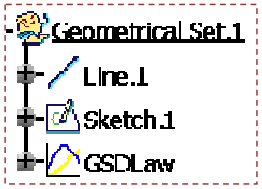
Combining Knowledge Advisor Laws and GSD Laws

You can use a combination of a Generative Shape Design law and a Knowledge Advisor law in the same relation.

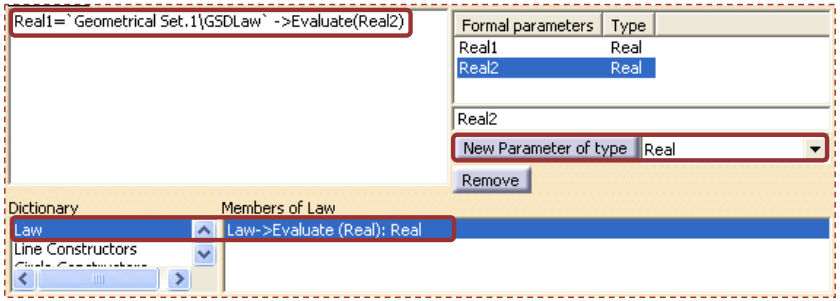
1 **Create a GSD law using a reference and a definition curve.**







2 **Create a new Knowledge Advisor law.**
Use the GSD law with Evaluate method to define it:



Student Notes:

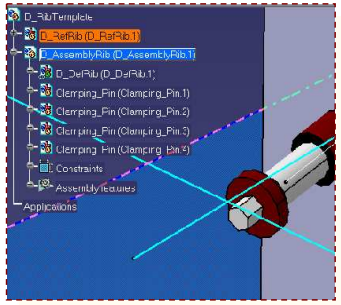
PKT Workbench Presentation

You will learn the concept of Templates and about the user interface and specific settings of the Product Knowledge Template Workbench.

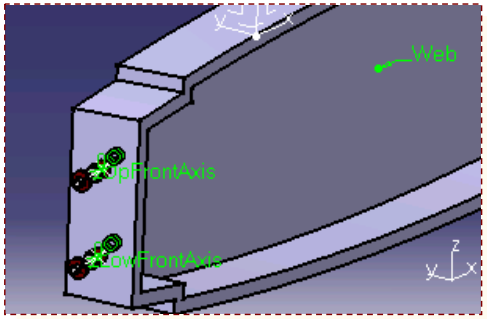
Student Notes:

What are Templates? (1/2)

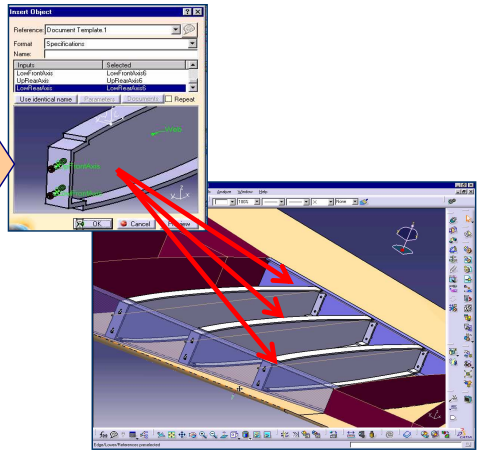
- **A template is a user-defined reusable component which automates engineering tasks.**
- **A template is built 100% interactively by generalization of an existing Design :**
  - **The generalization is performed by selecting the elements required in the Template : documents , geometric elements, parameters, rules, etc.**
  - **CATIA V5 will automatically determine which inputs will be necessary to re-create these elements when instantiating the Template (Template inputs)**



Interactive design of the model



Generalization



Multiple Instantiation

Student Notes:

What are Templates? (2/2)

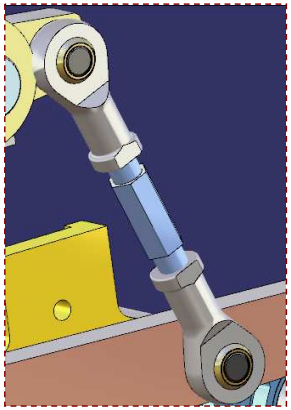
- Users can create 3 types of templates:
  - User Feature/Power Copy: a collection of CATIA features, including knowledge features, that can be reused in a part's design
    - Allows customers to manipulate their own semantic objects in place of V5 standard objects.
    - Once instantiated, users get a black box (in case of UDF) behaving like any other feature with published parameters that can be edited.
  - Part Template: a part and its associated documents (drawing, analysis, process) to be reused inside products
    - Once instantiated, the part is duplicated and you get an independent component which is adapted to the new context.
  - Assembly Template: a whole assembly and its associated documents for reuse inside products
    - Once instantiated, the assembly is duplicated and the embedded parts can be independent or reference the original one.

Student Notes:

Example of Templates

ASSEMBLY TEMPLATE

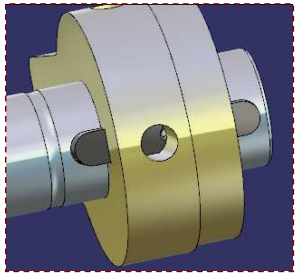
- Whole assembly duplication mechanism with associated documents
- Parts in Instance (copy) or Reference mode



Connecting Rod

PART TEMPLATE

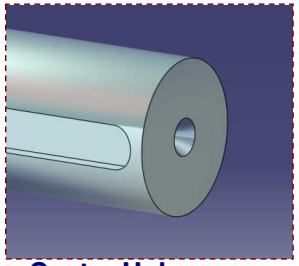
- Part duplication mechanism
  - Part number generation, New from
- Associated documents can be part of the template definition (drawing, analysis)



Parallel Key

POWERCOPY / UDF

- Set of features including Knowledge features
- Input selection
- Published parameters valuation
- Icon, Grab screen



Center Hole

Student Notes:

Accessing the Workbench

Access from:

1- the Start menu.

2- the Workbench Icon.

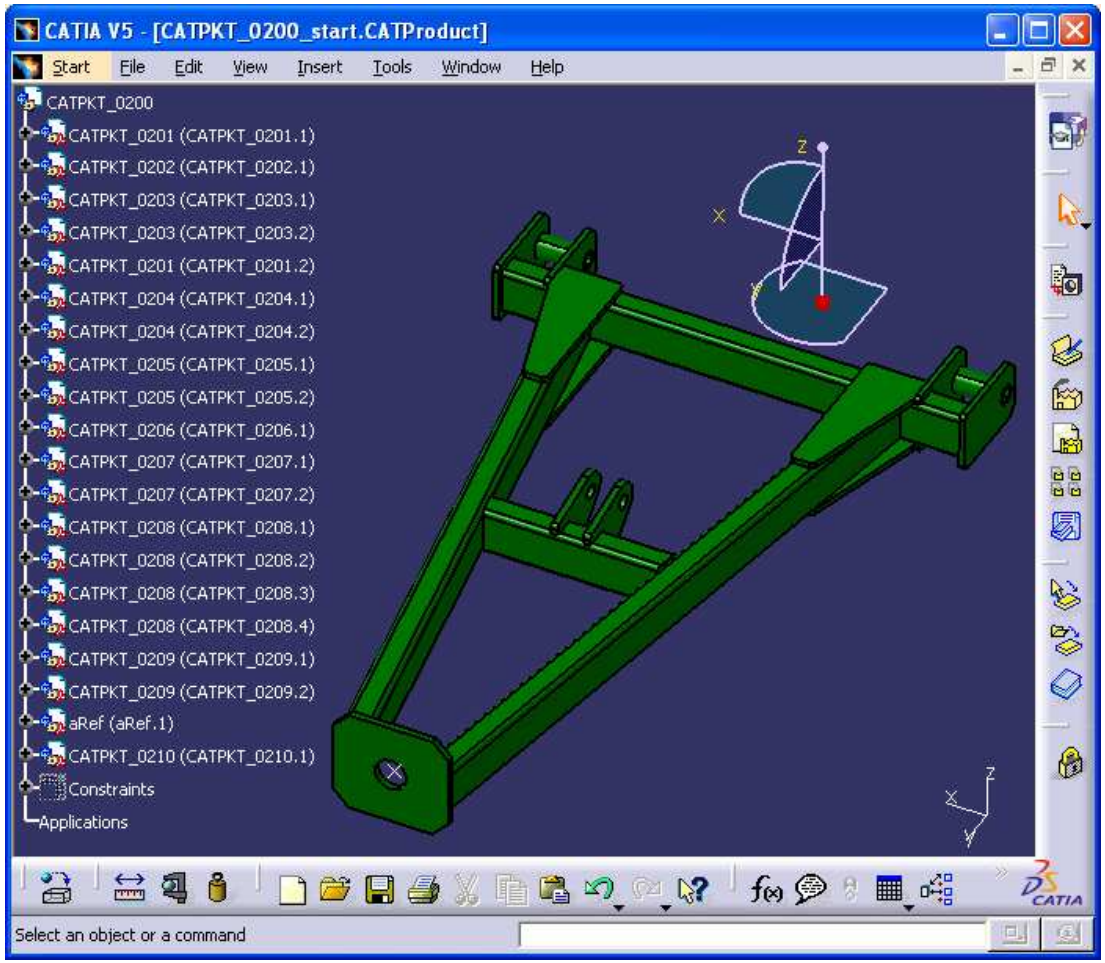
1-

2-

Use Tools/Customize+Start Menu to include Product Knowledge Template in your favourite workbenches.

Student Notes:

User Interface



Create a Generative Script *

Create a Powercopy
Create a User Feature
Create a Document Template
Create a Knowledge Pattern *
Save in Catalog

Instantiate from Selection
Instantiate from Document
Open Catalog

Debug an UDF

*** : topics not covered in this course. Refer to CATIA documentation for information.**

Student Notes:

Creating and Using PowerCopies

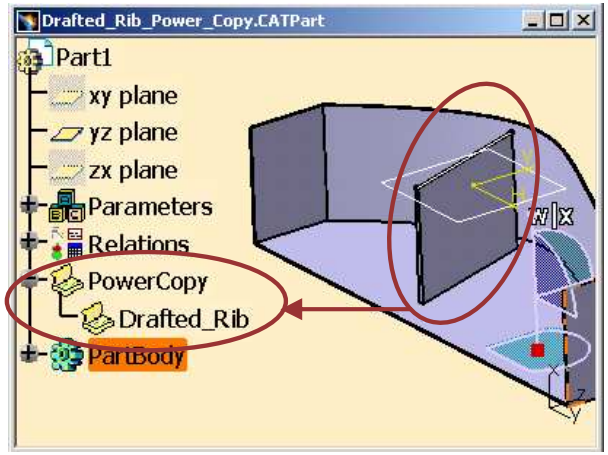
You will learn how to create and store reusable components called PowerCopies.

- **PowerCopy Presentation**
- **Creating a PowerCopy**
- **Saving a PowerCopy**
- **Instantiating a PowerCopy**
- **To Sum Up**

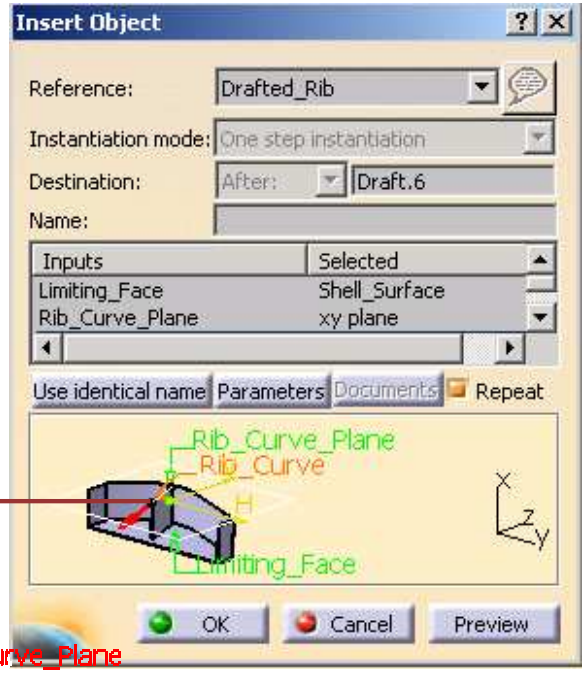
Student Notes:

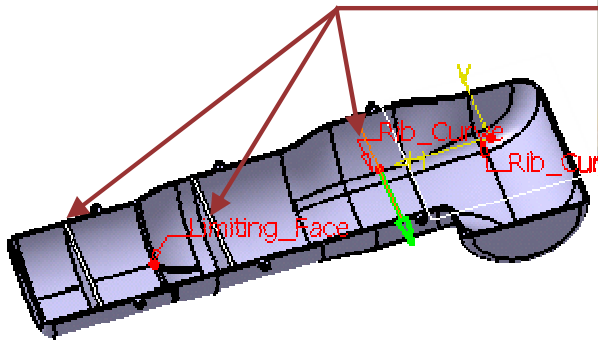
PowerCopy Presentation

In this lesson, you will have an overview of 'PowerCopy' and the way in which it can be used.



PowerCopy definition



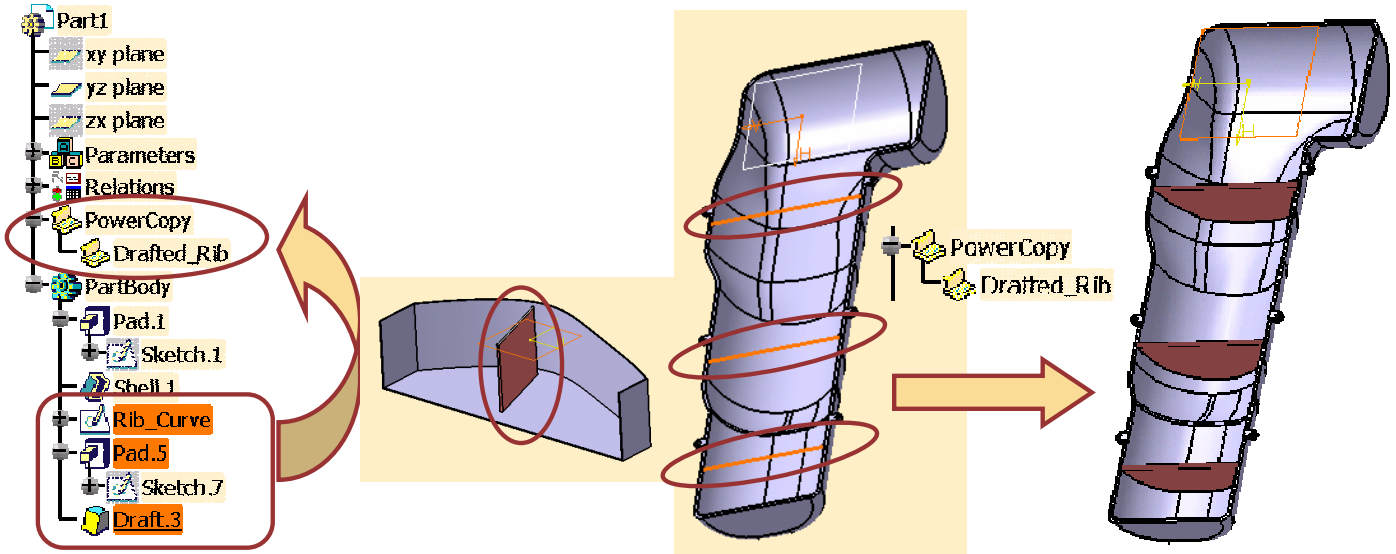


PowerCopy instantiation

Student Notes:

What is a PowerCopy?

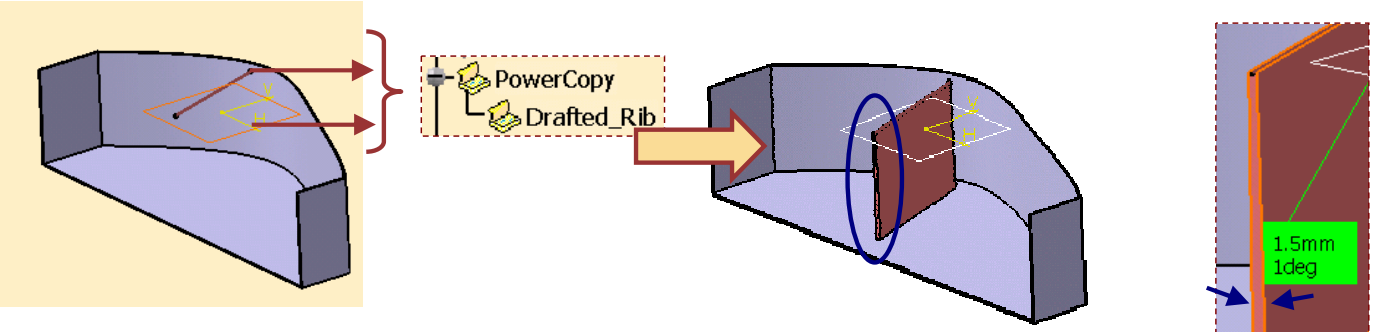
- **PowerCopy is a set of design features grouped together in order to be reproduced. It is a kind of advanced copying tool.**
  - **While defining it, you can specify the inputs that the user must provide.**
  - **During instantiation, you can customize it and insert it in the design of any part.**

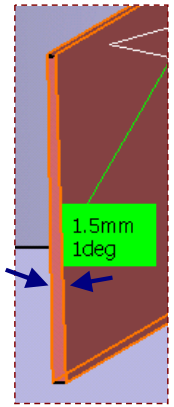


- **PowerCopy tools are available in the Insert menu > Knowledge Templates of the following workbenches:**
  - **Part design**
  - **GSD**
  - **SheetMetal Design**

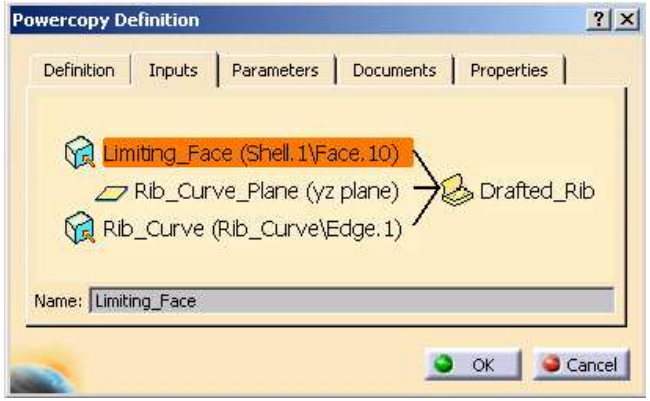
Student Notes:

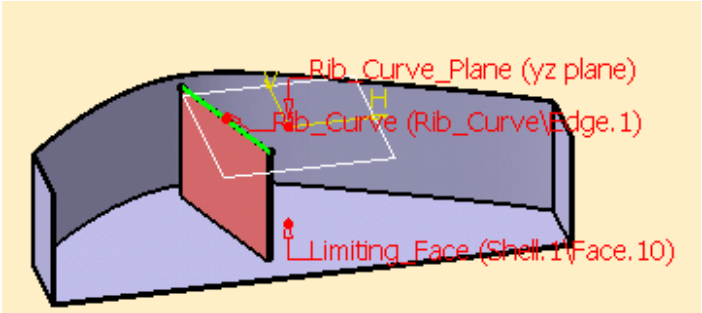
Example of PowerCopy (1/3)





In this example, we want to create a 'PowerCopy' which will require only a single 'Line' and 'Plane' as an input, and create a 'Drafted Rib' from it.

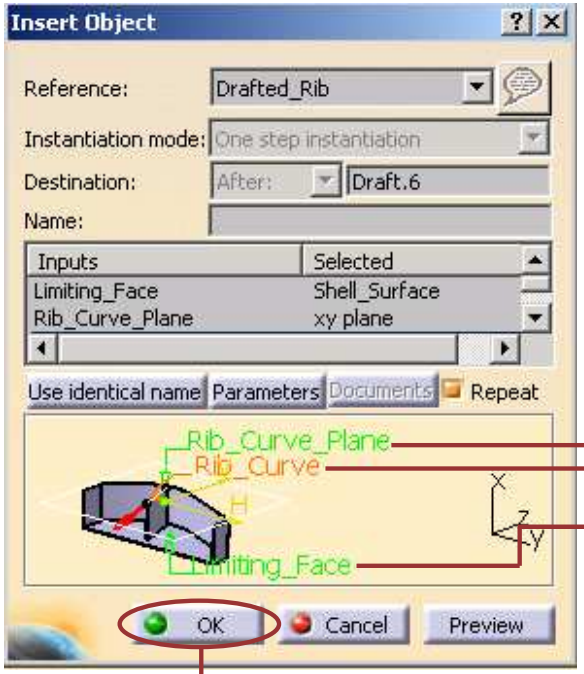




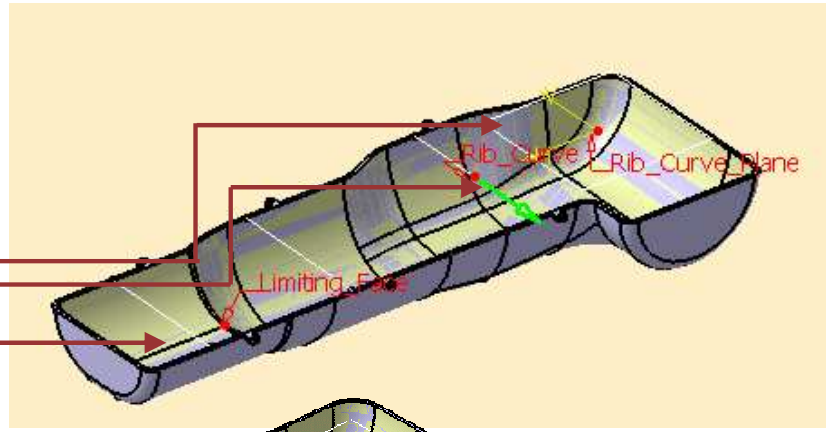
These are the inputs that the user will specify during the instantiation of the 'PowerCopy'.

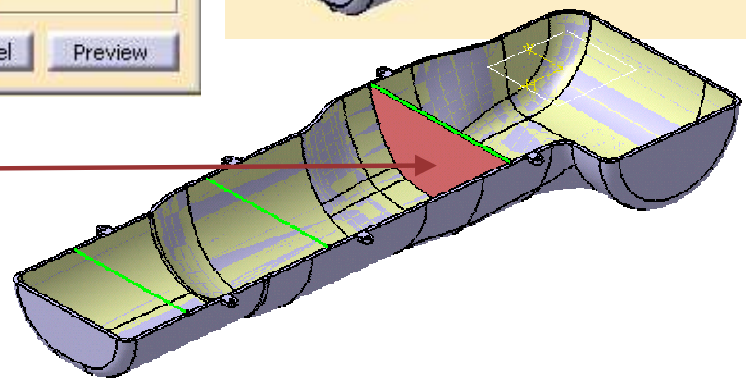
Student Notes:

Example of PowerCopy (2/3)



During the instantiation of the 'PowerCopy', the user has to select the inputs with respect to the destination part.

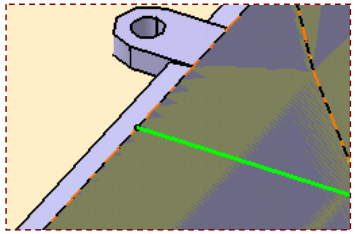


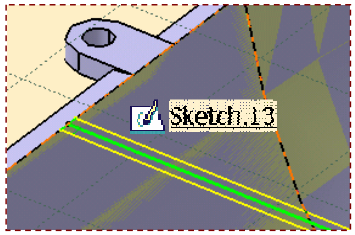


Student Notes:

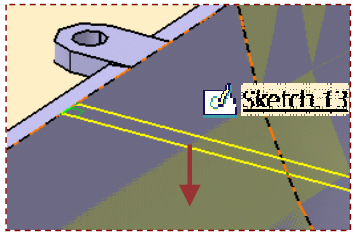
Example of PowerCopy (3/3)

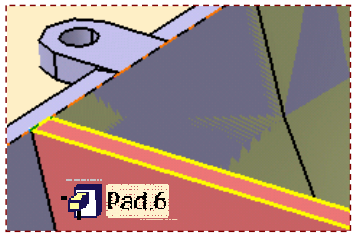
In this case, these are the geometries that the 'PowerCopy' feature creates automatically.



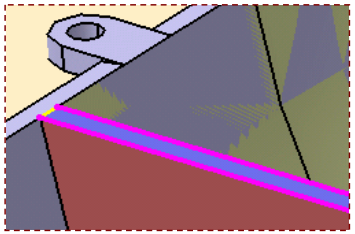


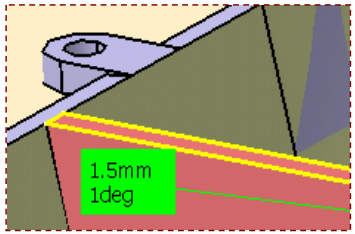
Creation of rectangular sketch from the selected rib line.





Extrusion of this sketch up to the selected 'Limiting Surface'.





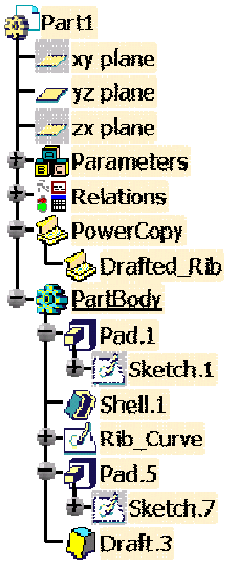
Application of 'Draft' to the extruded faces.

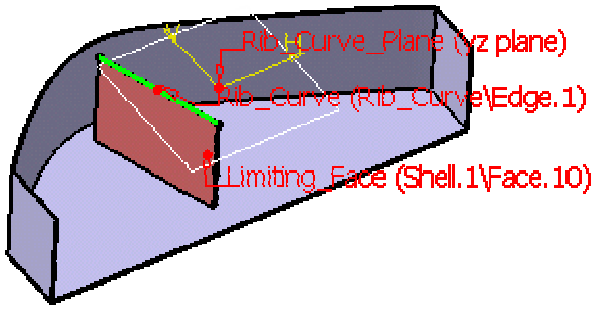
Thus, in this example you have seen how a PowerCopy feature can create a 'Drafted Rib' from a single 'Line' as input.

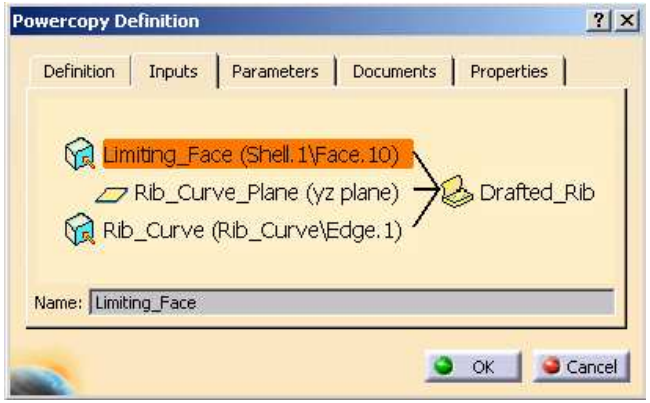
Student Notes:

Creating a PowerCopy

You will learn how to create a PowerCopy.





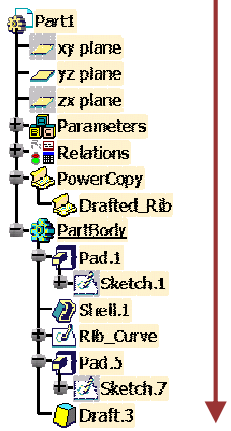


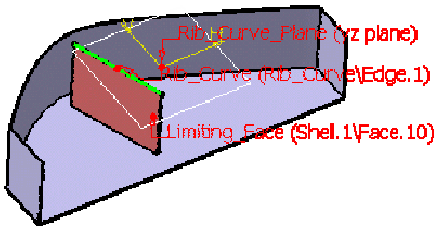
Student Notes:

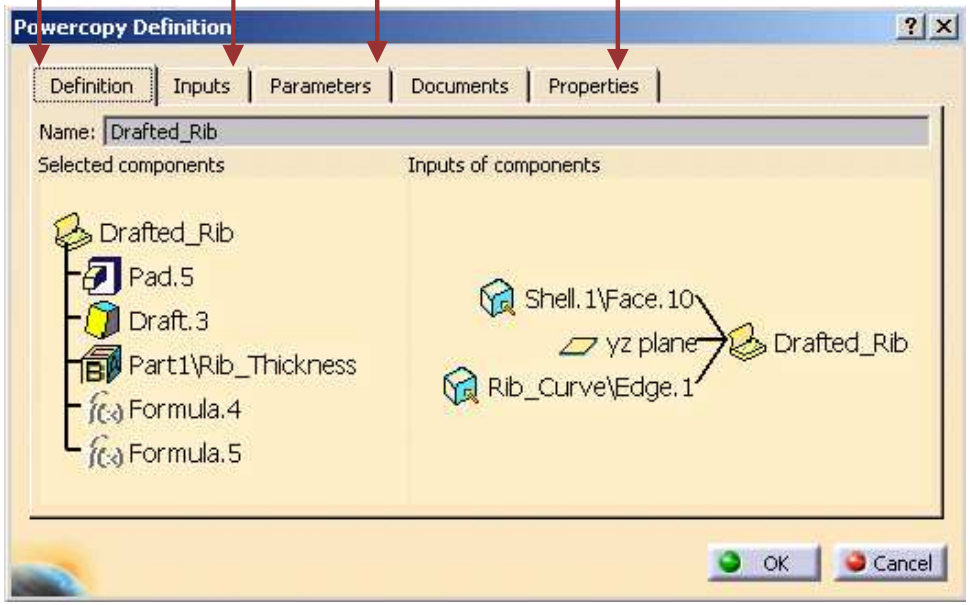
Process for PowerCopy Creation

Creation of PowerCopy consists of the following steps:

0. Making the Part ready for the creation of the PowerCopy
1. Setting Definition
2. Identifying and naming inputs
3. Publishing Parameters
4. Setting Icon and preview Properties







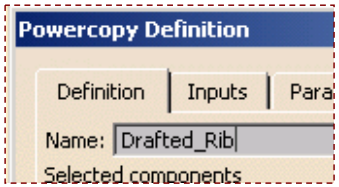
Student Notes:

How to Create a PowerCopy (1/5)

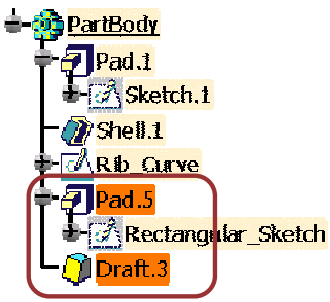
Once you have the right geometry in your CATPart, you can create the PowerCopy.

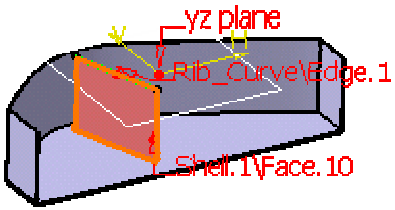
1a Select PowerCopy from the menu. (Insert > Knowledge Templates > PowerCopy)

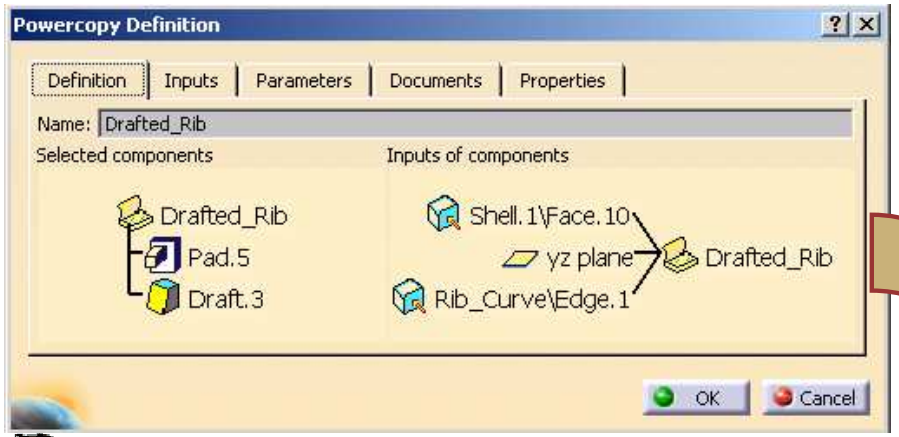
1b Type the name of the PowerCopy in the 'Definition Tab' of the 'PowerCopy Definition' dialog box.



1c From the specification tree, select the features that will make your 'PowerCopy'.





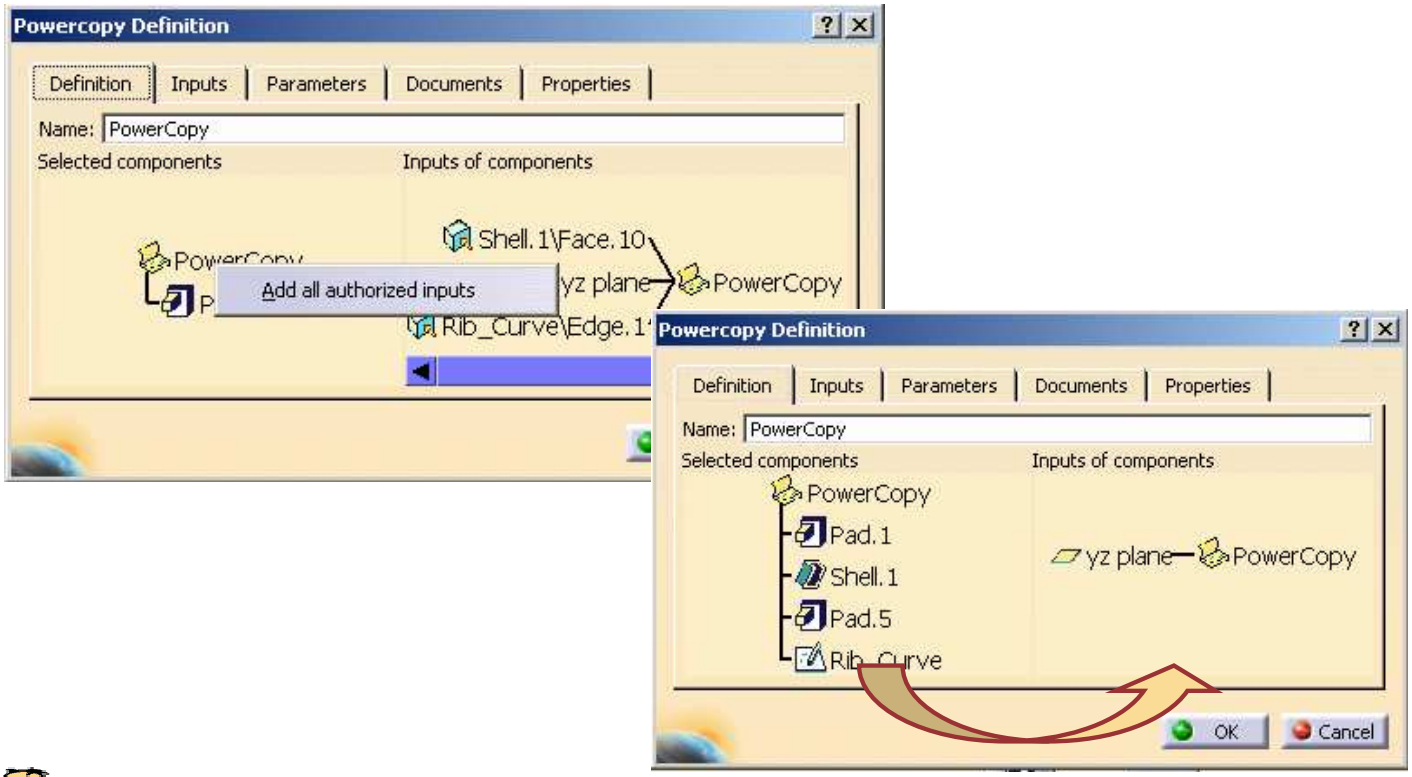


On selecting the features, the 'Inputs of components' are identified. These depend upon the features that you select to make your PowerCopy.

Student Notes:

How to Create a PowerCopy (2/5)

TIP: The contextual menu 'Add all authorized inputs' allows you to select all the possible components that can be created using minimum number of inputs.



Note that this tip is not used in this scenario. However, it can be used to select all the features and to later move the features from the 'Selected Components' field to 'Inputs of Components' field.

Student Notes:

How to Create a PowerCopy (3/5)

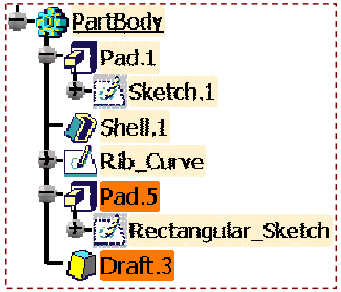
After selecting the features that make the PowerCopy, you can give names to the geometric inputs. During instantiation, the user will be prompted to select the geometries based on these new names.

In our case there are three inputs:

A. The edge (Edge.1) from 'Rib_Sketch' - > Using this sketch, the PowerCopy creates the 'Rectangular Sketch'.

B. The YZ plane on which the 'Rib_Curve' has been created.

C. The shell face (Face.10) up to which the 'Pad.5' was extruded.



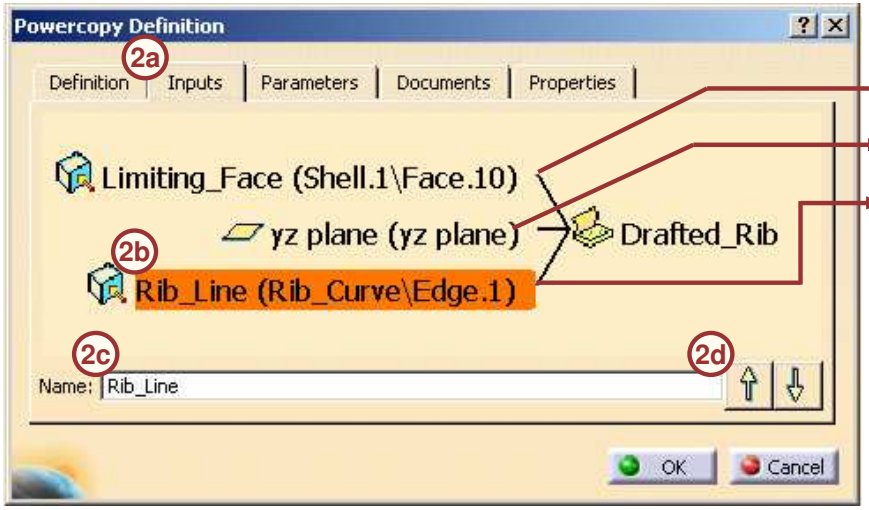
Let us give new names to these inputs from instantiation point of view.

2a Select the Inputs tab

2b Select the input to be renamed

2c Type a new name for the input

2d Using the arrow keys reorder the inputs, if required



New Name:

Limiting_Face

Rib_Curve_Plane

Rib_Curve

Reordering the inputs is sometimes required for displaying the inputs in a specific order in the PowerCopy instantiation dialog box.

Student Notes:

How to Create a PowerCopy (4/5)

After renaming the geometric inputs you can publish the parameters. During instantiation, the user can specify values for these published parameters.

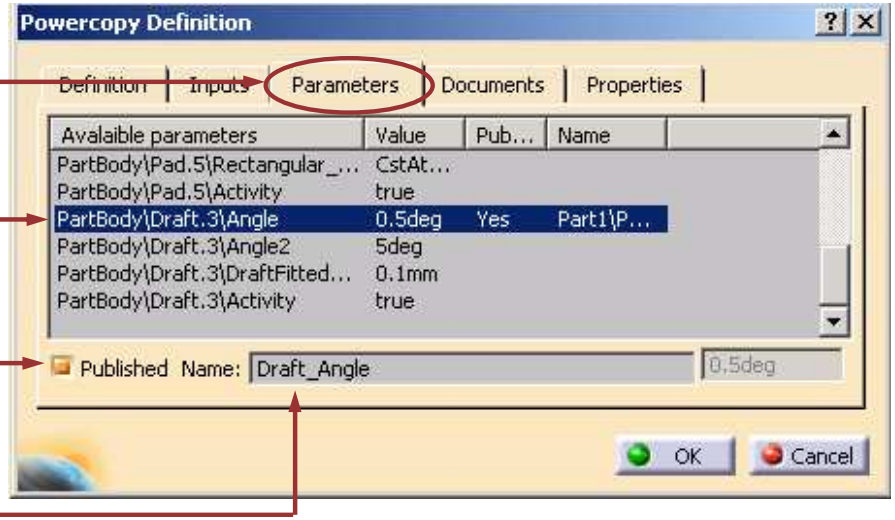
To publish the parameters,

3a Select Parameters tab

3b Select the parameter

3c Check the 'Published' option

3d If necessary, rename the parameter

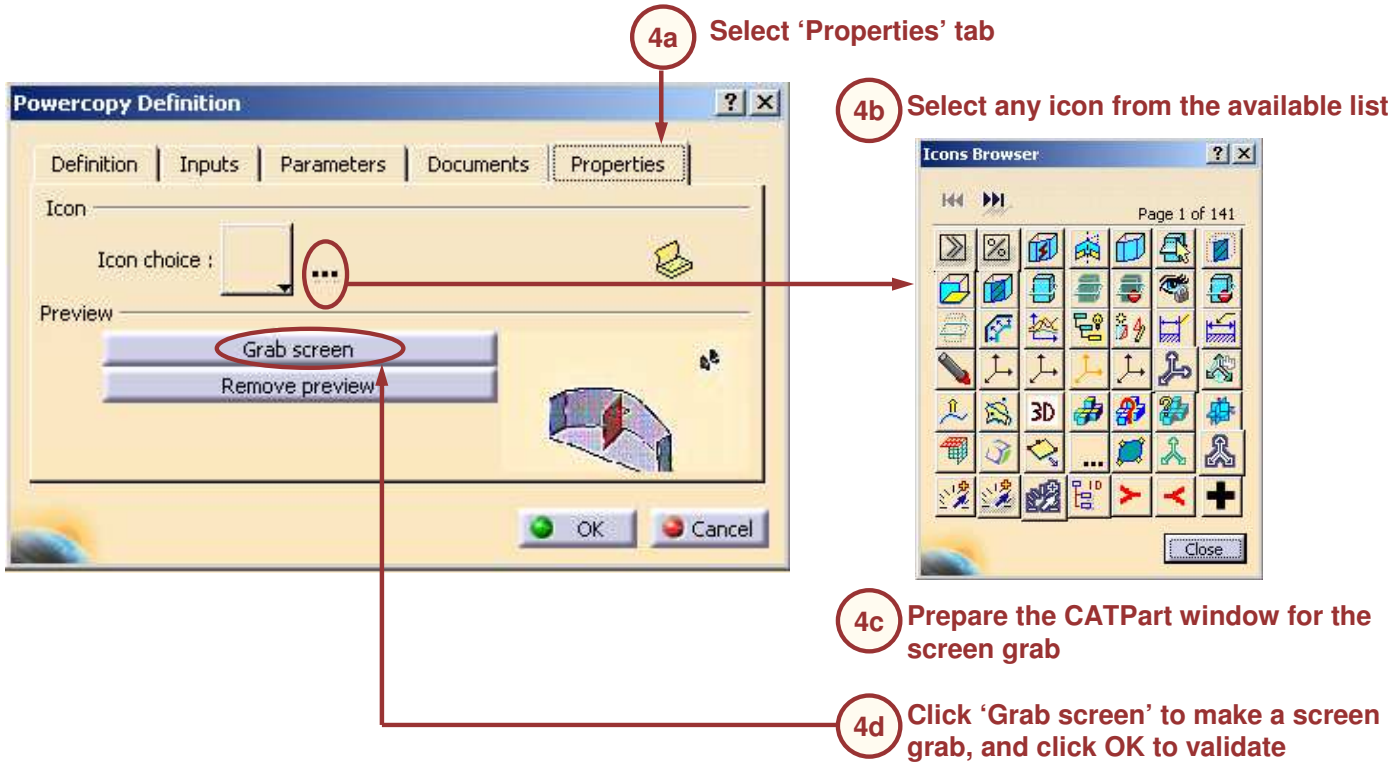


Note that it will be easier for you to recognize the parameters if you have already renamed them with the knowledgeware tools. [f(x)]

Student Notes:

How to Create a PowerCopy (5/5)

Once the parameters are published, you can select the icon for your PowerCopy and make a screen grab to create a preview of your PowerCopy for catalogs.

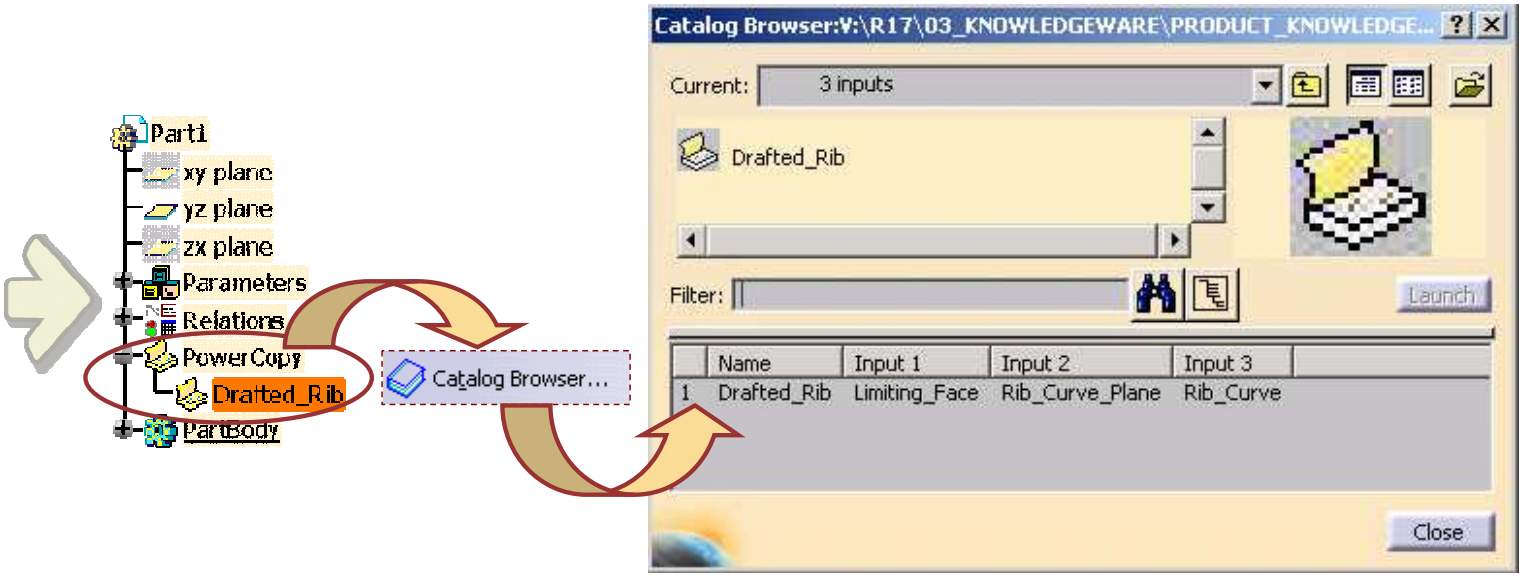


To prepare the screen grab, you can remove the tree and compass from the window and get the correct zoom and orientation.

Student Notes:

Saving a PowerCopy

You will learn how to save the PowerCopy in a catalog.



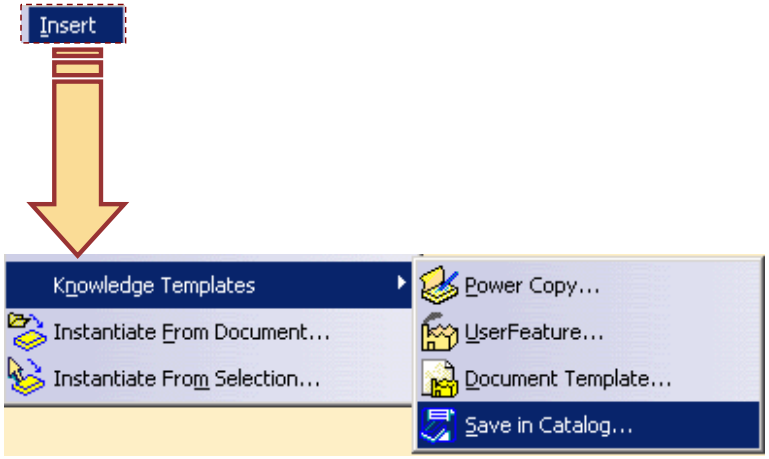
Student Notes:

Saving a PowerCopy

If you do not save the CATPart containing your PowerCopy, you will not be able to instantiate the PowerCopy.

You can save the PowerCopy in a new catalog and also in an existing catalog.

You can also update a catalog which makes reference to the PowerCopies of your CATPart.



Student Notes:

How to Save a PowerCopy in a Catalog

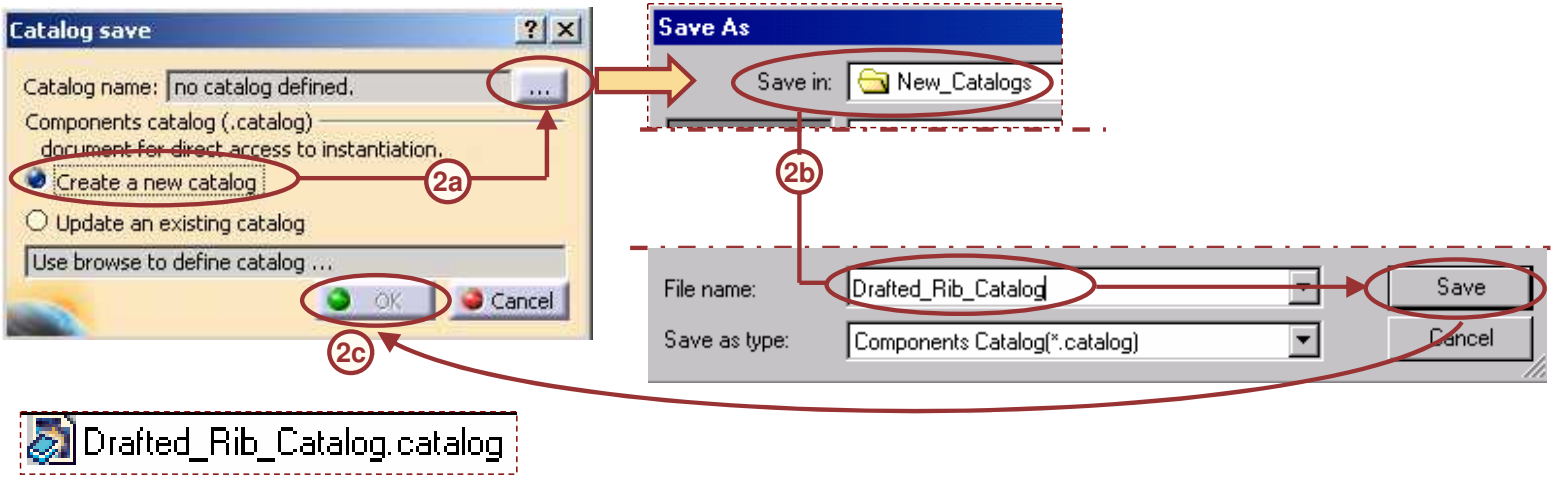
0 Save the CATPart containing your PowerCopy.

1 From the menu, select – Insert > Knowledge Templates > Save in Catalog.

2a Select the 'Create a new catalog' option and click the browse button (. . .) to define the path for new catalog.

2b Select the correct path, type the new name of the catalog and click Save. (The OK button of the 'Catalog save' dialog box will now be active)

2c Now click OK to the 'Catalog save' dialog box.

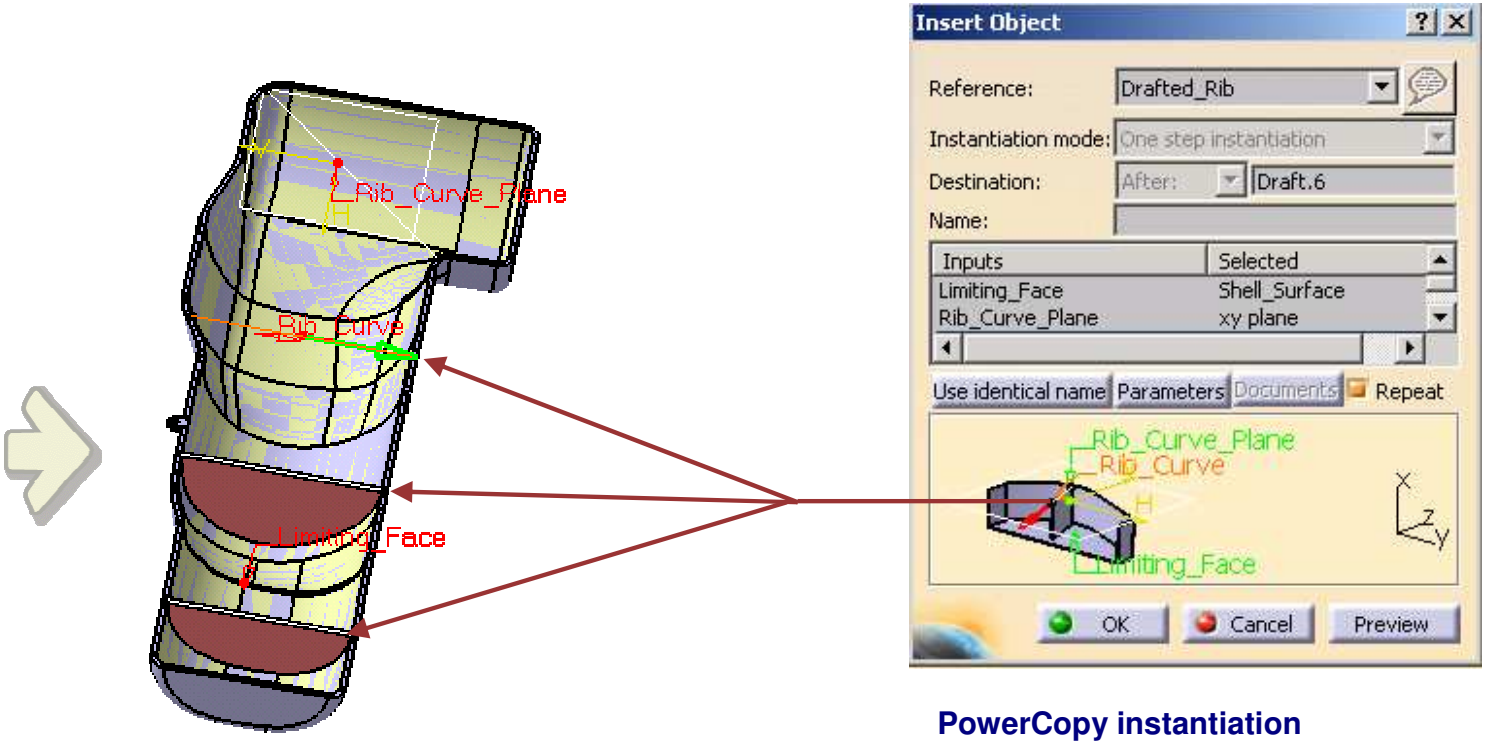


Drafted_Rib_Catalog.catalog

Student Notes:

Instantiating a PowerCopy

You will learn how to instantiate a PowerCopy differently at different places by varying the geometric inputs and the parameters while instantiating.



PowerCopy instantiation

Student Notes:

How to Instantiate a PowerCopy (1/4)

The first step of PowerCopy instantiation is accessing the PowerCopy.
You can access it:
a) From the CATPart file containing it.
b) From a catalog having its reference.

You can also use a VB macro to instantiate the PowerCopy. Refer CATIA online documentation for more information.

Before proceeding, please save all the CATIA documents that are attached to this screen to a local folder.

0 Open the CATPart in which you want to instantiate the PowerCopy.

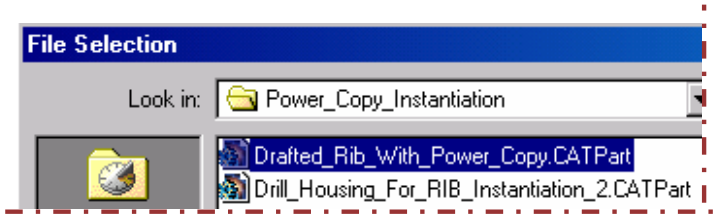
1 From the menu, select:

Insert > Instantiate From Document

OR

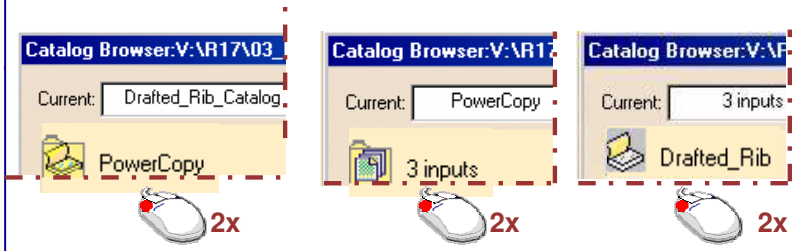
2 Select the CATPart file which contains your PowerCopy.

OR



1 Click on Catalog Browser and browse for the catalog.

2 After opening the catalog, double-click on 'PowerCopy', then on '3 inputs' and finally on 'Drafted_Rib' to open the instantiation dialog.

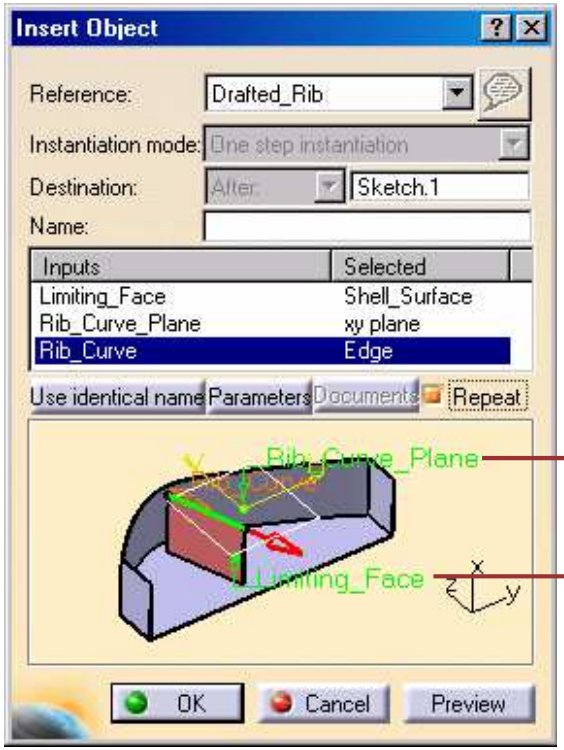


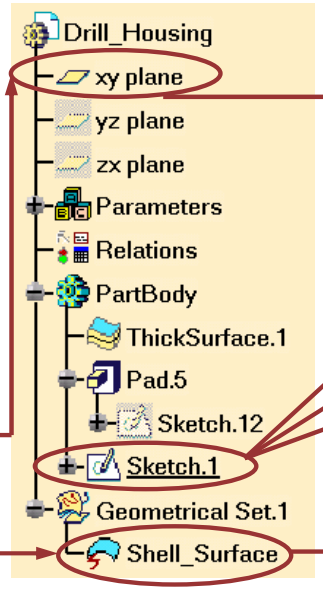
Student Notes:

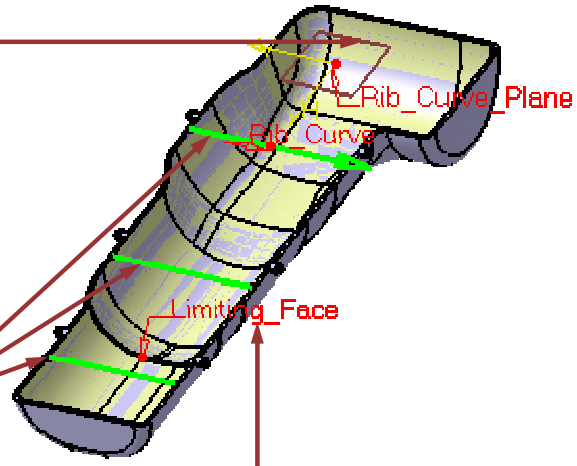
How to Instantiate a PowerCopy (2/4)

The second step of instantiation is selecting the geometric inputs of the PowerCopy.

3a **Select the geometric inputs of the PowerCopy as shown. For this example, select the 'Limiting Surface' and 'Rib_Curve_Plane' as shown.**







Now the first two inputs remain the same for all the three green 'Rib_Curves'. So in this case, you can use the 'Repeat' option.

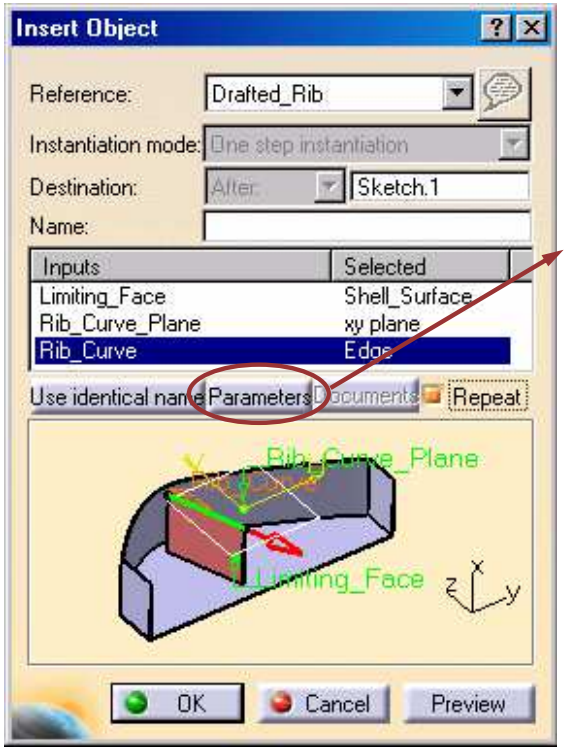
3b **Select the 'Repeat' option, select any one of the three green lines and click OK. Repeat the same process for any one of the remaining two green rib lines.**

Student Notes:

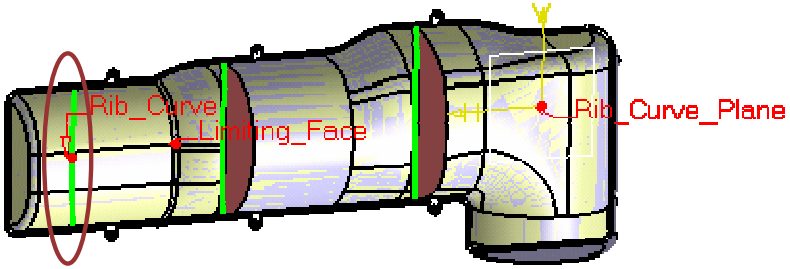
How to Instantiate a PowerCopy (3/4)

You can also change the values of the parameters that you have published during the PowerCopy creation.

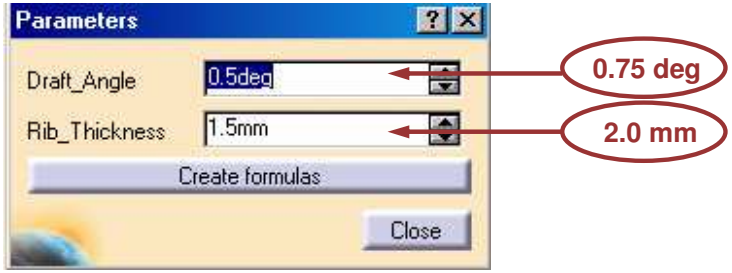
In this example, we will enter different values for the last rib line.



4a **Select the remaining Rib_Curve and click the 'Parameters' button.**



4b **Enter the values for the parameters as shown and close the 'Parameters' dialog box.**

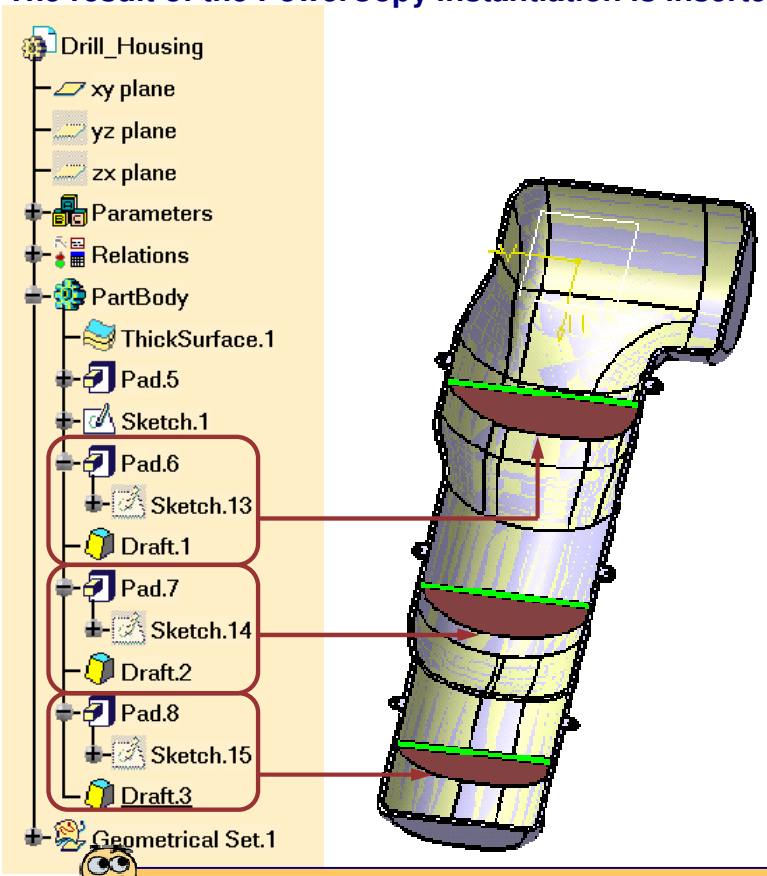


4b **Click OK on the 'Insert Object' dialog box to instantiate the last rib, and then click 'Cancel' to dismiss it.**

Student Notes:

How to Instantiate a PowerCopy (4/4)

The result of the PowerCopy instantiation is inserted after the "in work object".



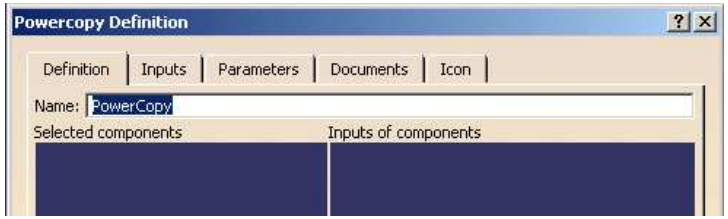
The result of PowerCopy instantiation is a set of editable features. They are not linked to the original features of the PowerCopy CATPart.

Student Notes:

To Sum Up ...

You have learned:

- **What is a PowerCopy**
  - **A PowerCopy is a set of design features grouped together to be reproduced. It is an advanced copy tool. PowerCopy tools are available in the Insert menu in Part design, Wireframe and surface, and Sheet metal design workbenches.**



- **How to create a PowerCopy**
  - **During creation, you have to set the definition, identify and name the inputs, publish the parameters, choose an icon and preview.**
- **How to save a PowerCopy**
  - **Saving a PowerCopy is necessary. If not saved, a PowerCopy can never be instantiated. This can be done through Insert menu > Advanced replication tools > Save in catalog.**
- **How to instantiate a PowerCopy**
  - **For instantiation, you have to first select a previously created PowerCopy. This can be done in two ways. The first way is through a catalog, and the second way is from Insert menu > Instantiate from document.**

Student Notes:

Creating and Using User Defined Features

You will become familiar with the use of advanced replication tools called User Defined Feature.

- **User Defined Features: Presentation**
- **Creating a User Defined Feature**
- **Saving a User Defined Feature**
- **Instantiating a User Defined Feature**
- **UDF Meta Inputs**

Student Notes:

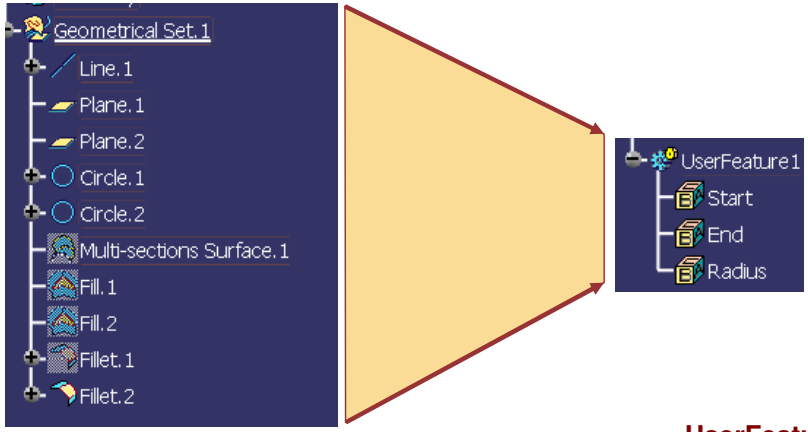
User Defined Features: Presentation

You will learn what are the benefits of advanced replication tools called User Defined Features.

Student Notes:

User Defined Features vs PowerCopies

- **User Defined Features (also called UDF) are similar to PowerCopies (at definition stage).**
- **But when instantiated, you get only one feature like any other V5 feature.**



UserFeature

PowerCopy

PowerCopies are mainly used to accelerate the dramatically design generation

User Defined Features allow the customer to manipulate their own semantic objects in the place of V5 standard objects

It introduces a true component-based approach for building designs

Student Notes:

What is a User Defined Feature? (1/2)

- A User Defined Feature is a template that works at the part level. From a collection of features (geometries, literals, formulas, constraints, etc.), the user can create his/her own feature. The result is a Part Design feature or a Wireframe and Surface feature that can be reused in the design of another part. The created feature can be saved in a catalog.

- A User Defined Feature:
  - Allows you to create applicative features
  - Allows you to hide design specifications and preserve confidentiality (for instance, to sub-contractors)

- The User Defined features (like a line for Drafting or a check for Knowledge Advisor) are open and shareable objects. This capability significantly increases the potential application of the user defined features, since it enables you to:
  - Find the user defined features by attributes
  - Generate the user defined features with the scripting language to simplify the process of creating scripts
  - Define the expert rules that work on user defined features with Knowledge Expert
  - Use the user defined features in Knowledge Advisor reactions
  - Develop the CAA functions based on the user defined variables

Student Notes:

What is a User Defined Feature? (2/2)

- **A UserFeature is a design feature made up of a group of other design features.**
  - **You can edit it (set contained features, entries, previews …)**
  - **You can instantiate and customize it in the design of any part**
  - **Instance of a UserFeature is a black box (users do not have any access to its contents)**

- **The UserFeature tools are available in the Insert menu (Knowledge Templates) of the following workbenches:**
  - **Part Design**
  - **Generative Sheetmetal Design**
  - **Generative Shape Design**

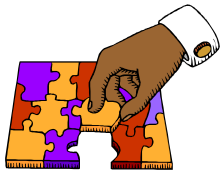
Student Notes:

User Defined Features Benefits

- **Simplification of designs and better evolutivity**
  - **The complexity in terms of the number of features used in a model is reduced**
  - **The component generated as a UserFfeature is easier to understand and modify (edit…)**
  - **Designers do not lose time in dealing much with the geometry**

- **Insurance of best practices usage**
  - **Rules embedded in the UserFeatures cannot be violated within a company**

- **Intellectual Property Protection**
  - **While exchanging a model using the UserFeatures, it is impossible to understand what is inside for the receiver (even if he can update it)**



- **Object-oriented designs**
  - **UserFeatures inherit from features standard behaviors**
  - **UserFeatures are recognized by Knowledgeware as any other V5 object**
  - **Behaviors can be added to a userfeature (thanks to reactions)**

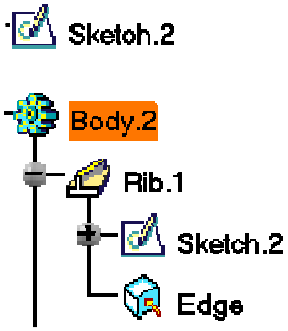
Student Notes:

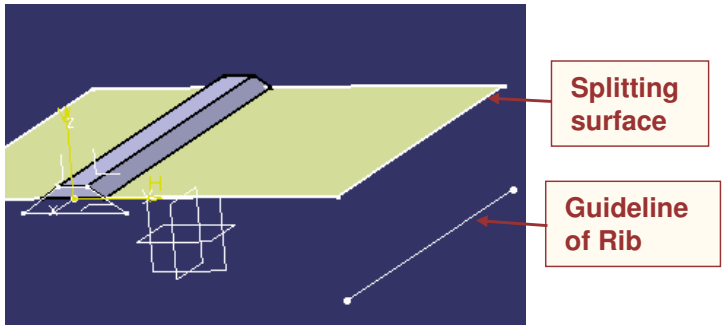
Example of User Defined Feature (1/2)

The UserFeature that we are going to create and instantiate is made up of…

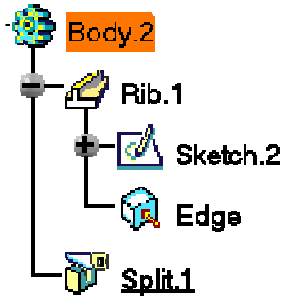
…one sketch (lying on a face of an axis system)…

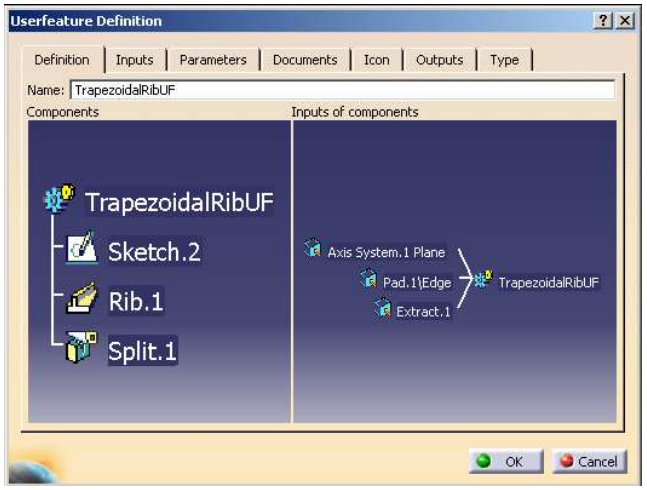
… one rib, based on this sketch and guided by a line inside an external body …





… and one split of the body by a surface



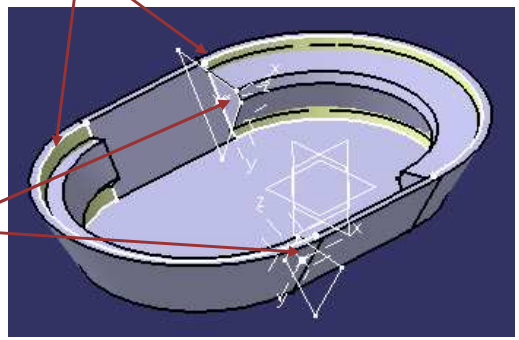


Student Notes:

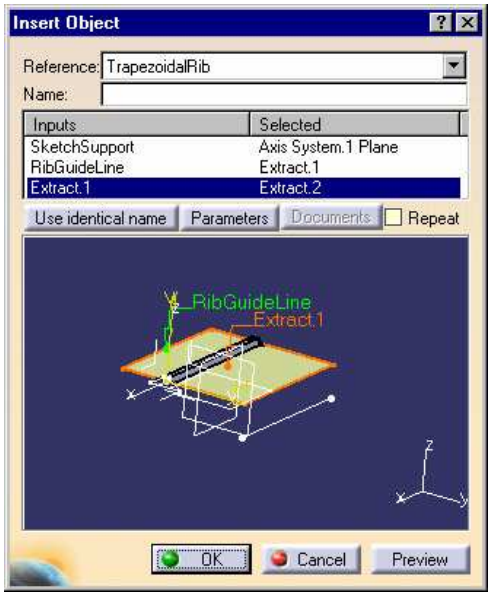
Example of User Defined Feature (2/2)

- **While instantiating the UserFeature, you will be able to customize:**

…The Inputs of the geometric data…



… The values of Top-Length, Top-Height and Rib-Angle parameters…



- **Instance of the userfeature will give out two geometric outputs: the main result and the profile sketch of the rib**

Student Notes:

Creating a User Defined Feature

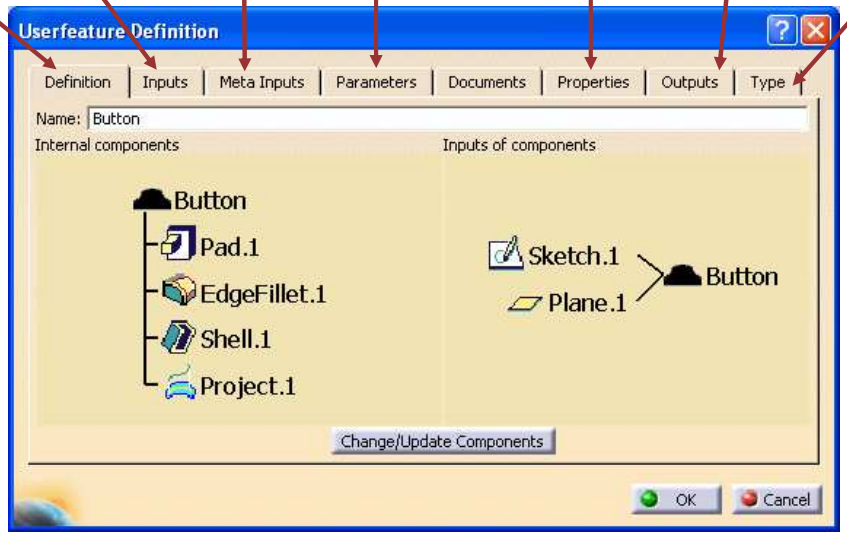
You will learn how to group the existing features in a black box in order to reuse them in another context.

Student Notes:

Process for UDF Creation

- **User Defined Feature creation process includes several steps:**

1. Selecting the existing features
2. Naming the input geometry
3. Optional: Selecting the meta inputs
4. Optional: Selecting and naming the public parameters
5. Optional: Selecting the icon and creating the preview
6. Optional: Selecting the outputs
7. Optional: Creating a new Type to define the UDF



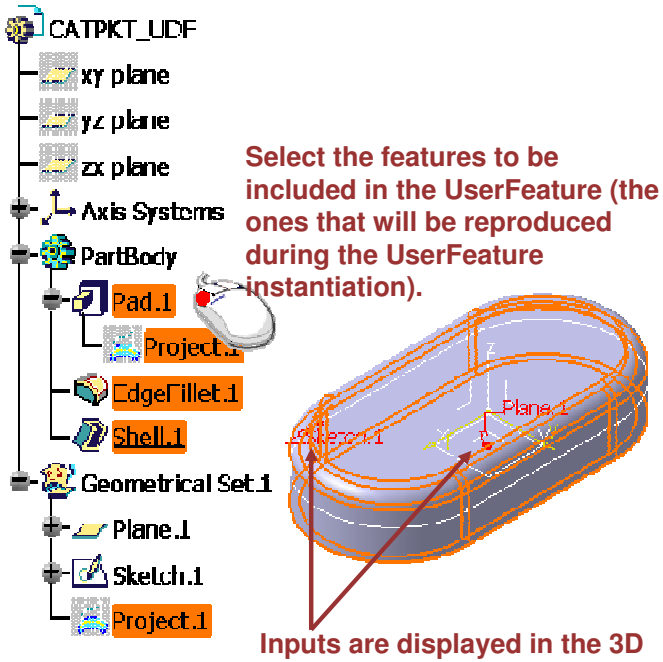
Student Notes:

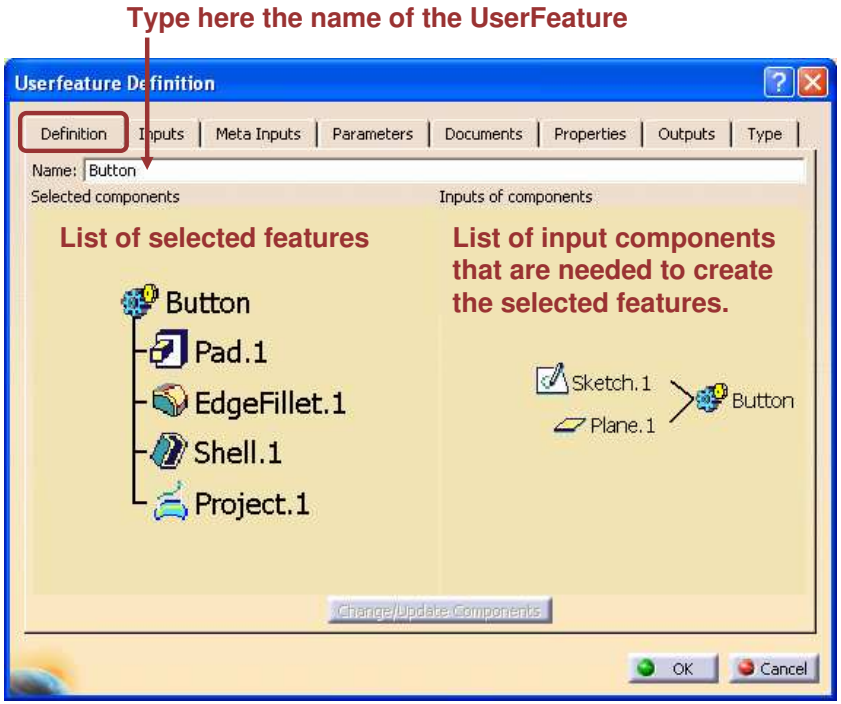
Creating a User Defined Feature (1/9)

Open the User Defined Feature Definition panel by clicking on the Create User Feature icon in the Product Knowledge Templates workbench.

The following panel appears.

The Definition tab allows you to key in the name of the UserFeature and see the features selected for its definition.

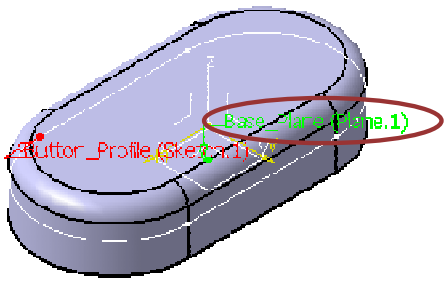




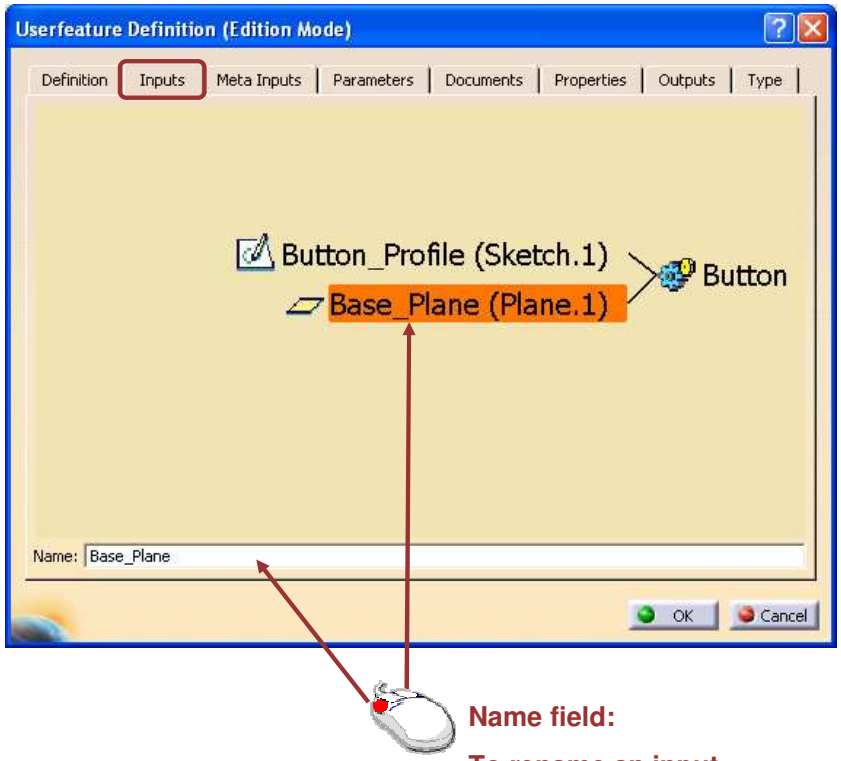
Student Notes:

Creating a User Defined Feature (2/9)

The Inputs tab allows you to see and rename the required geometric inputs for the instantiation of the userfeature.



While editing the inputs tab, required geometric inputs are shown in the geometry.



Name field:

To rename an input, select it, select the name field and key in the new name.

Student Notes:

Creating a User Defined Feature (3/9)

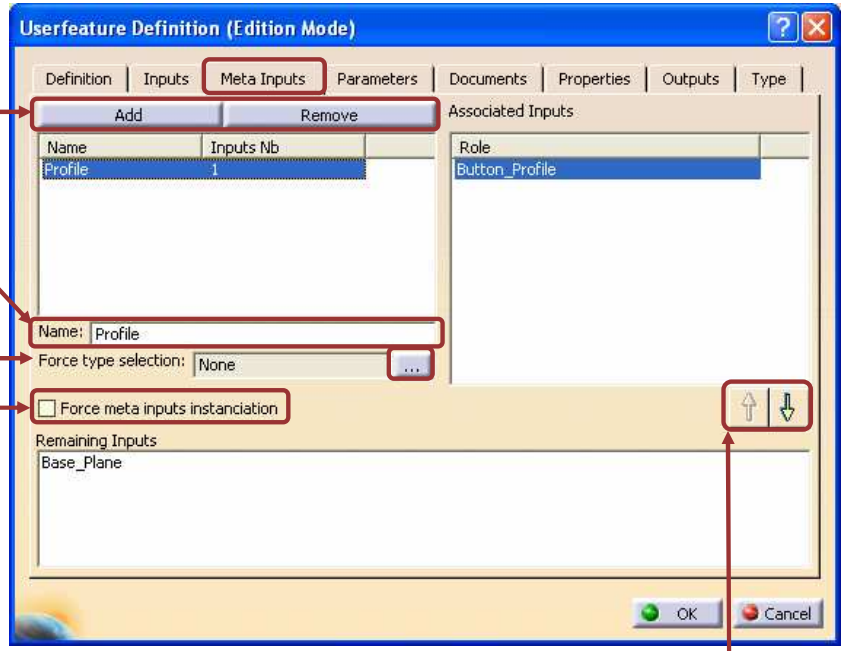
The Meta Inputs tab allows you to define the meta inputs and their association with real inputs. It also allows you to optionally associate a Type with each meta input.

The Add/Remove buttons enable you to add or remove the meta inputs.

The Name field enables you to enter the name assigned to the meta input.

The ... button enables you to associate a type to the meta input.

The Force meta inputs instantiation option enables you to decide if you want the user to select the instantiation mode. If checked, only the Meta inputs instantiation mode will be available in the Insert Object dialog box. If unchecked, the user will be able to choose the instantiation mode he wants to use i.e. the Meta inputs instantiation mode or the Meta inputs normal instantiation mode.



The arrow buttons enable you to remove or add the inputs from the list of inputs that makes the meta input.

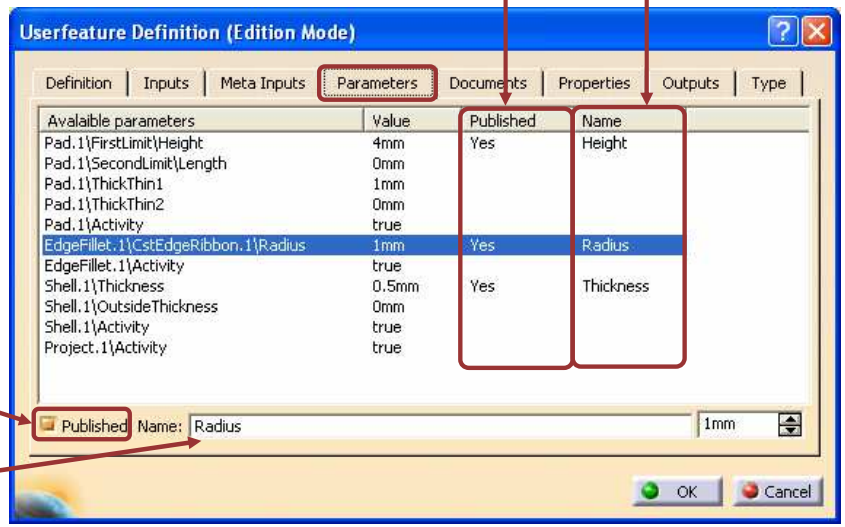
Student Notes:

Creating a User Defined Feature (4/9)

The Parameters tab allows you to see all the parameters participating in the definition of the Userfeature and allows you to make them Published for instantiation.

Published status column

Published name column



Check this button to make the selected parameter public.

Type here an explicit name for the published parameter.

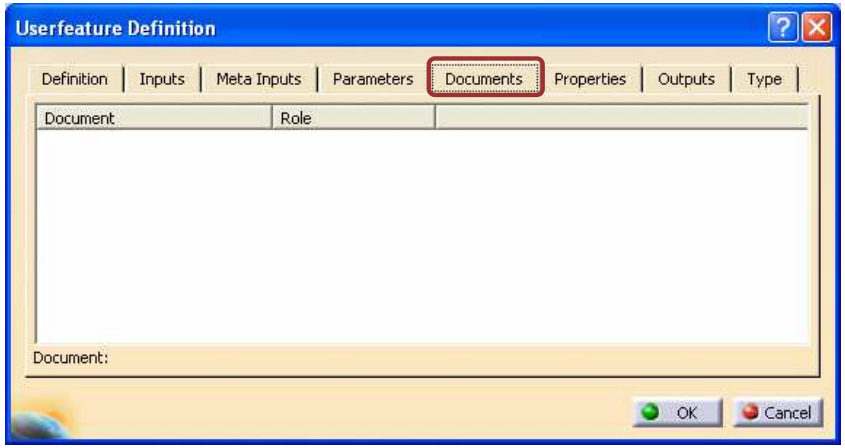
Parameters have the same names as in the formulas editor, if you want to recognize them easily, rename them with knowledgeware common tools.

Student Notes:

Creating a User Defined Feature (5/9)

The Documents tab shows the complete path and role of design tables referenced by an element included in the UserFeature.

This tab does not exhibit any document because only the design tables belonging to the selected object are displayed. While instantiating or editing the UserFeature, you will be able to change the document pointed by the internal design table.



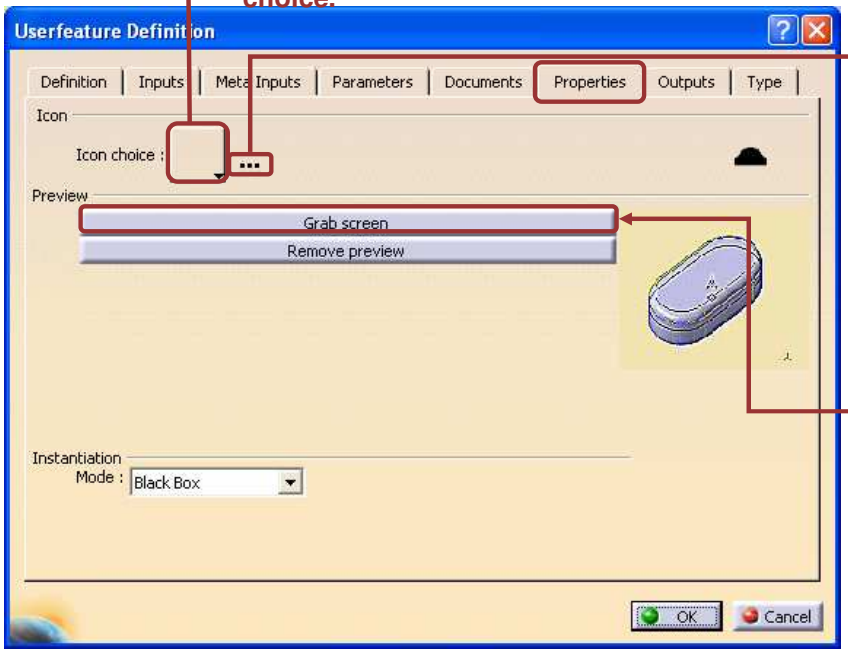
Student Notes:

Creating a User Defined Feature (6/9)

The Properties tab allows you to modify the icon identifying the UserFeature in the specification tree.

A subset of icons is available from the Icon choice.

Click on ... to open the Icon Browser. You will have the choice between all icons that are loaded in your CATIA session.



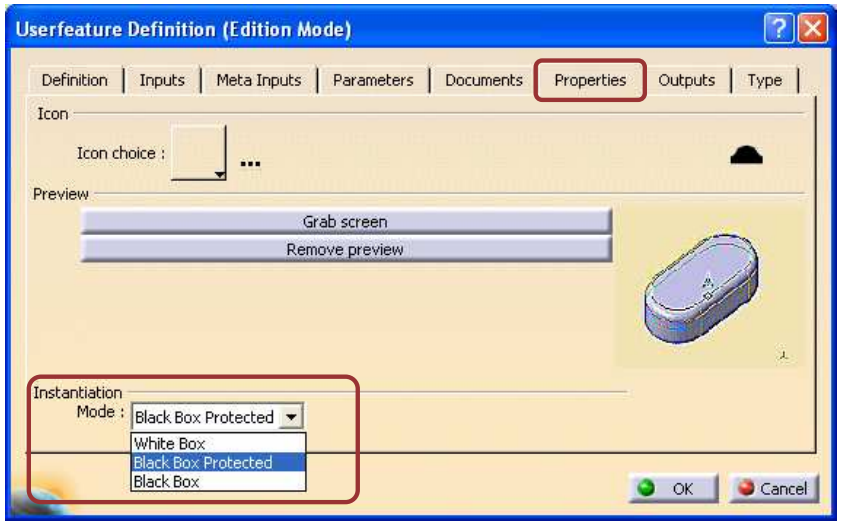
Click on Grab screen button to capture an image of the UserFeature that will be stored with its definition.

This preview will be useful while referring to a UserFeature in a catalog…

Student Notes:

Creating a User Defined Feature (7/9)

The Instantiation Mode combo box list enables you to choose the view that will be created at instantiation.

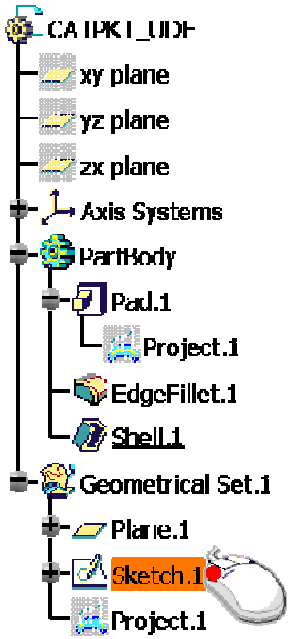


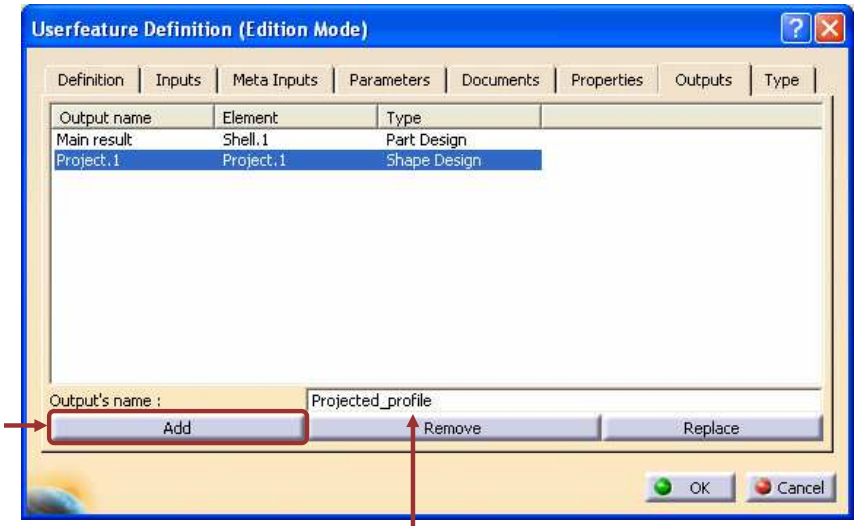
- **Select the White Box mode if you want the end-user to display the UserFeature internals.**
- **Select the Black Box mode if you want the end-user to be able to lock and unlock the UserFeature instance.**
- **Select the Black Box Protected mode if you do not want the end-user to access the internals. This mode is the standard User Defined Feature view.**

Student Notes:

Creating a User Defined Feature (8/9)

The Outputs tab allows you to select geometric outputs other than the Main result for instantiation.





Click on Add button…

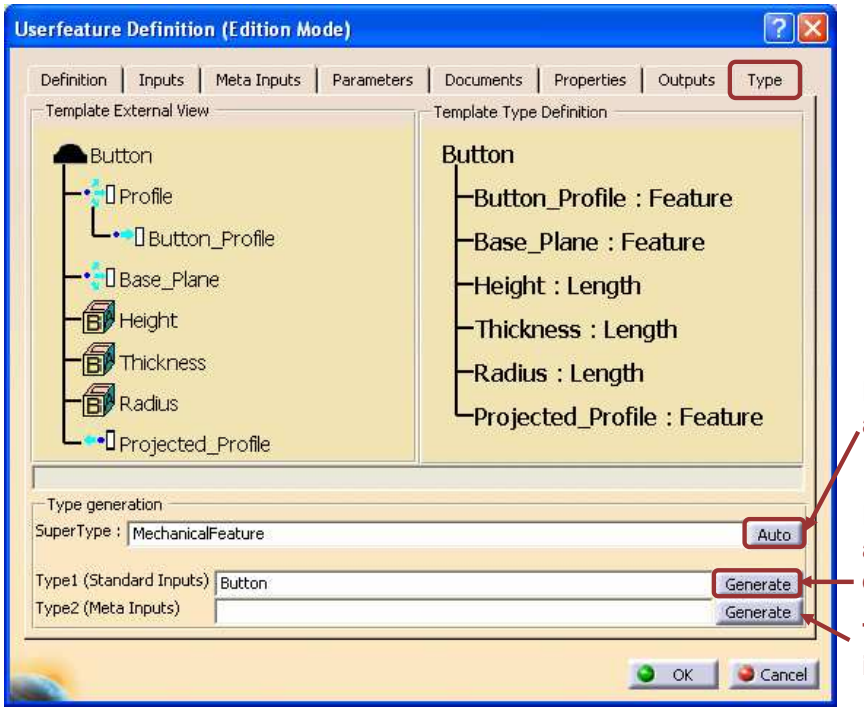
…and select in the tree the element of the UserFeature you want to recover with instantiation

Key in a new output name after output selection.

Student Notes:

Creating a User Defined Feature (9/9)

The Type tab provides you a way to associate a Type with a UserFeature. This can be used in search operations, expert checks, and to instantiate UDFs using Knowledge Pattern Functions.



First click the Auto button to have the super type automatically displayed by the application

Enter the name of the type that you want to assign to the UserFeature (button in this example) and click the Generate button.

Type2 allows you to define a type for the meta input.

If you want to reuse the generated type in another CATIA session, save the CATGScript file in the Directory indicated in the Reference Directory for Types field (see Tools>Options>Parameters and Measure>Knowledge Environment tab).

Student Notes:

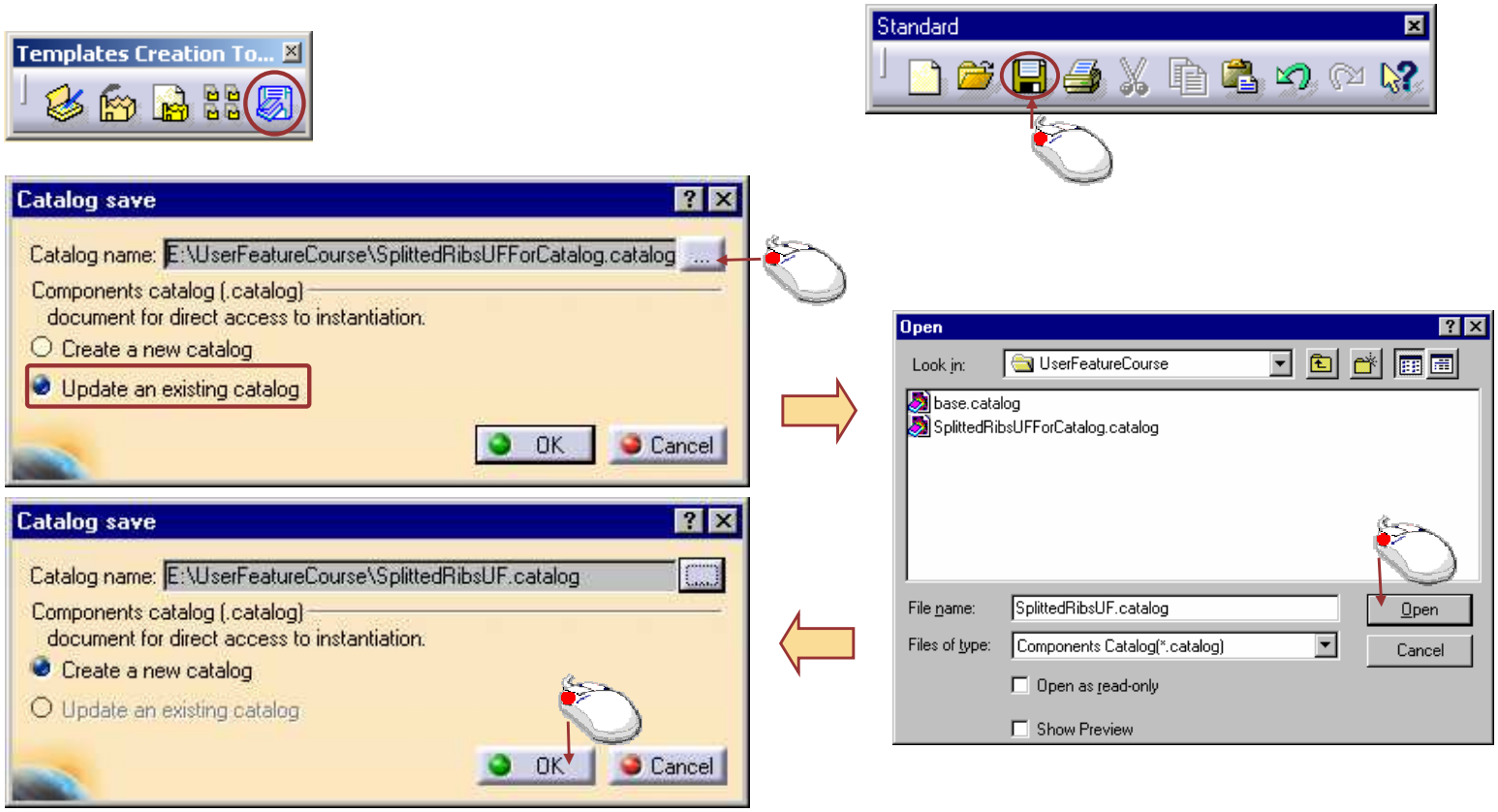
Saving a User Defined Feature

You will learn how to store a User Defined Feature in a catalog document in order to make it available for other users.

Student Notes:

Saving a User Defined Feature

You are to save the CATPart file containing the UserFeature, but you can also reference all the userfeatures of the edited CATPart in a catalog by using one of the UserFeature Tools.



Student Notes:

Instantiating a User Defined Feature

You will learn how to import an existing User Defined Feature from a catalog in your document, and how to make it fit to the specifications of your design.

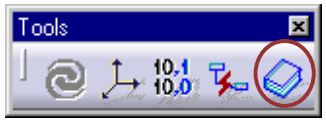
Student Notes:

Instantiating a User Defined Feature (1/3)

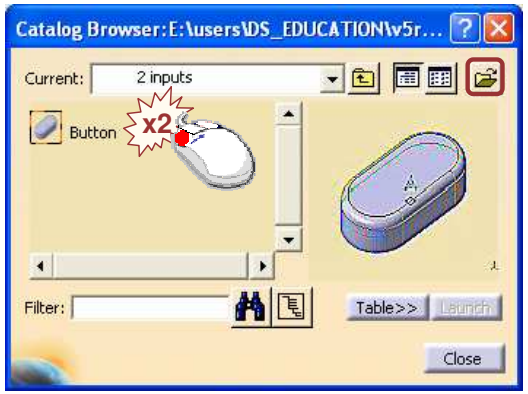
There are different ways to launch the instantiation of a UserFeature.

Select the Instantiate From Document option in CATIA Insert menu.
A document browser appears to allow you to select the document containing the UDF to instantiate.

If the document containing the UDF is already opened in a session:
Select Instantiate From Selection option in CATIA Insert menu.
Switch to the window of the document containing the UDF to instantiate and select the UDF in the tree.



If the UDF is stored in a catalog:
Open the CATIA catalog browser. Open the catalog containing the UDF to instantiate. Browse it in order to access the UDF component in the catalog.
Double-click on the UDF component to launch its instantiation.



You can also use a VB macro to launch the instantiation. Refer the CATIA online documentation for more information.

Student Notes:

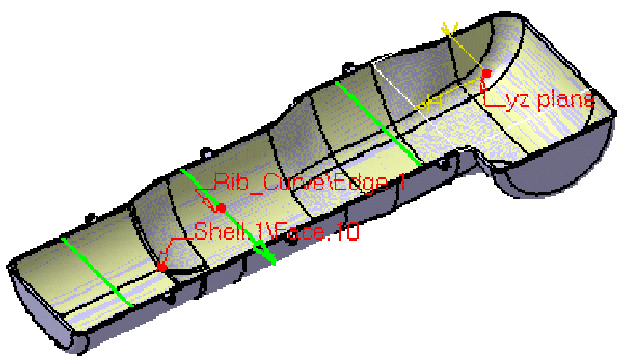
Instantiating a User Defined Feature (2/3)

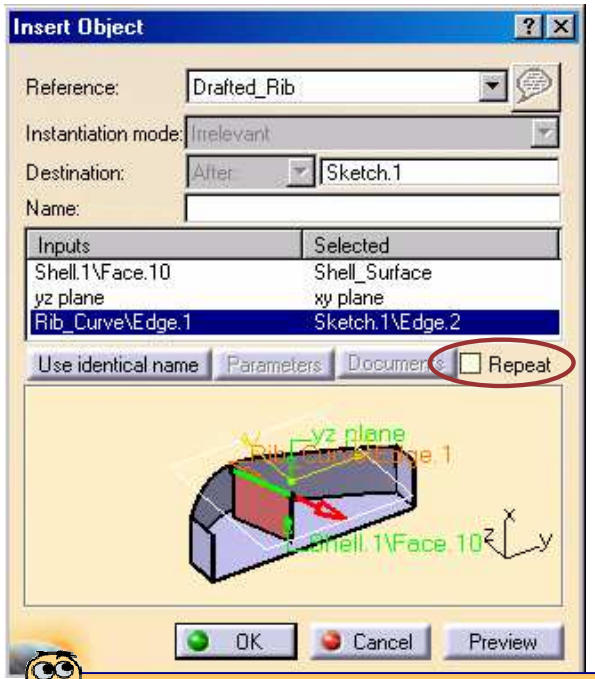
- **UserFeature instantiation is made of several steps:**
  - **Selection of the geometric inputs**
  - **Setting of the published parameters values**

Select the inputs in the receiving part. You can select them either from the graphic area or in the tree.

Click the Use identical name button to have the inputs automatically filled in. Use this option only if the inputs of the receiving document have the same name than the inputs of the UDF.

If needed, click on the arrow in the 3D to invert it.





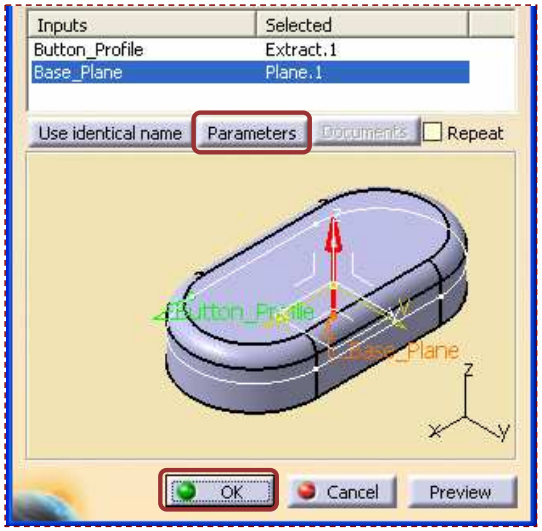
Check the Repeat button if you want to instantiate the UDF several times.

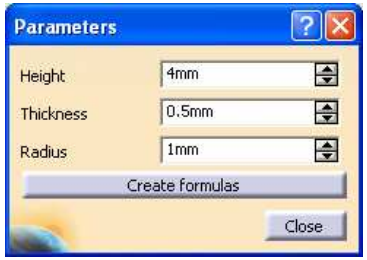
Be very careful to the orientation of the arrows while selecting the geometric inputs. It may change drastically the result of the instantiation.

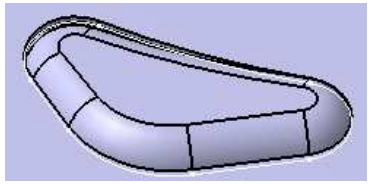
Student Notes:

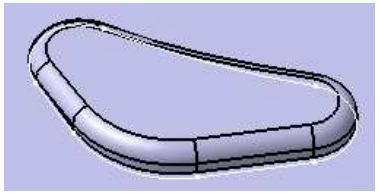
Instantiating a User Defined Feature (3/3)

Once all the geometric inputs are selected, the Parameters button allows you to change the value of the published parameters.







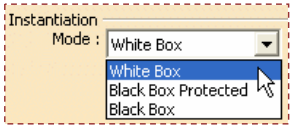


Click on Preview to check the result and then on OK to validate the creation of the UDF.

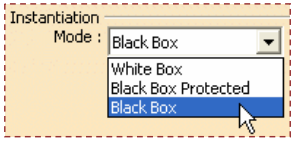
Student Notes:

Debugging a UDF

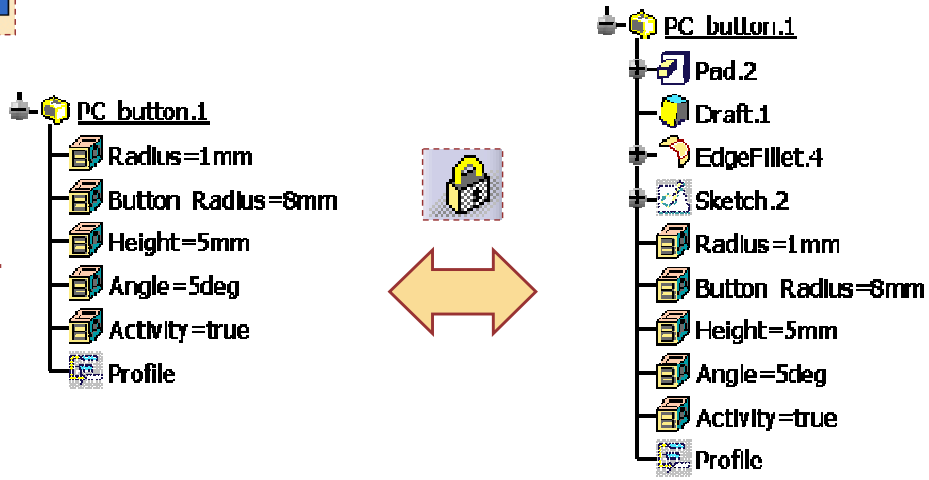
The UDF Debug icon allows you to visualize what is inside the User Feature instance. It allows you to switch between the expanded and the simplified view mode of the UDF.



If the instantiation mode of the UDF is White Box, you will visualize what is inside the UserFeature instance at instantiation.

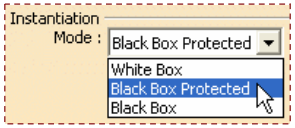


If the instantiation mode of the UDF is Black Box, the UserFeature instance view will be simplified at instantiation.



Simplified view. Black Box default.

Expended view. White Box default.

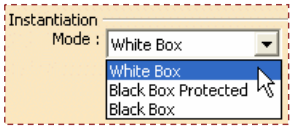


The Black Box Protected mode ensures a locked view of the UserFeatures, thus ensuring secure exchanges. In this mode, the UDF cannot be extended using the UDF Debug button.

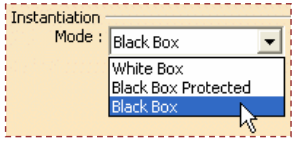
Student Notes:

How to debug a UDF (1/5)

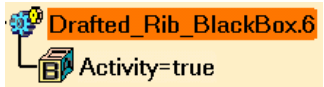
The UDF Debug icon allows you to visualize what is inside the User Feature instance. It allows you to switch between the expanded and the simplified view mode of the UDF.

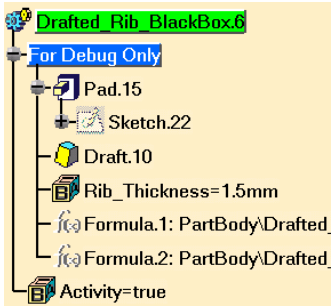


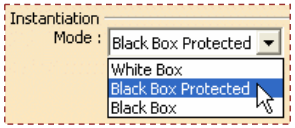
If the instantiation mode of the UDF is White Box, you can visualize the contents of the UserFeature instance for debug purposes.



If the instantiation mode of the UDF is Black Box, using the 'Udf Debug' tool, you can expand and visualize the contents of the UDF instance.







The Black Box Protected mode ensures a locked view of the UserFeatures, thus ensuring secure exchanges. In this mode, the UDF cannot be extended using the UDF Debug tool.

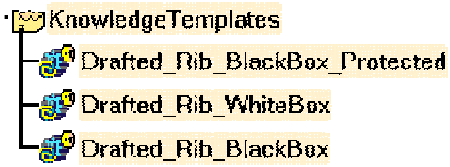
Student Notes:

How to debug a UDF (2/5)

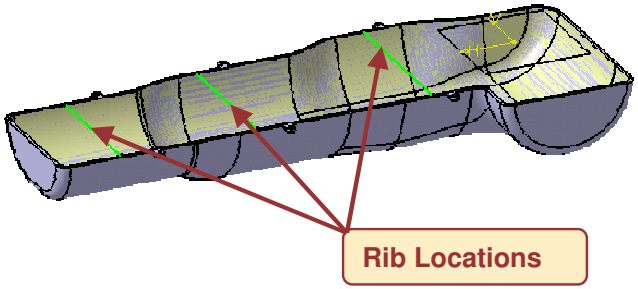
CATIA data used: Drill_Housing_For_RIB_UDF.CATPart

CATIA data used: Drafted_Rib_UDF_Source.CATPart

1. **Open the attached CATIA documents and save it to some location.**
2. **Open Drafted_Rib_UDF_Source.CATPart and note that there are three UDFs present in it.**



3. **Close this part and open 'Drill_Housing_For_RIB_UDF.CATPart'.**

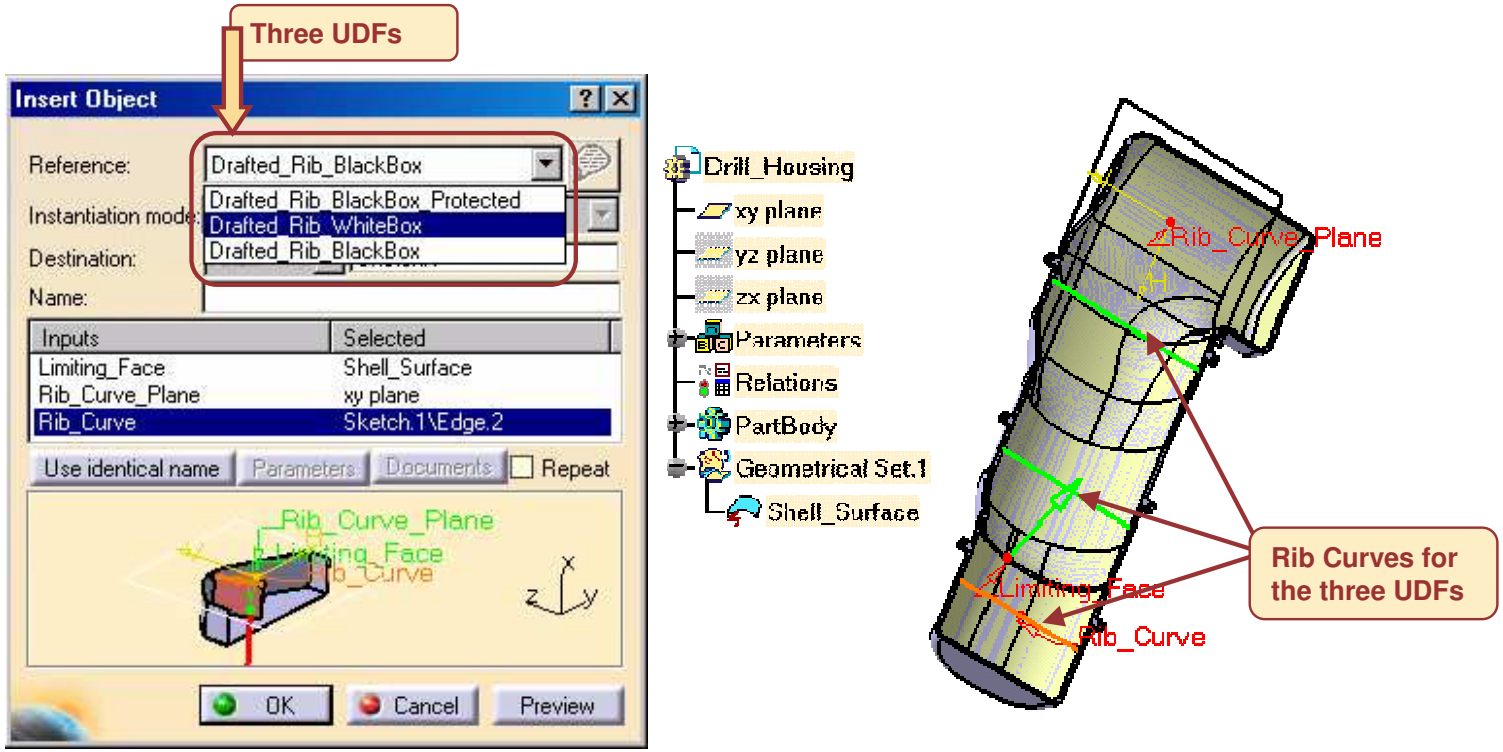


This part contains three locations where you will instantiate UDFs in the form of 'Ribs' using the three different UDF templates from the 'Drafted_Rib_UDF_Source.CATPart' part.

Student Notes:

How to debug a UDF (3/5)

4 **Instantiate the ribs using the three different UDFs from 'Drafted_Rib_UDF_Source.CATPart'. Select the inputs as shown below.**

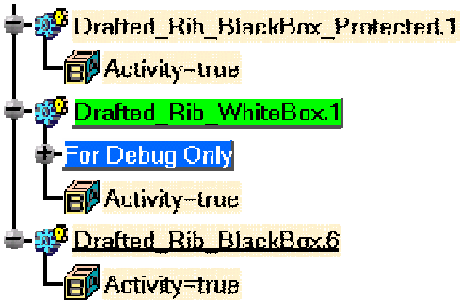


The inputs 'Limiting_Face' and 'Rib_Curve_Plane' are the same for all the three cases. The only variable input in the three UDF instantiations is the 'Rib_Curve', for which you have to select the three green lines in each individual case.

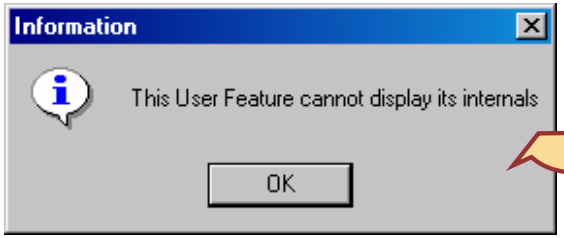
Student Notes:

How to debug a UDF (4/5)

5 Note the three different UDFs in the specification tree.



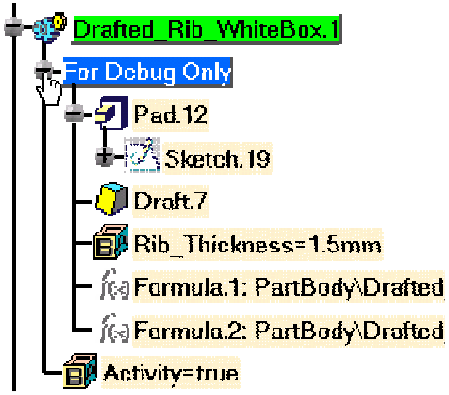
6 Click on 'Drafted_Rib_BlackBox_Protected.1' in the specification tree and click the 'Udf debug' tool.



You will receive this message because the internals of the UDF are protected.

7 Expand the 'Drafted_Rib_WhiteBox.1 and note the internals of the UDF.

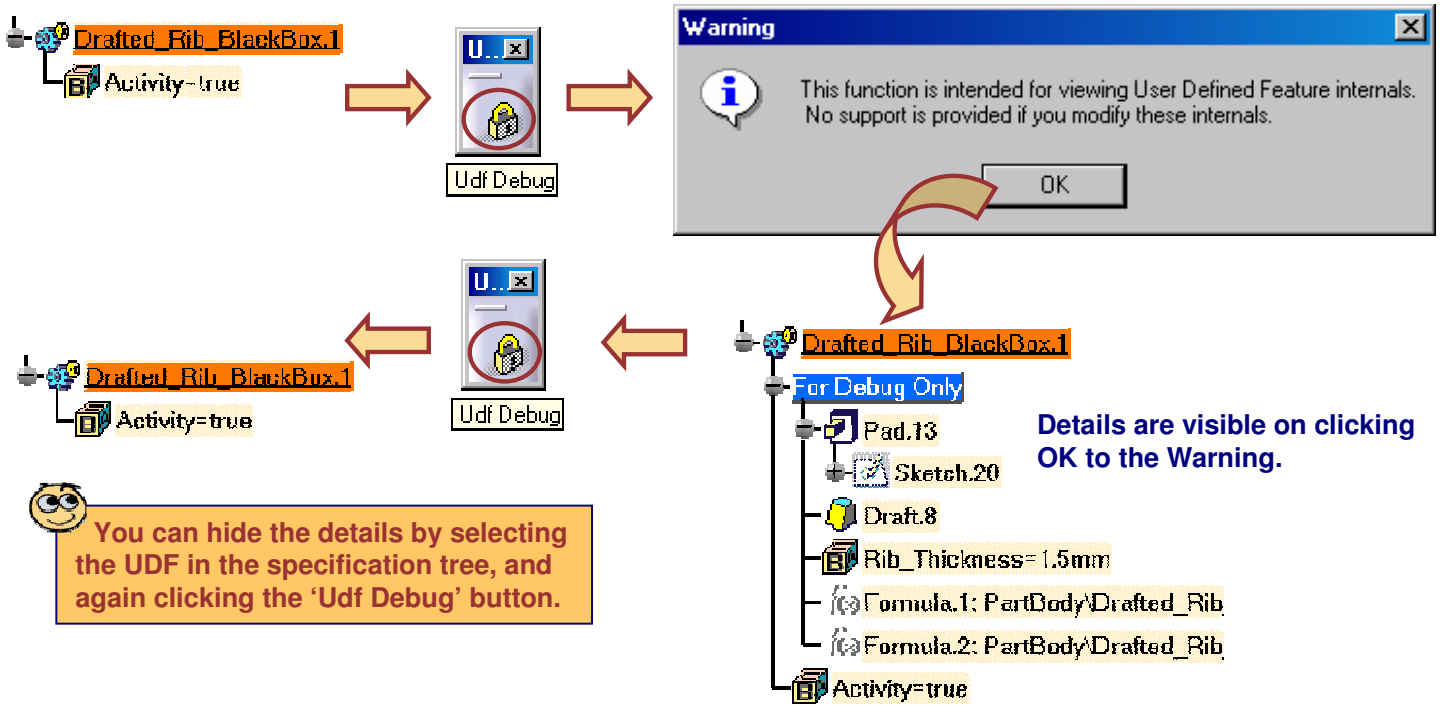
These internals are used for debugging purposes in case of any errors after instantiation.



Student Notes:

How to debug a UDF (5/5)

8 To view the details of the 'Drafted_Rib_BlackBox.1' UDF, select it from the specification tree and click the 'Udf Debug' tool.

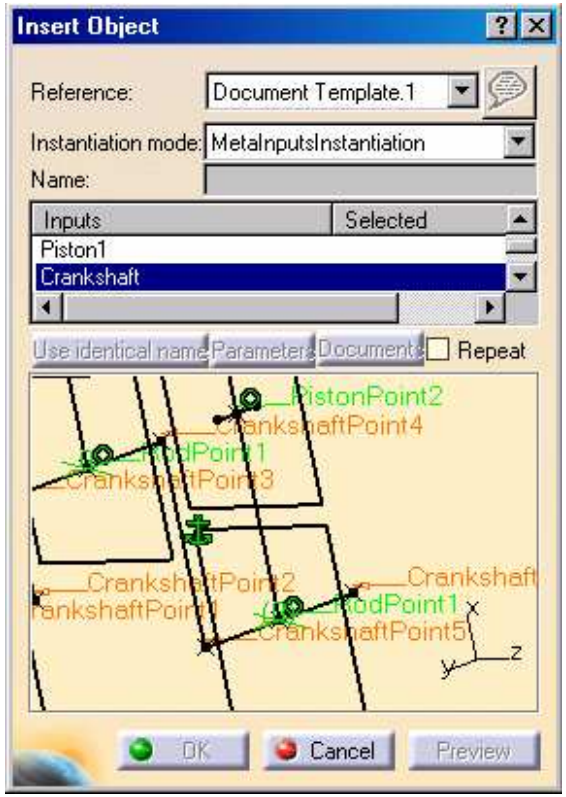


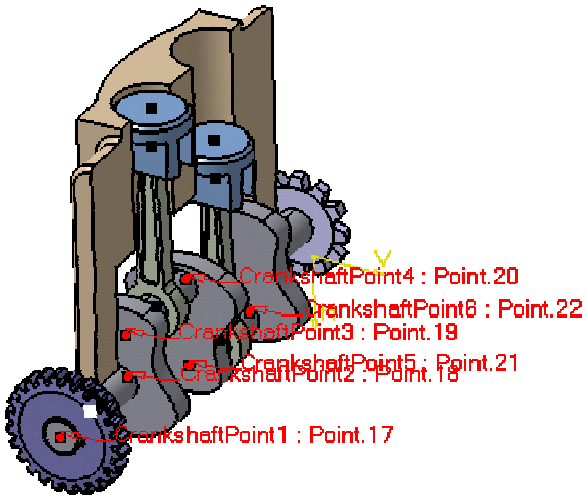
You can hide the details by selecting the UDF in the specification tree, and again clicking the 'Udf Debug' button.

Student Notes:

UDF – Meta Inputs

In this lesson, you will learn how to define and use the 'Meta Inputs' feature of UDF.

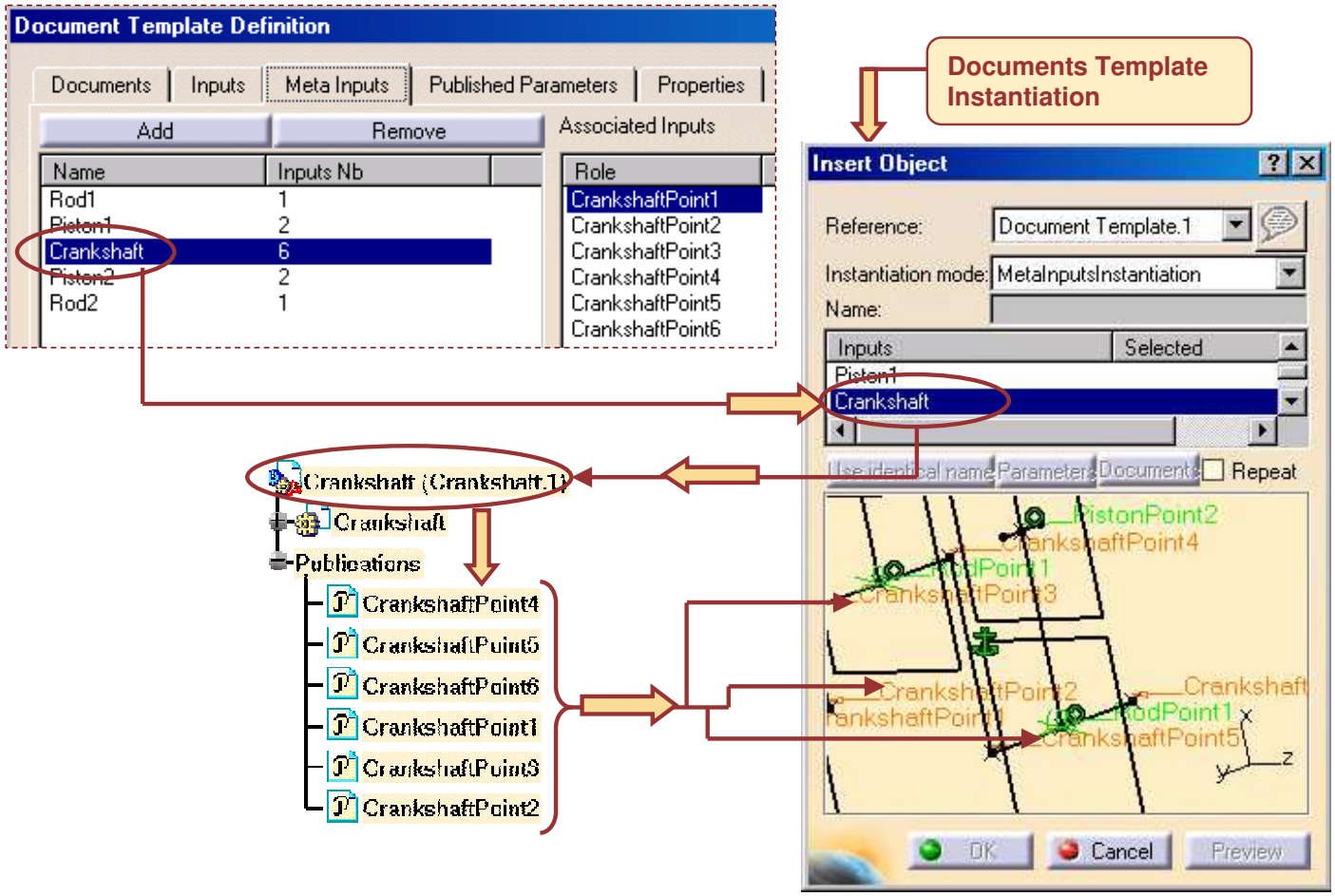




Student Notes:

What are 'Meta Inputs'

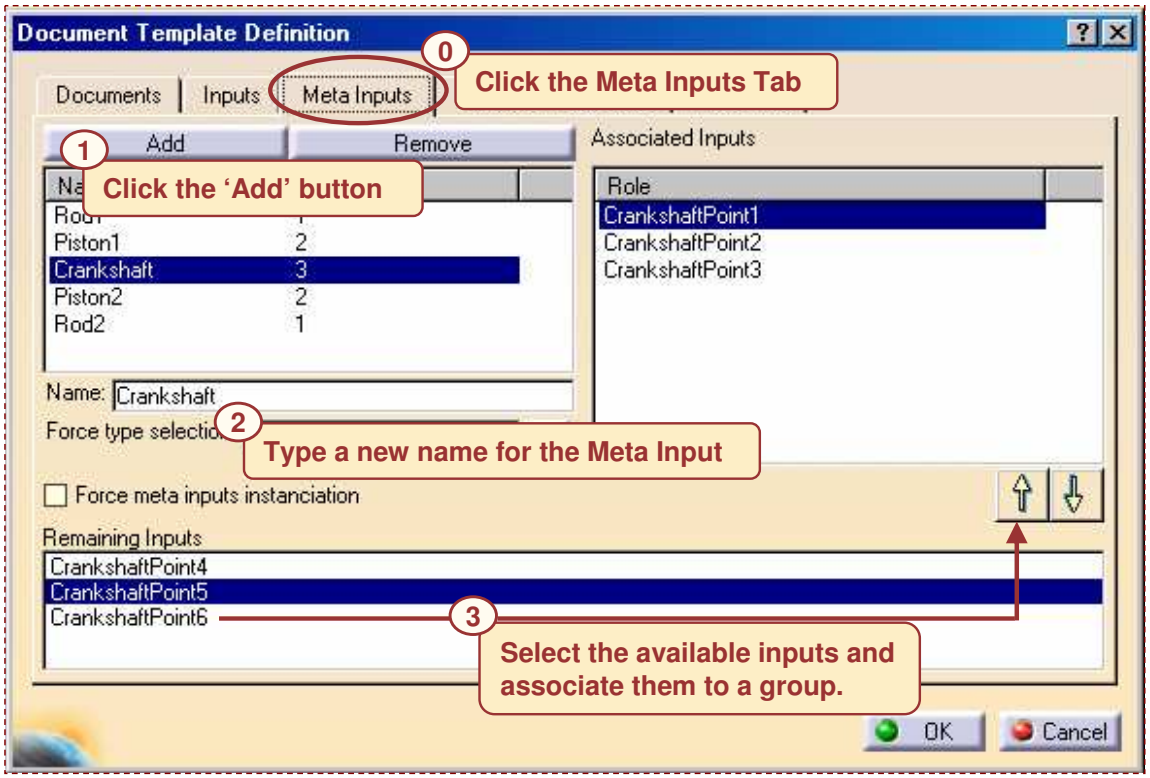
The Meta Inputs tab provides a facility to directly select a group of inputs simply by selecting a component in the specification tree during UDF instantiation.



Student Notes:

How to define 'Meta Inputs' for UDF

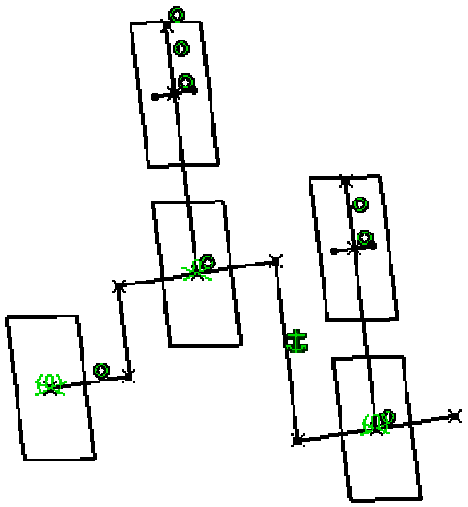
Defining the 'Meta Inputs' involves creating a 'Group of Inputs' and associating a number of individual inputs to this group.

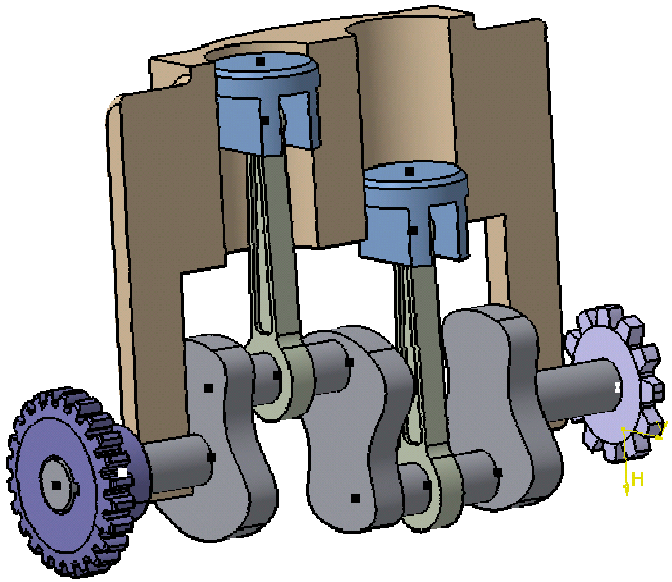


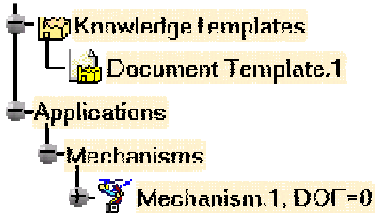
Student Notes:

Example of Meta Inputs

For example, the geometry shown below (image on the left) consists of a wireframe mechanism of a two-cylinder engine. A document template of this product can be created and instantiated in an assembly of a two-cylinder engine (image on the right).



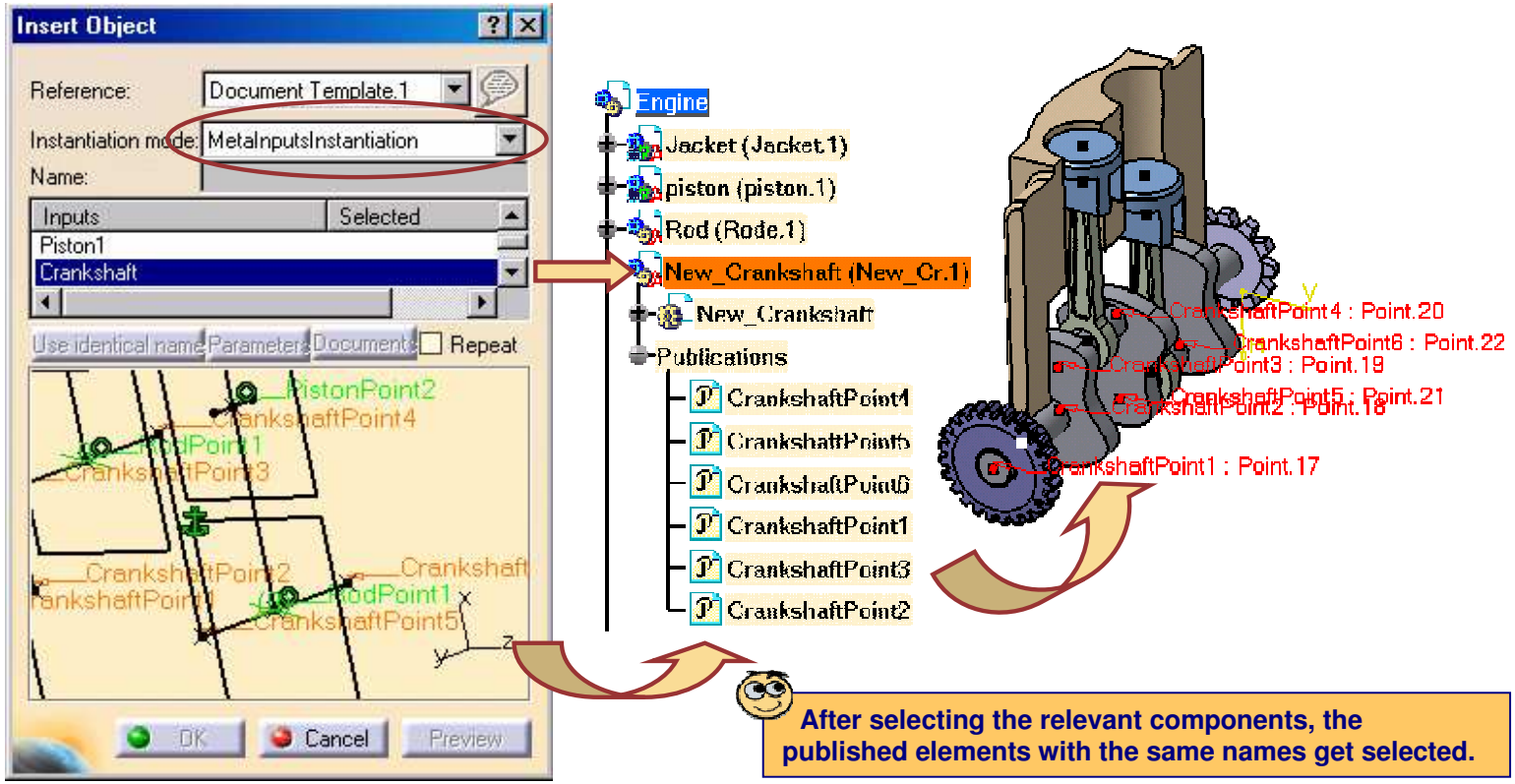




Student Notes:

How to select 'Meta Inputs' during UDF Instantiation

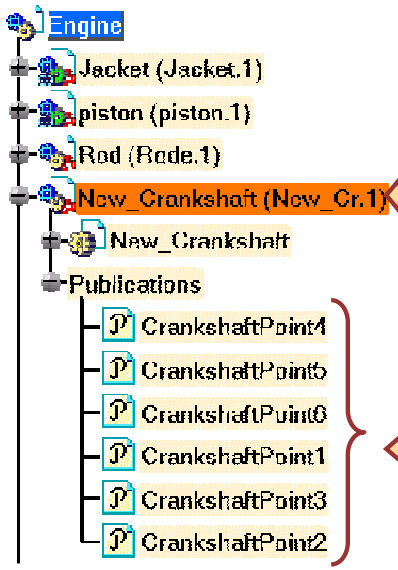
To be able to select the Meta Inputs during UDF instantiation, you have to select the 'Instantiation Mode' as 'MetaInputsInstantiation' and select the relevant components in the specification tree.



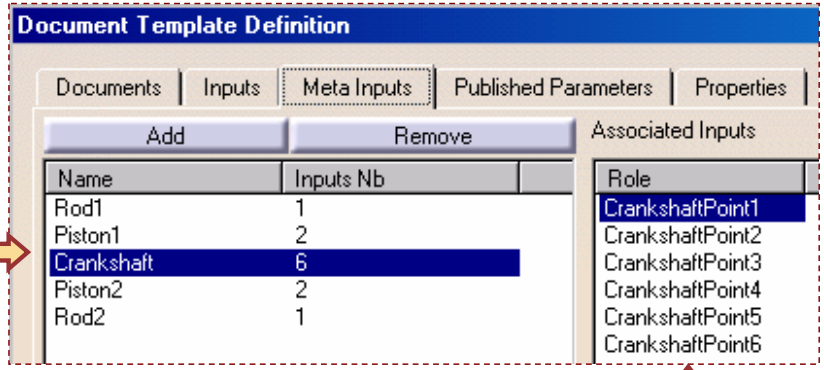
After selecting the relevant components, the published elements with the same names get selected.

Student Notes:

How to select 'Meta Inputs' during UDF Instantiation



Names need not be the same



Names should be the same

It is not necessary that the names of the 'Meta Inputs' should match with the names of the components in the specification tree. However, it is necessary that the names of the published elements in the components should match with the names of the 'Associated Inputs' of the 'Meta Inputs' in the definition dialog box.

Student Notes:

Creating and Using Part and Assembly Templates

You will become familiar with the use of advanced replication tools called Part and Assembly Templates.

- **Presentation of Document Templates**
- **Creating a Document Template**
- **Saving a Document Template**
- **Instantiating a Document Template**

Student Notes:

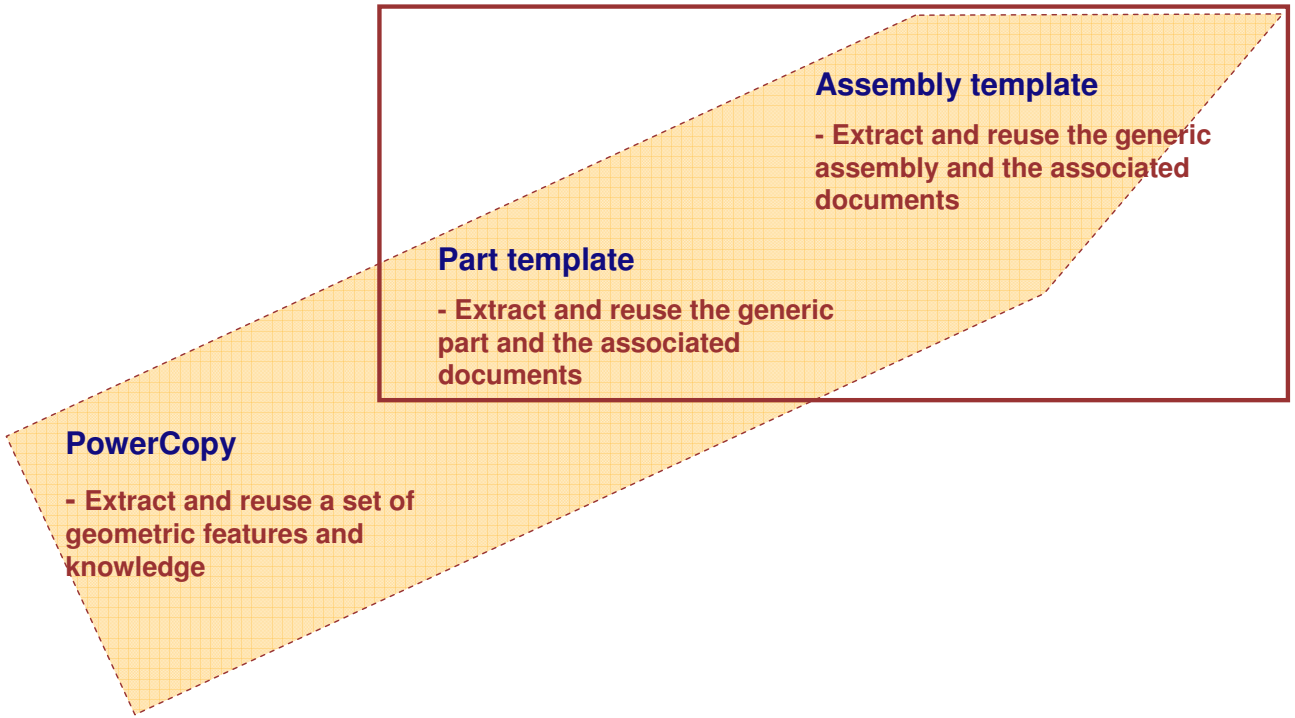
Presentation of Document Templates

You will learn about the benefits of Document Templates and their differences with respect to PowerCopies and User Defined Features.

Student Notes:

Document Templates vs PowerCopies (1/2)

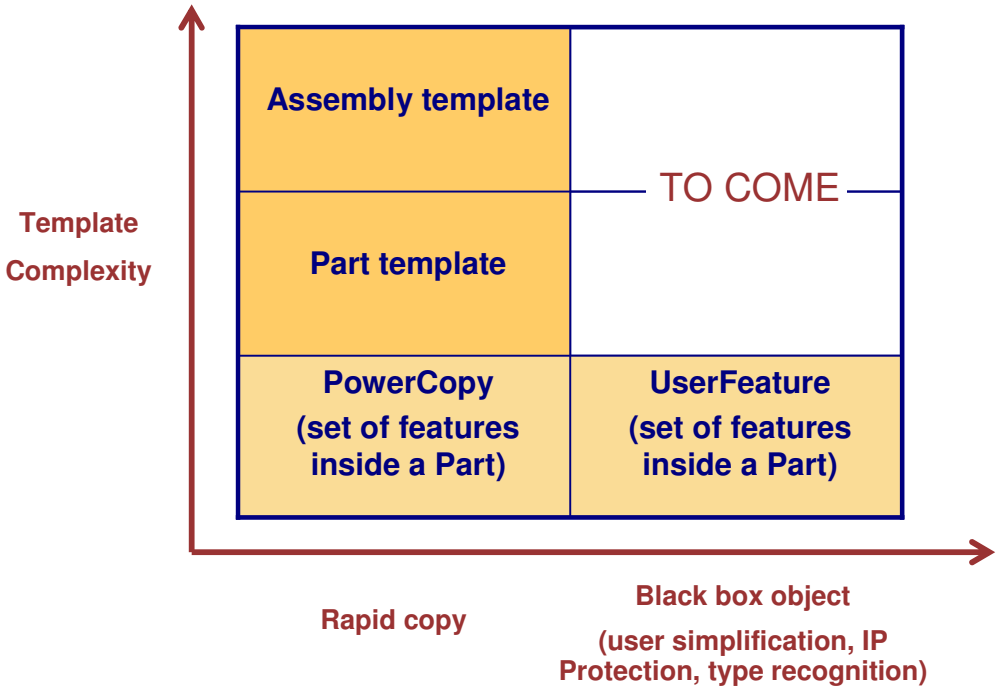
- **Part and Assembly Templates are an extension of the PowerCopy capability at the level of the assembly.**



Student Notes:

Document Templates vs PowerCopies (2/2)

- **Those templates are similar to PowerCopies and not to UserFeatures: they do not produce a single object when instantiated.**



Student Notes:

What is a Part Template?

- A Part created in CATIA may contain user parameters and geometry data. It is not a contextual part. The user can create a part template that references that part. This template is a feature that is created in the CATPart document itself (very similar to the PowerCopy definition) and stored in a catalog. Several part templates may be defined in the same CATPart document.

- To create a Part Template, the user:
  - Selects parameters and geometry data that will be considered as the template inputs (he can assign a role and a comment to each input).
  - Publishes some internal parameters (name and comment). The part number is automatically published.
  - Gives a name, comment, URL, and icon to this template.

- Once the template is created, the user stores it in a catalog and uses it in another context. In product structure context, the part is inserted as a component of the current product.

Student Notes:

What is an Assembly Template?

- A user creates an Assembly interactively. Then, he wants to create an Assembly Template that references the root product of this assembly.

- To create an assembly template, the user:
  - Selects parameters and geometry data that will be considered as the template inputs (he can assign a name to each input).
  - Publishes some internal parameters (name and comment).
  - Chooses if:
    - The part numbers of replicated components are automatically published
    - For each part or each sub-assembly this sub-component will be replicated at instantiation, or if only a reference to this sub-component will be created (a standard component)
    - He wants to select external documents (Drawings / Analysis) that references elements of the product structure. Those elements will be replicated at instantiation
  - Assigns a name, comment, URL and icon to this template.

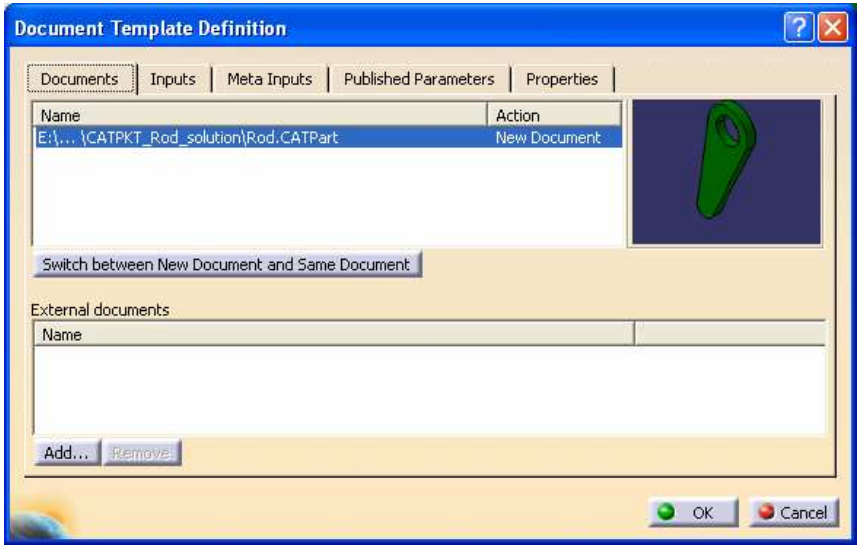
- Once this template is created, the user stores it in a catalog and uses it in another context.

- The template definition is a feature located in the CATProduct document itself. Several assembly templates may be defined in the same CATProduct document.

Student Notes:

Creating a Document Template

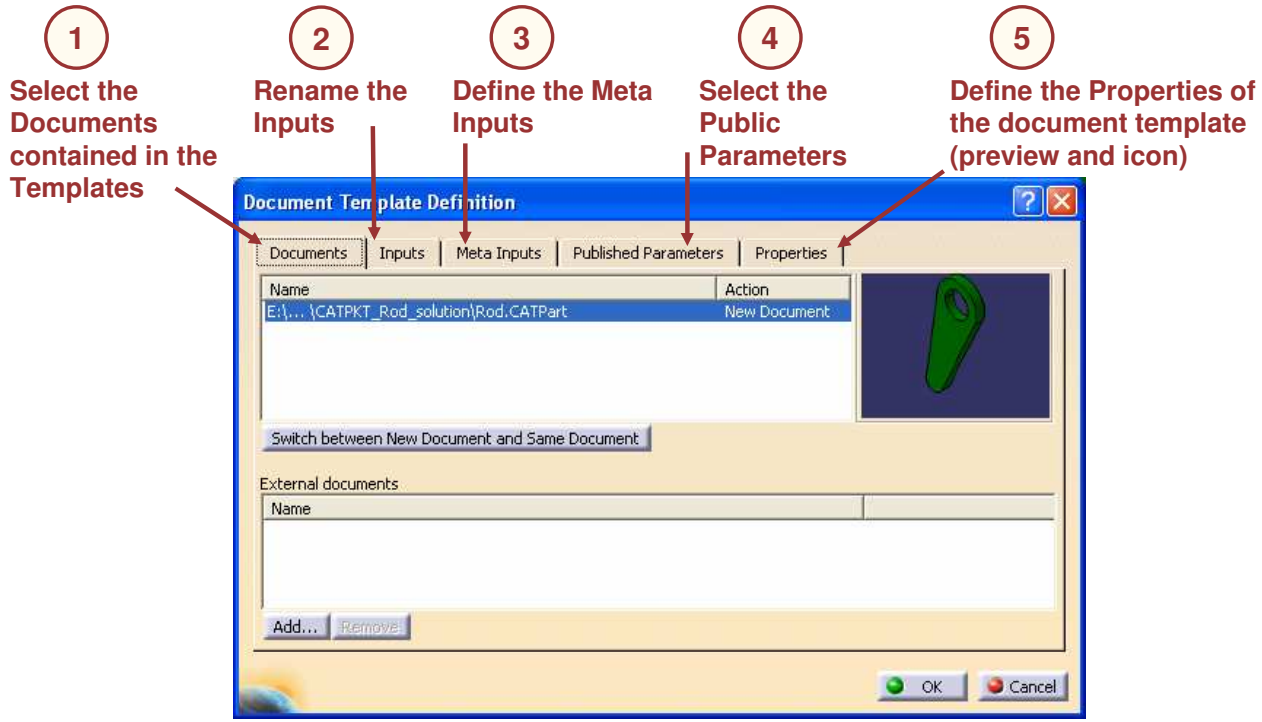
You will learn how to store documents in a Template Feature in order to reuse them later in another context.



Student Notes:

Document Templates Creation Process

- **The Document Template definition can be accessed in the Insert menu (Document Template Creation) of these workbenches:**
  - **Part Design**
  - **Generative Shape Design**
  - **Assembly Design**
  - **Product Structure**

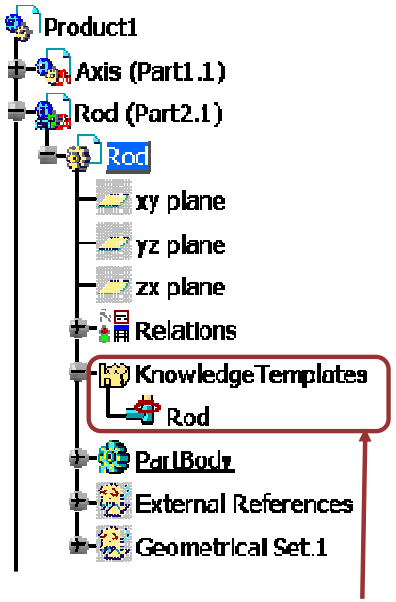


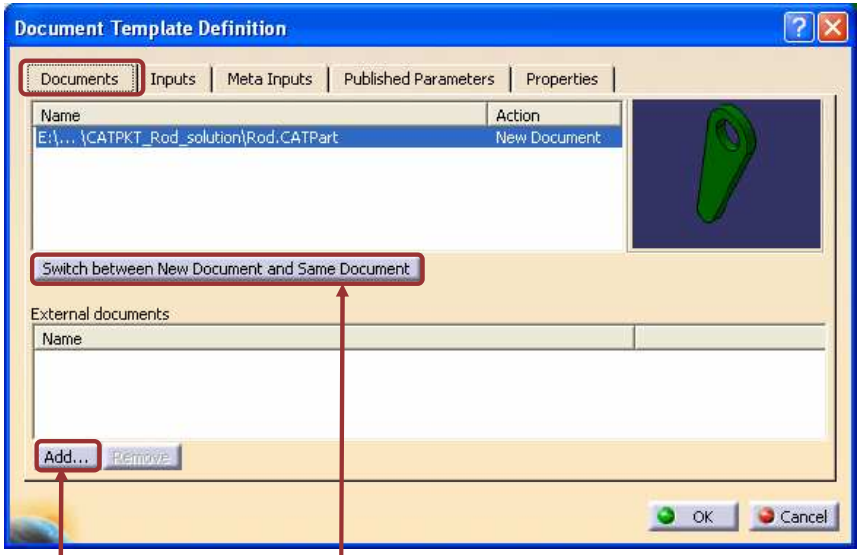
Student Notes:

Creating Document Templates (1/4)

The Documents tab shows the complete path and action of the files referenced in the template.

Be active at the level of the document you want to create when launching the Document Template creation.





Created Document Templates are stored under the Knowledge Templates node

Insert External documents to the Template (CATDrawing for instance)

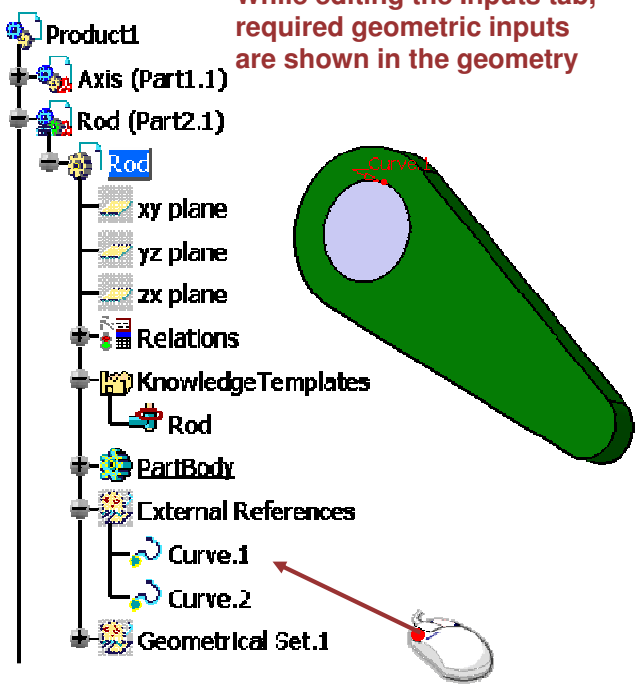
Switch between the Instanciated mode (the document has no links with the original component) and the Referenced mode (links maintained with the original file)

Student Notes:

Creating Document Templates (2/4)

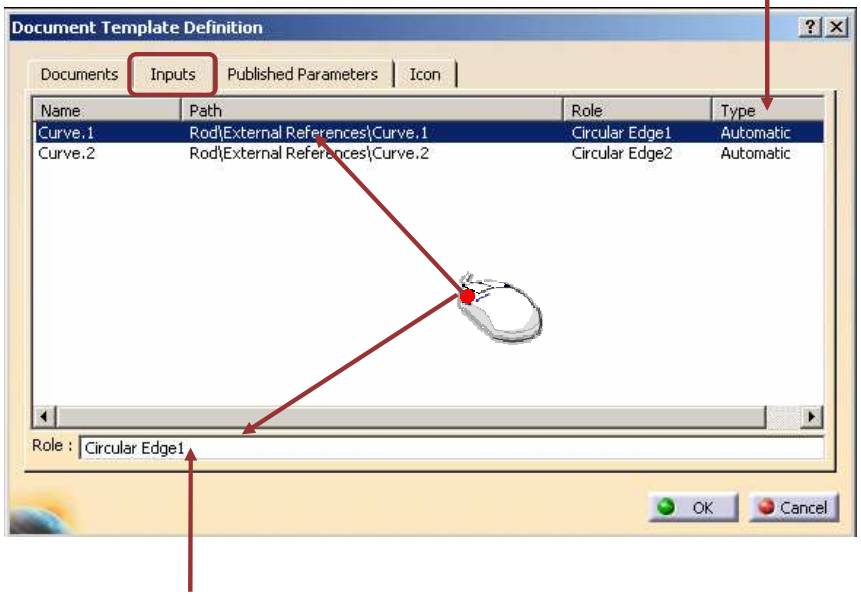
The Inputs tab enables you to define the reference elements (making up the template) by selecting them in the geometry or in the specification tree.

While editing the Inputs tab, required geometric inputs are shown in the geometry



For manual inputs, selection is done by clicking on the features in the tree

If you have selected a document designed in context, the Inputs will be automatically selected

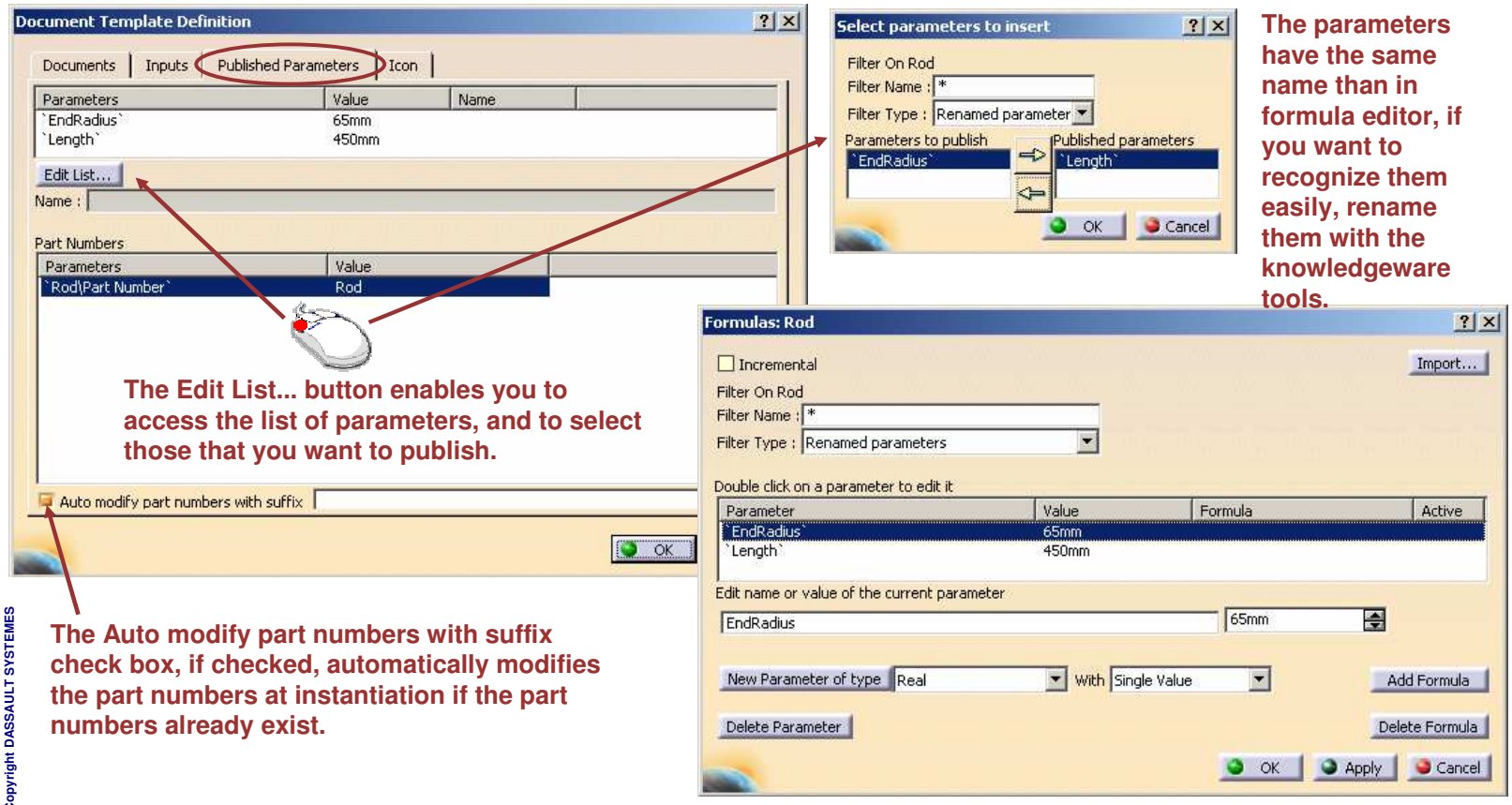


The Role field enables you to select one of the items displayed in the window and rename it

Student Notes:

Creating Document Templates (3/4)

The Published Parameters tab enables you to define which of the parameter values used in the template you will be able to modify when instantiating it.



The parameters have the same name than in formula editor, if you want to recognize them easily, rename them with the knowledgeware tools.

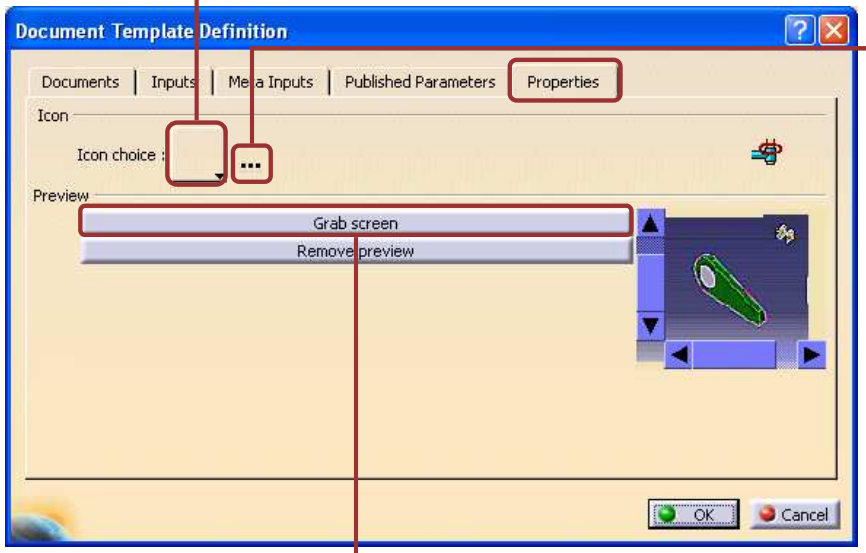
The Edit List... button enables you to access the list of parameters, and to select those that you want to publish.

The Auto modify part numbers with suffix check box, if checked, automatically modifies the part numbers at instantiation if the part numbers already exist.

Student Notes:

Creating Document Templates (4/4)

The Properties tab enables you to modify the icon identifying the template in the specifications tree. A subset of icons is available while clicking the Icon choice button.



You can consult the list of all available icons with the browser

The Grab screen button enables you to capture an image of the template to be stored along with its definition

The Grab screen makes a grab of CATIA Window to put it as the preview of the Document Template: you can prepare the CATIA window for the grab (remove dialog box, compass and tree, and make the correct zoom)

Preview will be useful while referencing a document template in a catalog.

Student Notes:

Saving a Document Template

You will learn how to store a Document Template in a catalog in order to share it with other users or to reuse it later in another context.

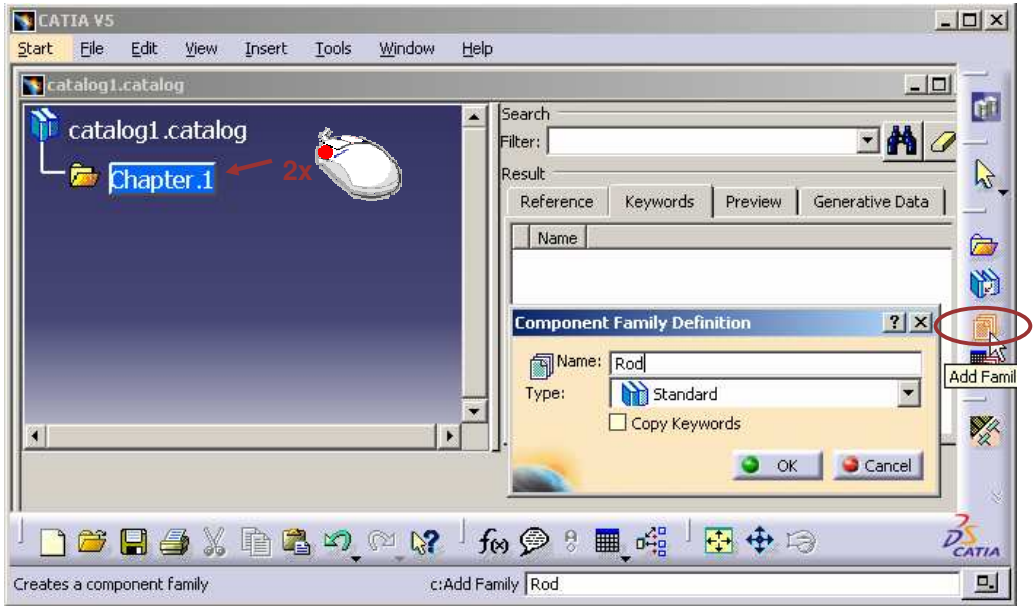
Student Notes:

Saving Document Templates in a Catalog (1/2)

Save the file containing the Document Template.

From the Start menu, select the Infrastructure->Catalog Editor command. The Catalog Editor opens.

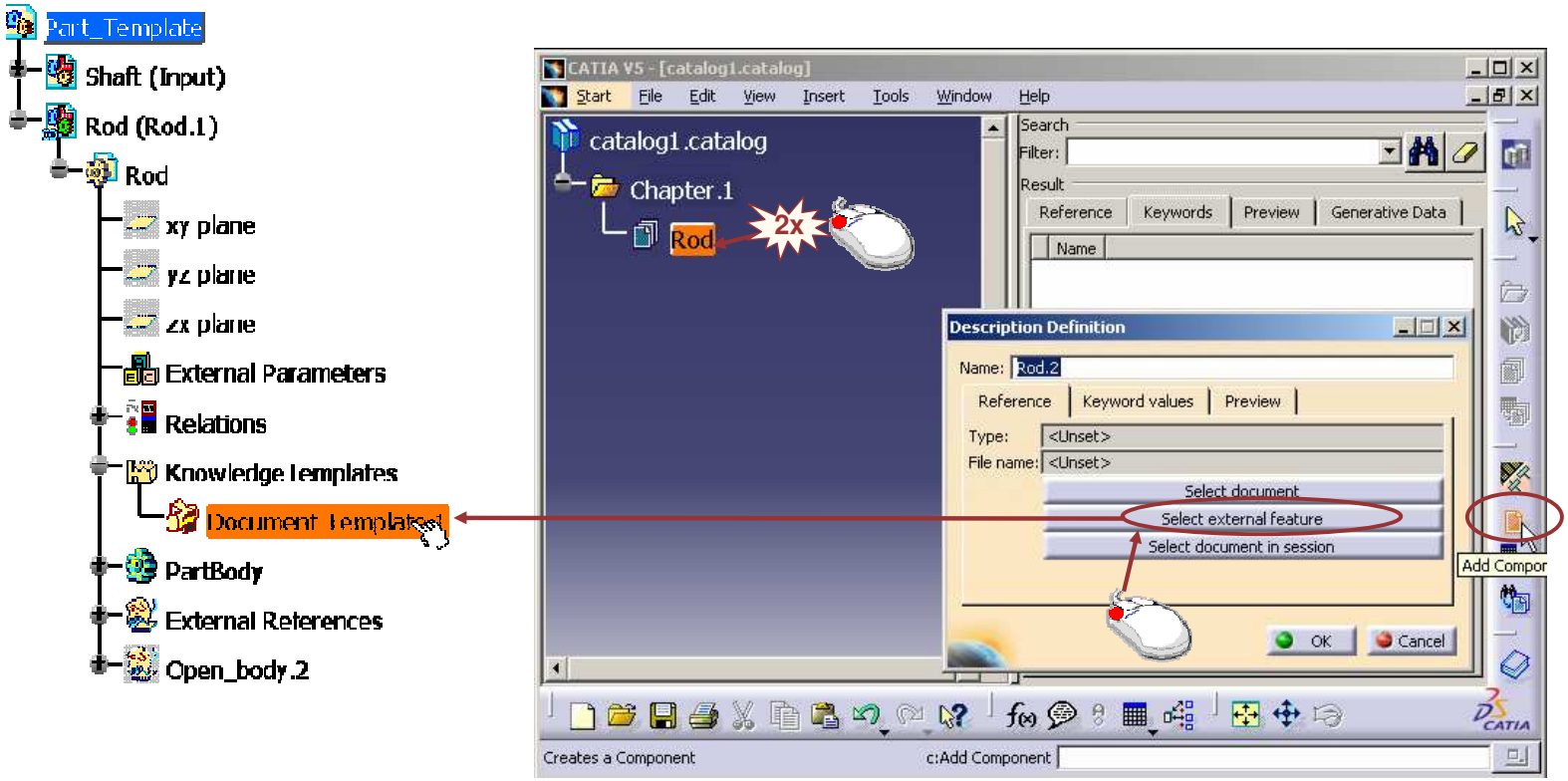
Double-click Chapter.1 and click the Add Family icon to create a family. Indicate the name of the family in the Name field, Rod in this scenario, and click OK. The Rod family is added below Chapter.1 in the tree.



Student Notes:

Saving Document Templates in a Catalog (2/2)

Double-click Rod in the tree and click the Add Component icon. The Description Definition dialog box appears. Click the Select external feature button and click the Document Template in the file to select it. The template is added to the Description Definition window. Click OK.



Save the catalog and close it

Student Notes:

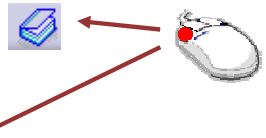
Instantiating a Document Template

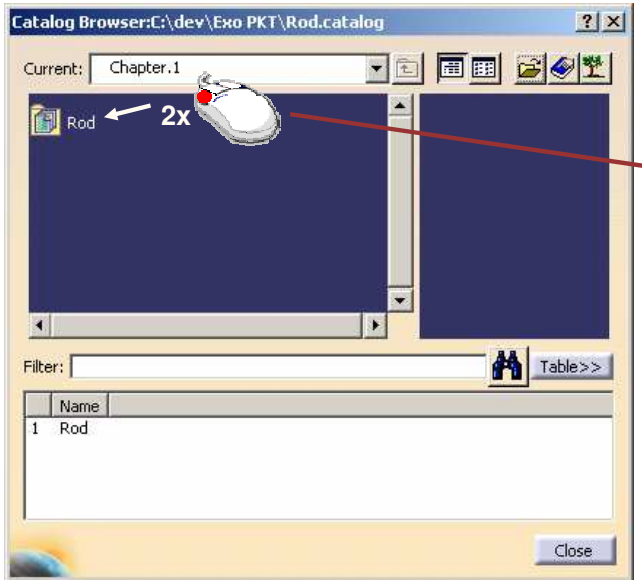
You will learn how to import a Document Template in a new context and how to adapt it to this context.

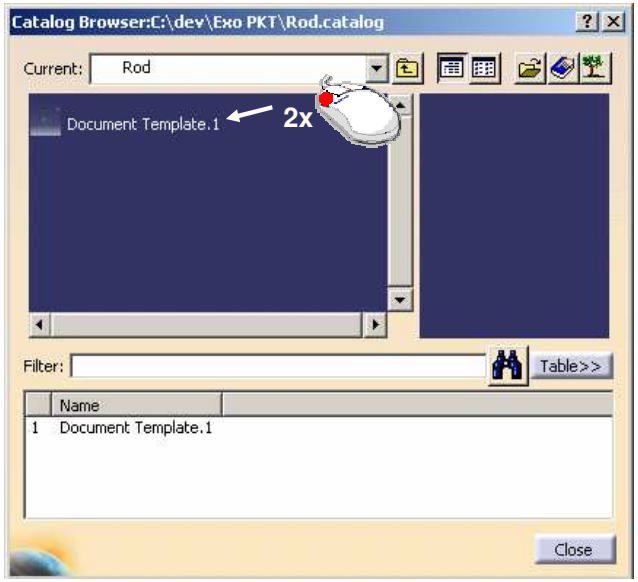
Student Notes:

Instantiating Document Templates from a Catalog (1/4)

Click the Catalog icon and select the catalog you have created.





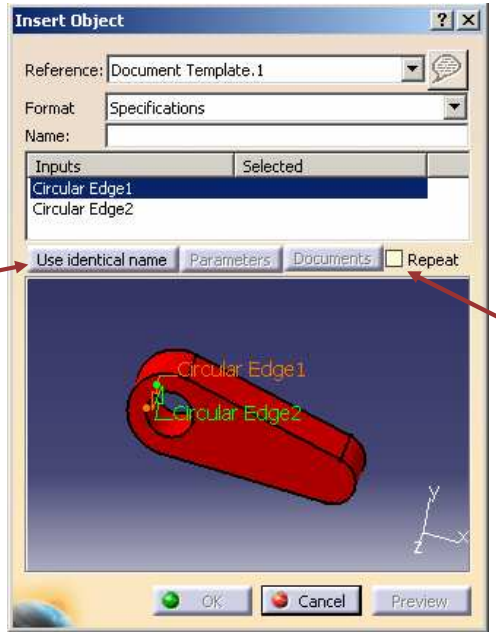


Double-click the Keypad family and the Document Template.1 template

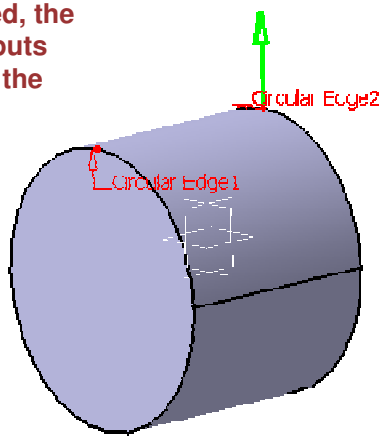
Student Notes:

Instantiating Document Templates from a Catalog (2/4)

Select in the geometry the inputs of the Document Template.



When selected, the geometric inputs are shown in the geometry



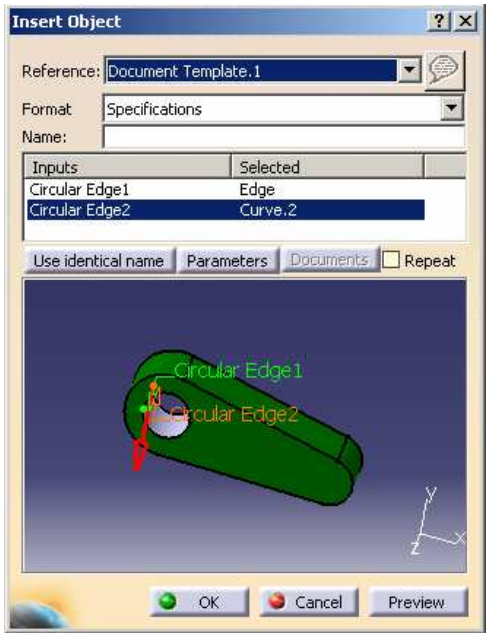
Using identical names allows the automatic selection of the geometric inputs that have the same name as those used for the creation of the PowerCopy

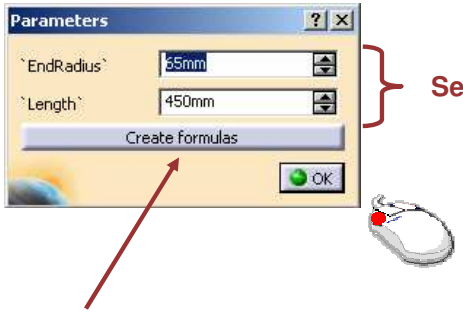
If you want to insert the document template several times, check the repeat option

Student Notes:

Instantiating Document Templates from a Catalog (3/4)

Click on Parameters to set values of parameters that have been published at the creation of the Document Template.





Set values

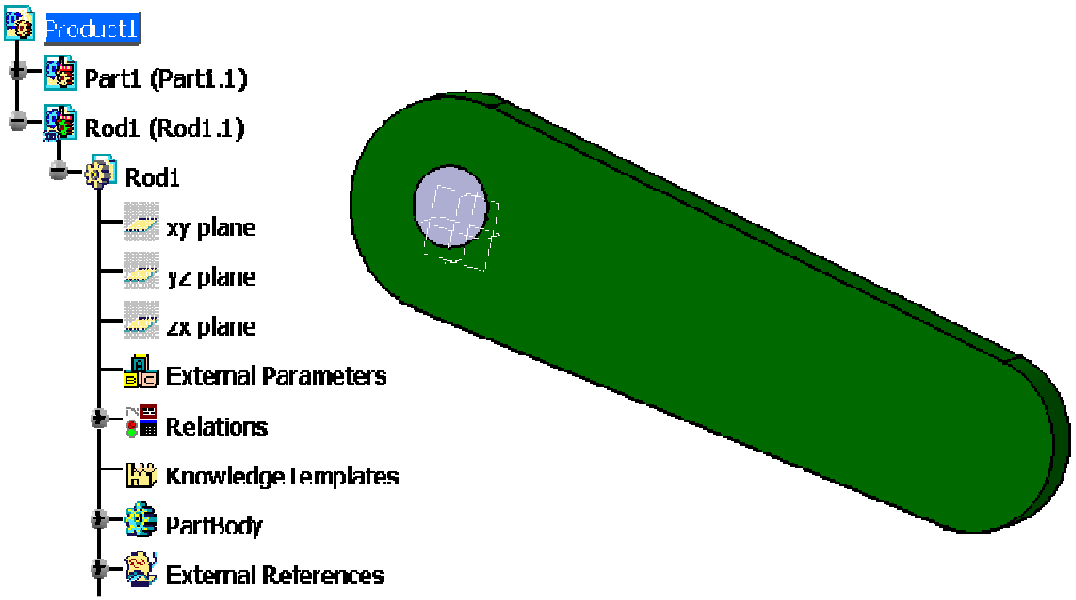
Click OK

If you want, you can create on the fly formulas with parameters having the same names

Student Notes:

Instantiating Document Templates from a Catalog (4/4)

Click OK to the 'Insert Object' dialog box to instantiate the template.



Student Notes:

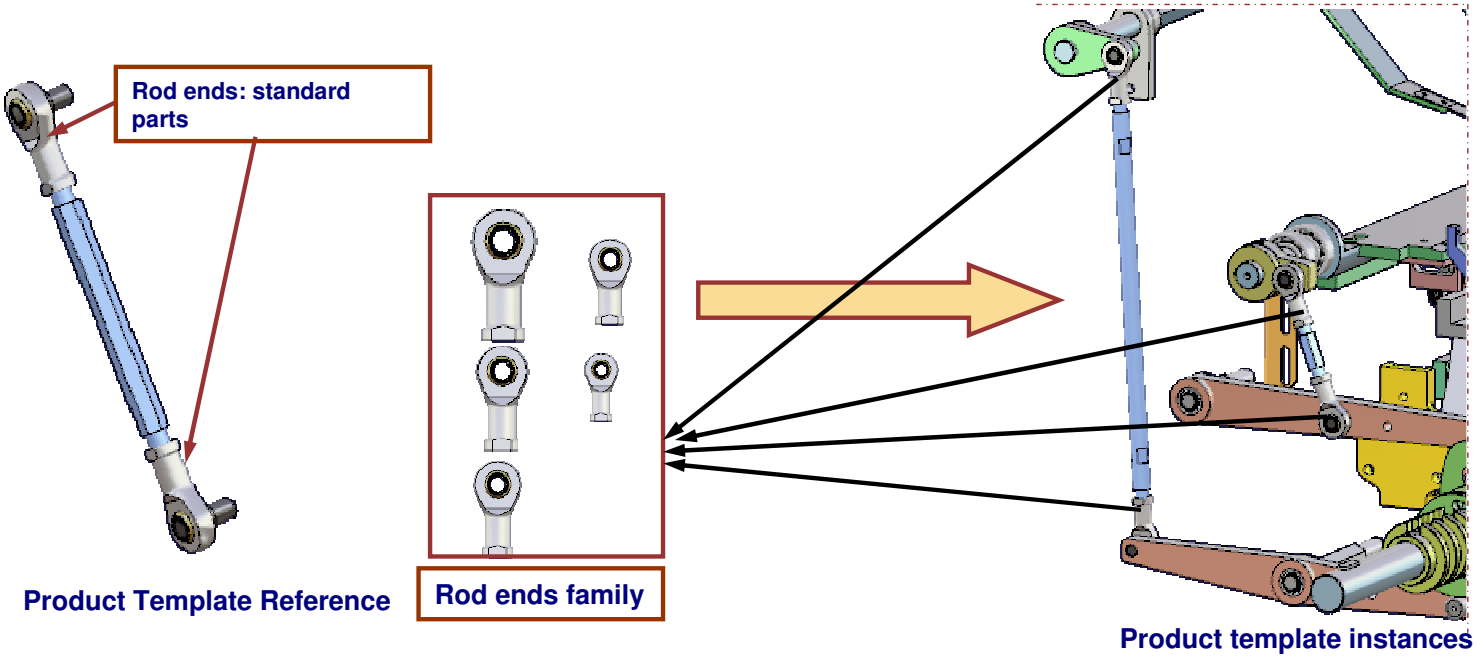
Managing Standard Components

You will learn how to deal with standard components in Document Templates.

Student Notes:

Introduction

- **An assembly template is usually made of a mix of specific components and standard components. In the example below, the female rod ends are standard parts, whereas the conecting bar and the pins are specific for each connecting rod:**



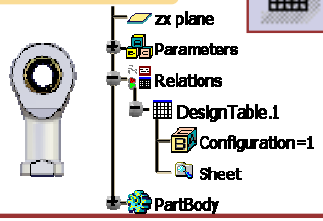
- **Expected behavior after template instantiation: no new documents are generated for the rod ends. The Rod ends family documents are used inside the template instances.**
- **CATIA V5 R18 knowledge language enhancements allow to reach this behavior.**

Student Notes:

Methodology Overview

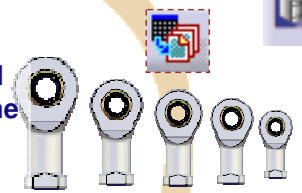
Create generic standard parts

For each standard part create a generic CATPart document containing a Design Table



Create a standard parts catalog

Create a standard parts family catalog (resolved components) reusing the previous generic CATParts



Create an ARM catalog

Create an ARM catalog referencing the standard parts catalog or directly standard components

Create Reactions inside the product

Use ManageInstance function inside Reactions to choose the right standard component to include.

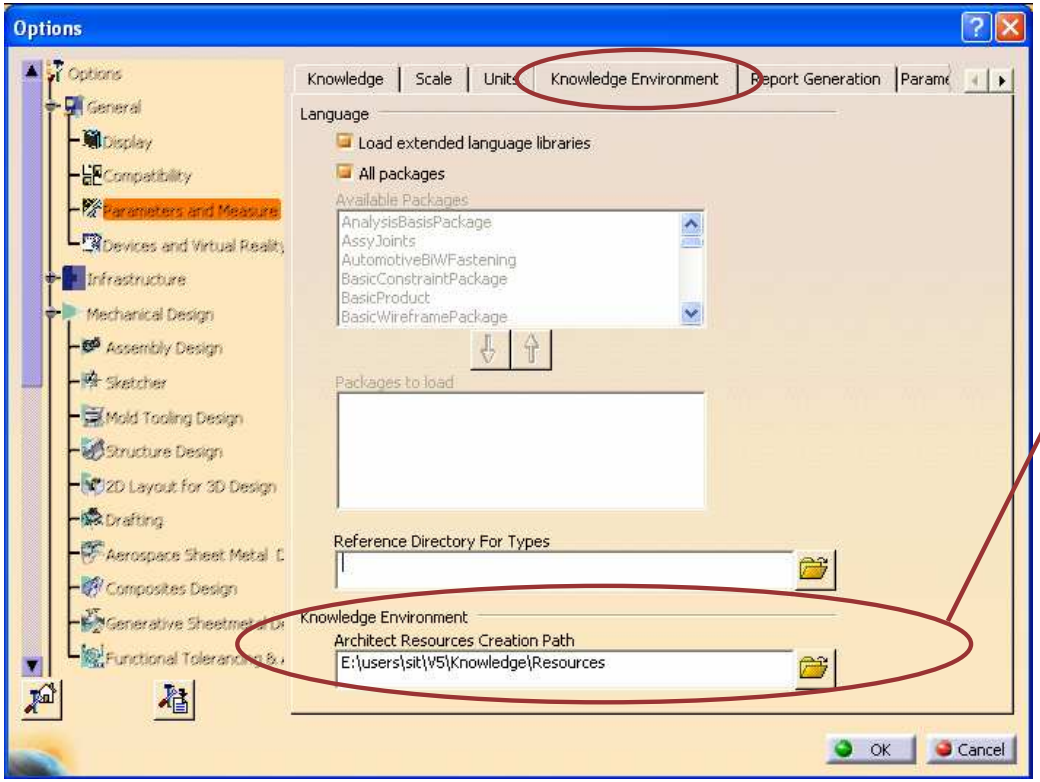
Create the document template

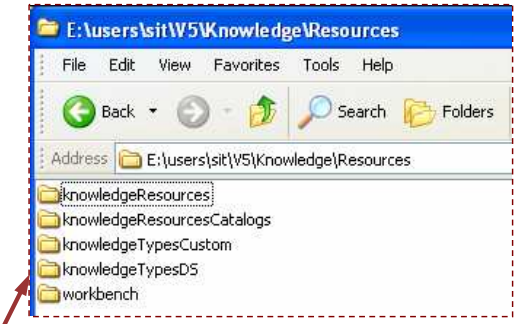
Create the document template, add other documents (drawings, FE Analysis), and save it in a catalog

Student Notes:

Knowledge Environment Settings

- **Fill the "Architect Resources Creation Path", it corresponds to the folder that contains other subfolders, among them: "knowledgeResourcesCatalogs" which contains the ARM catalogs (see next screen)**





Student Notes:

About ARM Catalogs

- Application Resource Management (ARM) .catalog files establish a link between the logical name of a resource and the physical resource referenced through the catalog. The objective of the catalogs is to answer this simple question: "give me the object named XXX".
- ARM uses the “logical” referencing mechanism: the resource is referenced from the application by using the “logical name” instead of using its full path.
- The logical name is then used as the keyword in the ARM catalog.
- The catalog is the standard CATIA catalog created in the Catalog Editor. It must be based on a fixed structure containing the following keywords:
  - Name: corresponds to the name of the resource i.e the one created by default by the catalog application
  - Logical Name: corresponds to the logical name of the resource. It represents the resource identifier. The value must be unique since it will be used by ARM to find the corresponding resource in the catalog
  - Type: corresponds to the type of resource that you want to reach
  - Usage: corresponds to a comment indicating what this resource is used for

Type and Usage keyword values can be kept unset

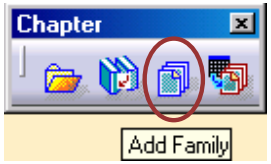
Student Notes:

Creating an ARM Catalog (1/3)

Create the Keywords

1. Create a new CatalogDocument (File>New).

2. Click the Add Family icon to add a family to your catalog. Click OK in the Component Family Definition dialog box. Double-click the family in the tree.



3. Click the Add Keyword icon or select Insert > Add Keyword... from the main menu to display the Keyword Definition dialog box.

4. Specify a name for the new keyword, Logical Name

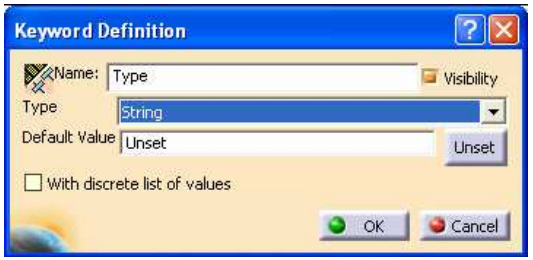
5. Use the drop-down list to select the keyword Type, String. This list provides all knowledge types, i.e. Integer, String, Boolean, Angle, and so on.

Student Notes:

Creating an ARM Catalog (2/3)

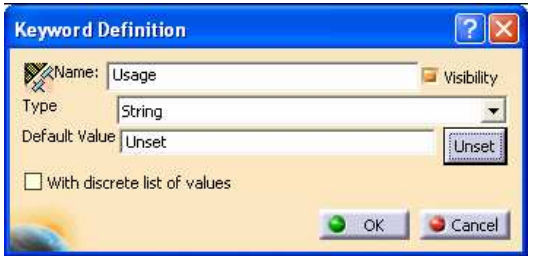
Create the Keywords

6. Repeat steps 3 & 4 to create the new keyword Type

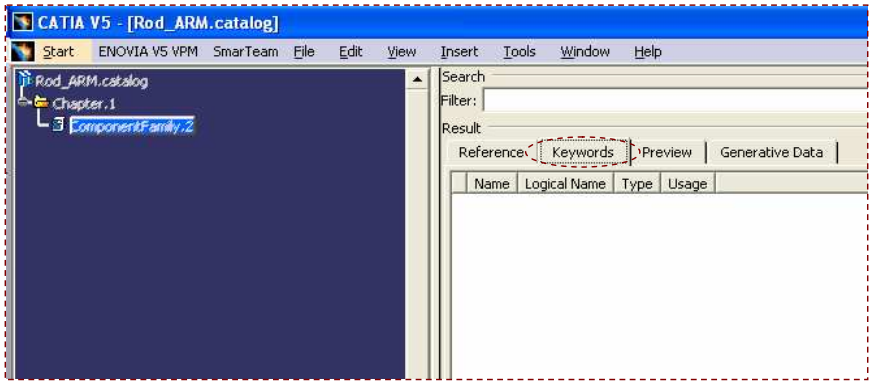


7. Use the drop-down list to select the keyword Type, String

8. Repeat steps 3 & 4 to create the new keyword, Usage



9. Use the drop-down list to select the keyword Type, String



Student Notes:

Creating an ARM Catalog (3/3)

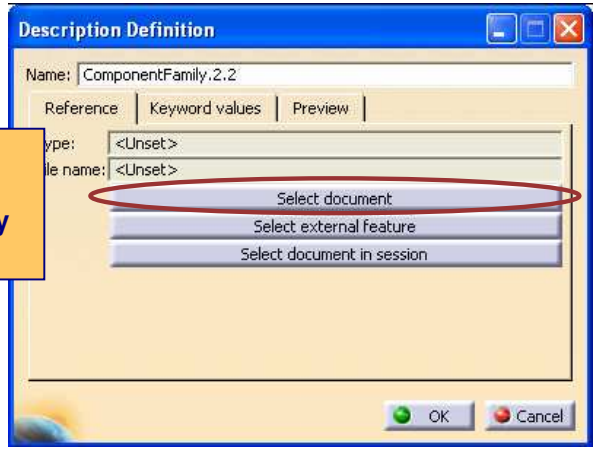
Add a standard parts family catalog

1. In the Catalog Editor, click the Add Component icon ( ). The Description Definition dialog box opens.

2. Click the Select document button.

3. Change the Files of type field to All Files(*.*) and select the .catalog file containing the standard parts family. Ignore the error message which is displayed.

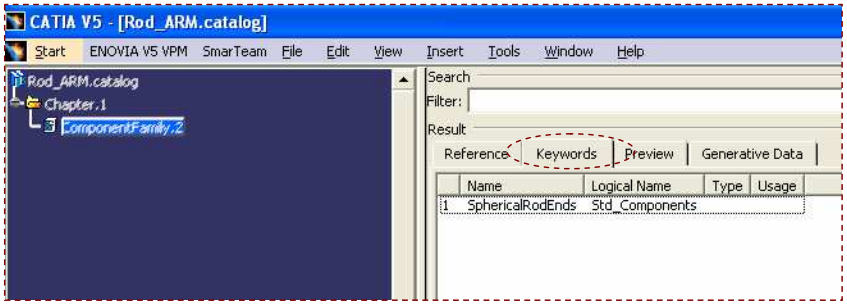
Refer to the Knowledge Advisor lesson to learn how to create a part family catalog.



4. Click the Keyword values tab. Click the Logical Name line and enter the desired logical name, click OK when done.

5. You can keep the default value Unset for Type and Usage keywords.

6. Save the catalog in the \xxx\knowledgeResourcesCatalogs directory to be taken into account by the Application Resources Management system.



Student Notes:

Choosing the Right Standard Component (1/5)

- The Knowledgeware language provides the ManageInstance and the RemoveInstance functions to address three major cases:
  - Case 1: need to switch between different standard components
  - Case 2: need to switch on/off a standard component
  - Case 3: need to search for the right element inside a standard parts family

- Syntax
  - Product->ManageInstance(arm : String, chapterName : String, query : String, instanceName : String): Product
    - ARM: Application Resource Management string. It is composed of two parts separated by "|": <catalogName>|<value of the keyword "Logical Name" of the catalog description>. The catalog description has to reference either a CATPArt document or a CATProduct document (case 1& 2) or a catalog document (case 3).
    - Chaptername: This argument is only used in case 2. In this case the ARM resource is a catalog document, and if the chapter name is specified, the system looks for the catalog chapter of this name.
    - Query: This argument is only used in case 2. in this case, the query is used to retrieve the part family elements that fit this query, either in a specific chapter i.e. the chapterName argument is filled, or in the whole catalog.
    - InstanceName: The ManageInstance method either creates or replaces a product instance. This argument is used to retrieve the existing instance, if any. It is also used to rename the created instance.
  - Product->RemoveInstance(instanceName : String

Student Notes:

Choosing the Right Standard Component (2/5)

Case 1 example: Switching standard motors inside a conveyor

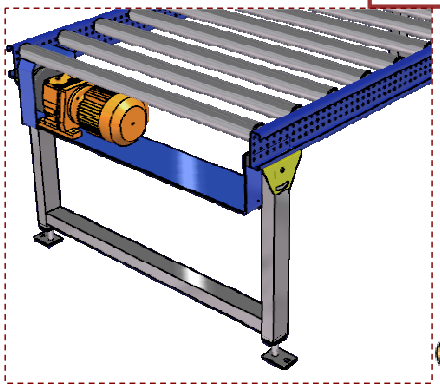
```
Let Prod (Product)
if (Motor=="0.5HP")
   Prod = Conveyor>ManageInstance("ARM|Motor0.5HP","","","Motor")
if (Motor=="1HP")
   Prod = Conveyor->ManageInstance("ARM|Motor1HP","","","Motor")
```

In this case, the second and the third arguments are useless

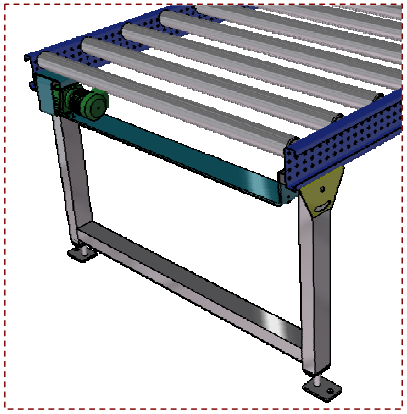
Assembly that contains the standard component

Logical name for component corresponding to a logical resource in an ARM (Application Resources Management) file referencing the standard components

Name of the instance is used to determine whether we are in the insertion or the replace mode



Automatic motor switch



Use Publications inside standard components to keep the assembly constraints, and the external references connected after the replace operation.

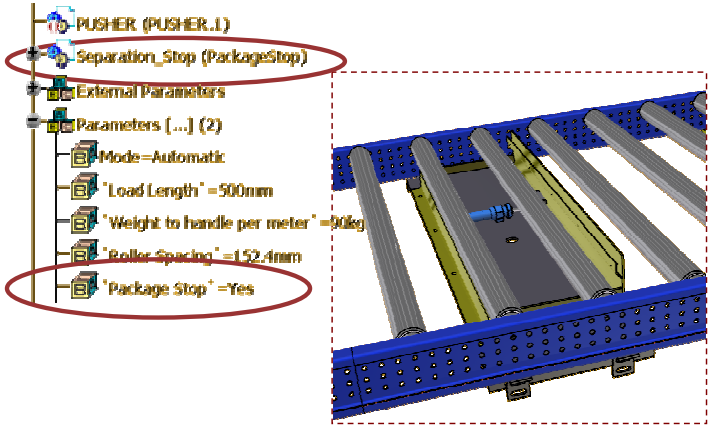
Student Notes:

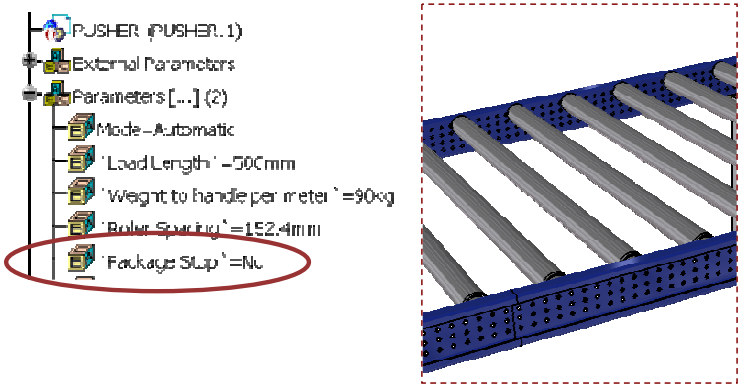
Choosing the Right Standard Component (3/5)

Case 2 example: Switching on/off a package stop in a conveyor

```
let optionproduct(Product)
if`Package Stop`  =="Yes"
        {set optionproduct= Master\Conveyor.1 ->ManageInstance("ConveyorR18|Stop","", "","PackageStop")
        Constraints\Coincidence.77\Coincidence.77\Activity =true
        Constraints\Coincidence.78\Coincidence.78\Activity =true
        Constraints\Coincidence.79\Coincidence.79\Activity =true}
if `Package Stop`  =="No"
        {Constraints\Coincidence.77\Coincidence.77\Activity =false
        Constraints\Coincidence.78\Coincidence.78\Activity =false
        Constraints\Coincidence.79\Coincidence.79\Activity =false
        Master\Conveyor.1  ->RemoveInstance("PackageStop“)}
```

Do not forget to activate/inactivate the assembly constraints linked to the standard component.





Package stop option: Yes/No

Student Notes:

Choosing the Right Standard Component (4/5)

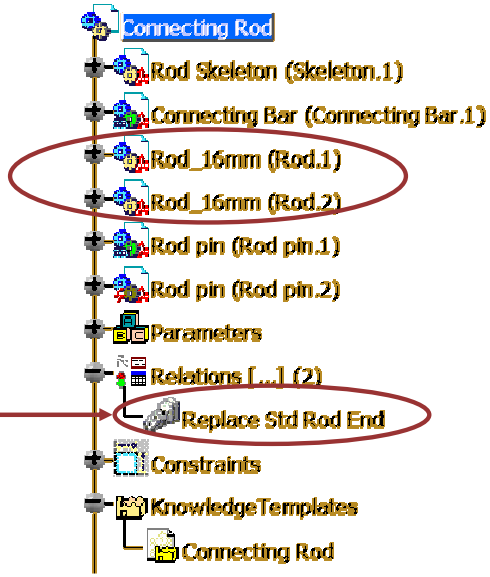
Case 3 example: Choosing the right standard rod ends inside a connecting rod assembly

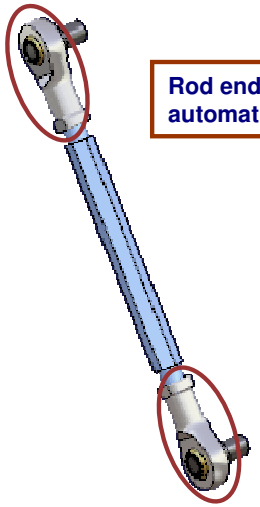
```
let newinstance (Product)
let instancename1 (String)
let instancename2 (String)
let query(String)
instancename1="Rod.1"
instancename2="Rod.2"
query="x.D=="+ToString(`Rod Skeleton\End Rod Diameter`  )
newinstance= `Connecting Rod`  ->ManageInstance("Rod_ARM|Std_Components","SphericalEndRods",query,instancename1)
newinstance=`Connecting Rod`  ->ManageInstance("Rod_ARM|Std_Components","SphericalEndRods",query,instancename2)
```

Query to perform in the catalog: search for rod ends with a diameter (D) equal to End Rod Diameter parameter value

Logical name for the catalog corresponding to the logical resource in an ARM (Application Resources Management) file, referencing a catalog

Name of the chapter in the catalog





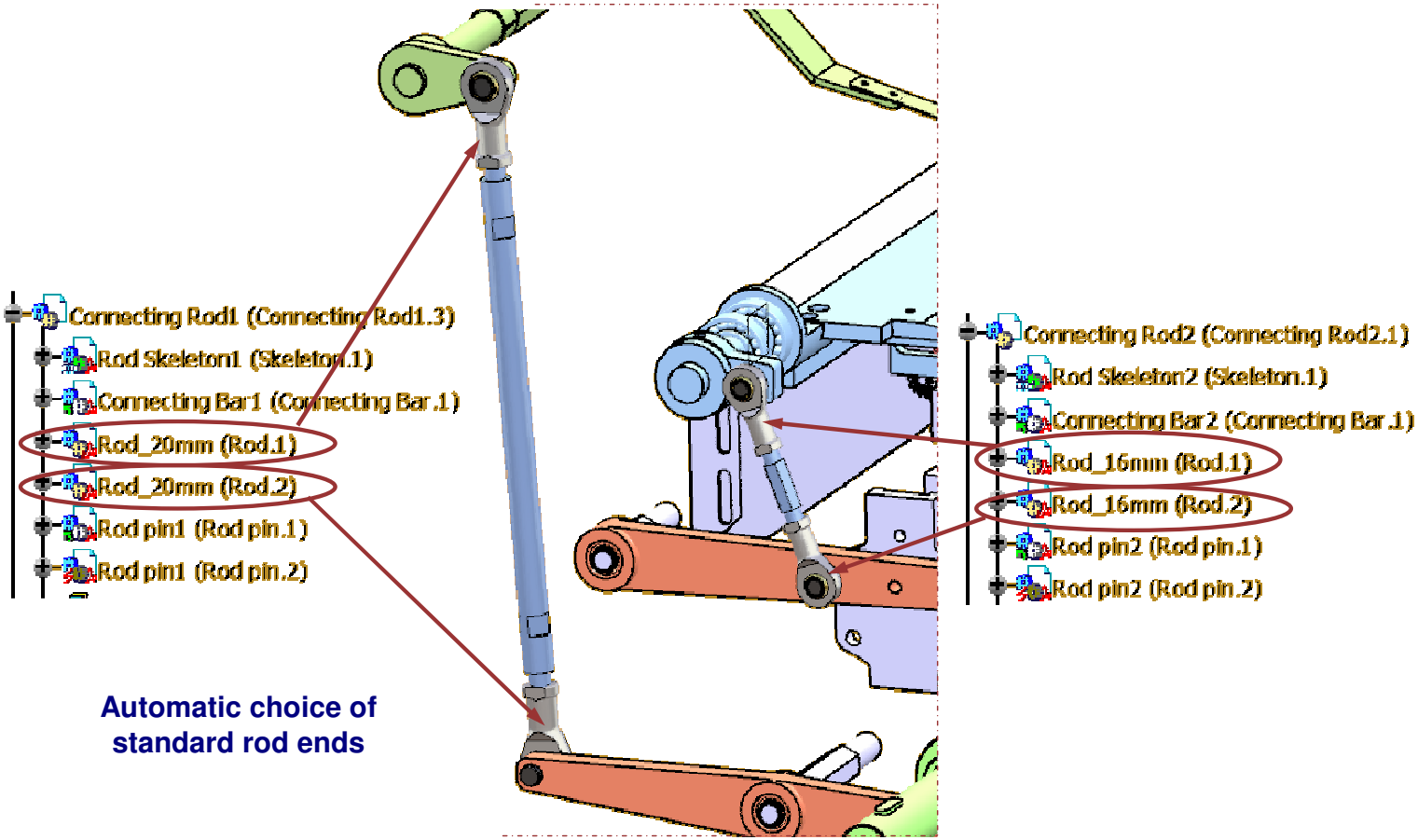
Rod ends: standard parts to be automatically replaced

Add the knowledge instructions inside a Reaction, which reacts to a skeleton's parameter value change, End Rod Diameter parameter in this case. This parameter is computed from the selected geometry at the template instantiation step.

Student Notes:

Choosing the Right Standard Component (5/5)

Case 3 example: Product structure result after document template instantiation

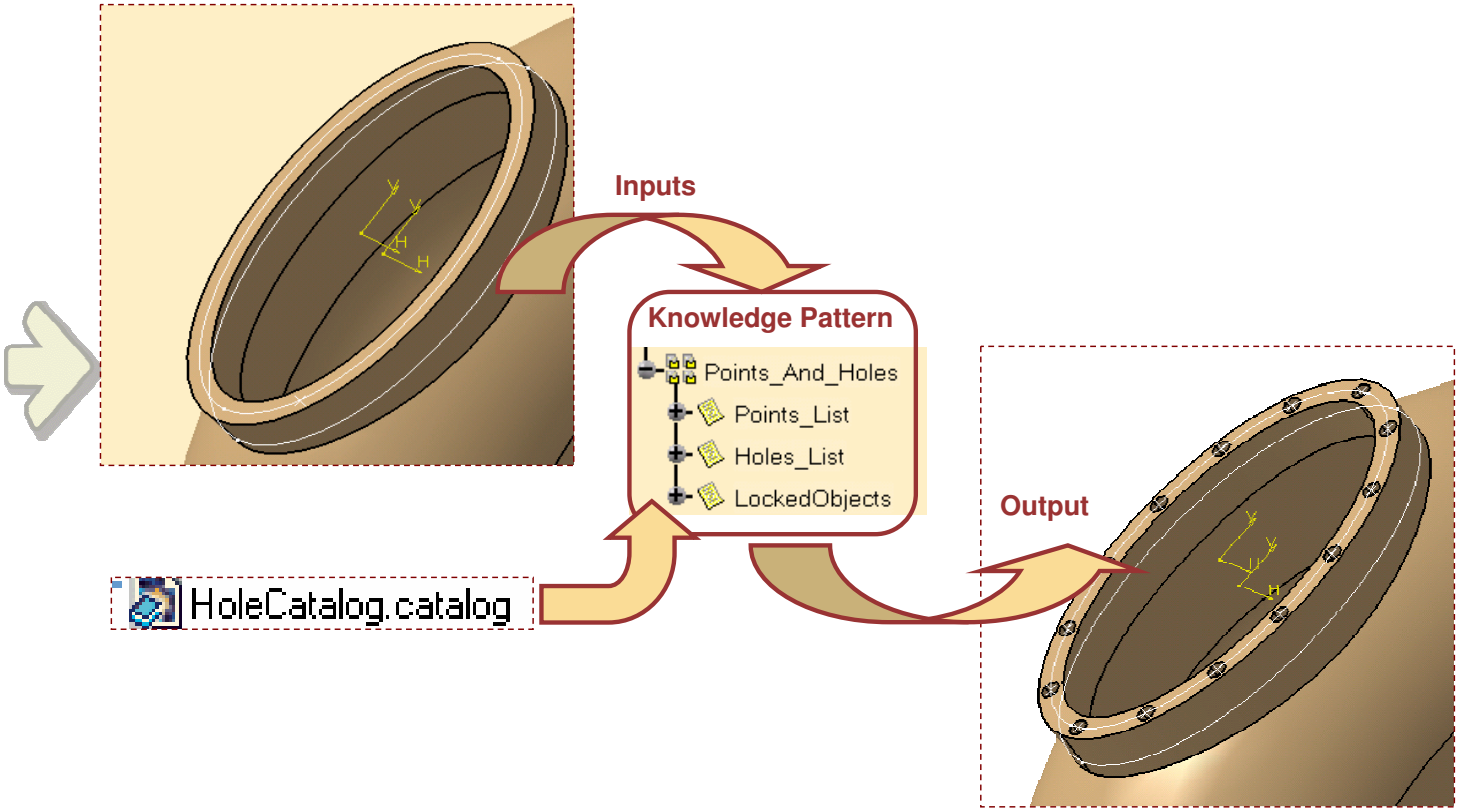


Automatic choice of standard rod ends

Student Notes:

Knowledge Pattern

In this skillet, you will learn the mechanism of 'Knowledge Pattern' and its applications. You will also learn how to create and instantiate 'Knowledge Pattern'.

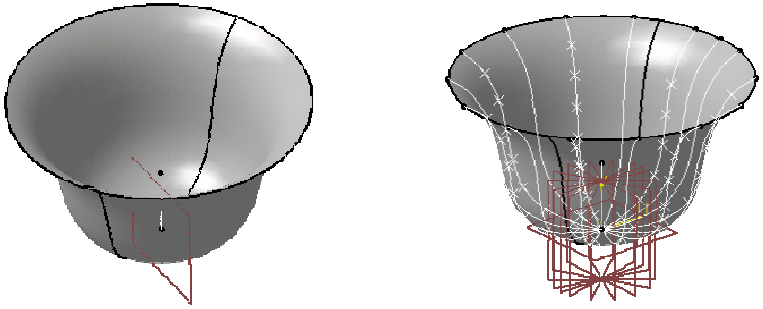


Student Notes:

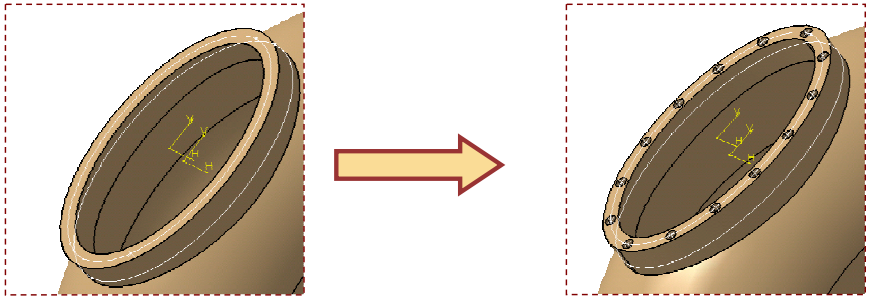
Where do we use 'Knowledge Pattern'?

Knowledge Pattern is primarily used for the following purposes:

- **To create a sequence of geometric elements resulting from certain user defined instructions.**



- **To instantiate a series of user defined features.**

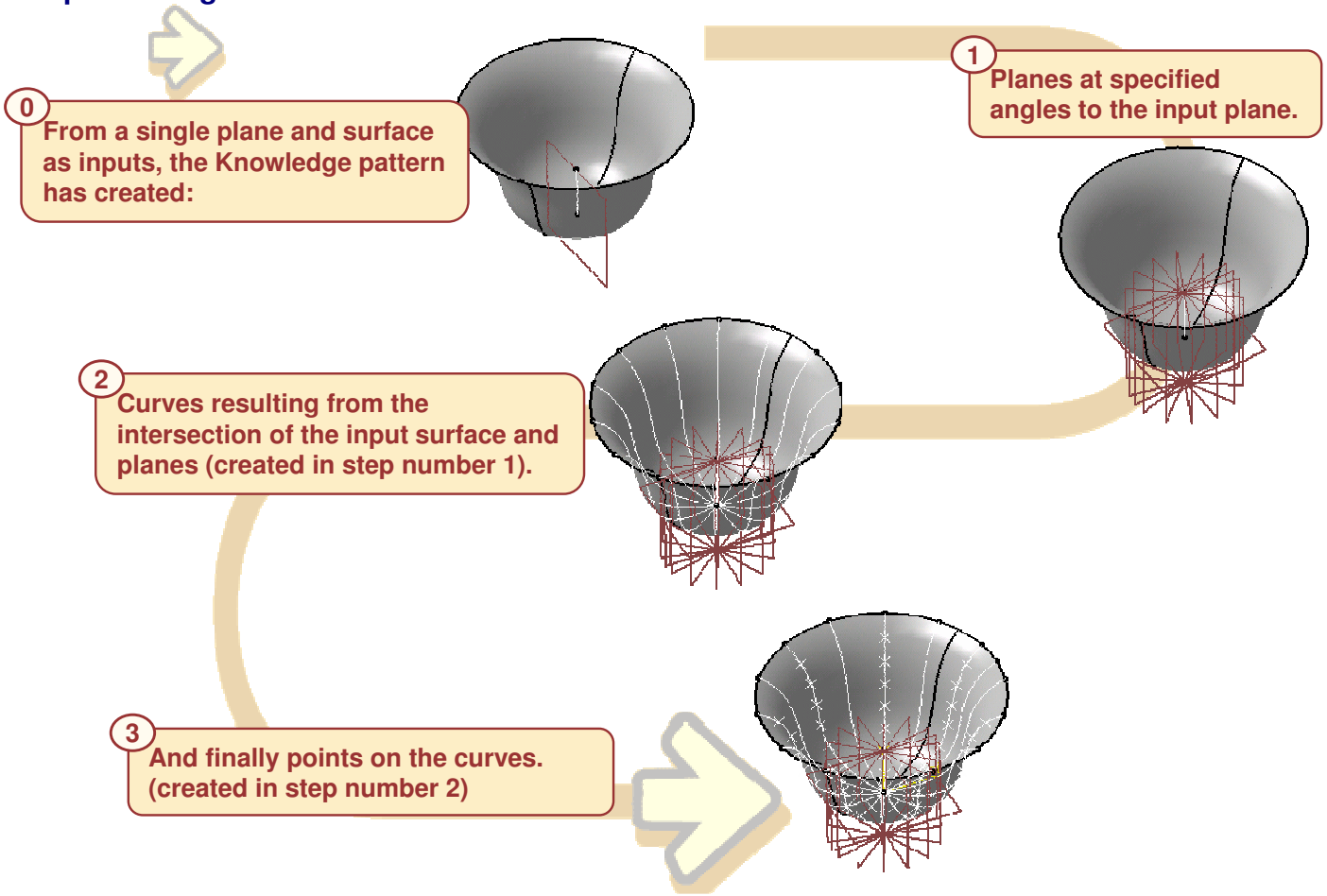


- **Knowledge Pattern must be used in place of the Knowledge Advisor Loop functionality which is now obsolete.**

Student Notes:

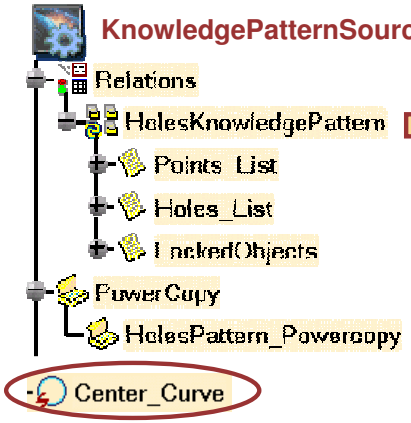
Example of Knowledge Pattern

In this example, you can notice how the 'Knowledge Pattern' is used to create sequence of geometric elements one after the other.



Student Notes:

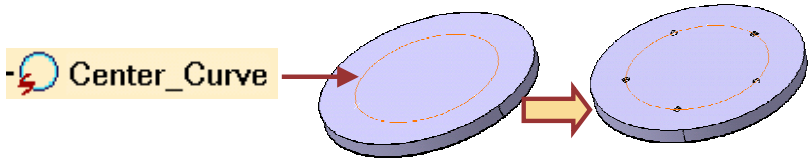
The Mechanism of Knowledge Pattern (1/2)



Contains the instantiation code based on the names of the geometric elements of the source CATPart.

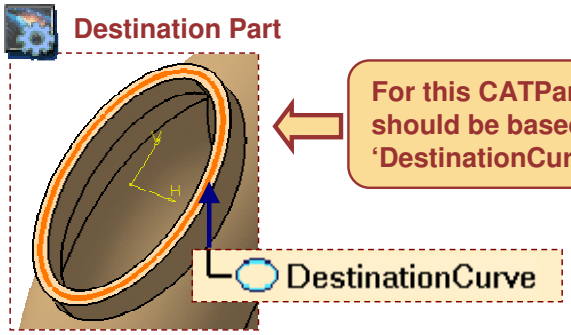
Code Instructions in source part

On Center_Curve, create five equidistant points and at every such point, instantiate a hole from the catalog file.



The mechanism of Knowledge Pattern involves geometry creation / instantiation by writing code of instructions, using the 'Knowledge Pattern Feature'.

However, if this Knowledge Pattern' is to be reused, as the names of the geometrical elements in the CATPart could be different for different CATParts, the same code of instructions may not work for other CATParts.

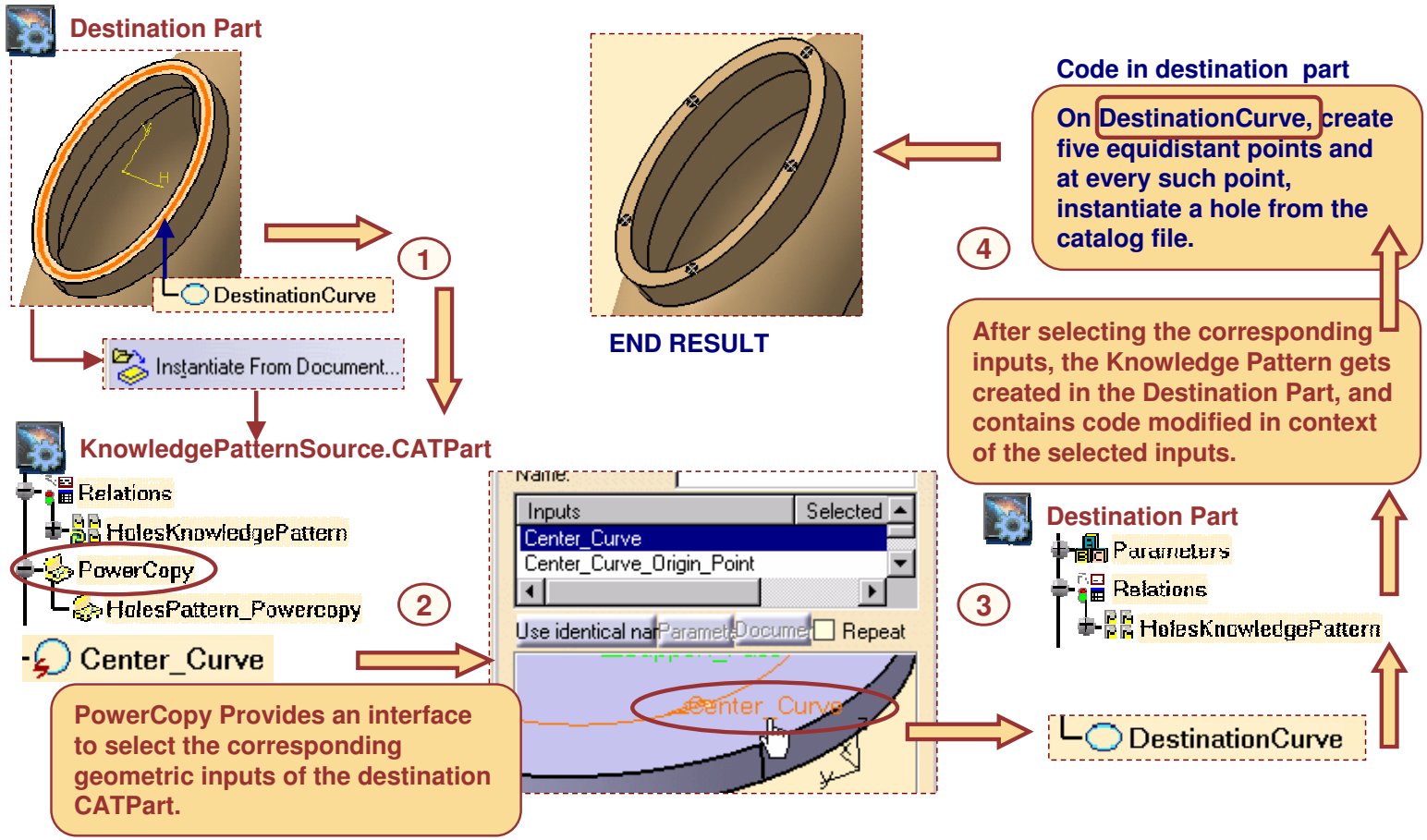


For this CATPart, the code should be based on the name 'DestinationCurve'.

On DestinationCurve, create five equidistant points and at every such point, instantiate a hole from the catalog file.

Student Notes:

The Mechanism of Knowledge Pattern (2/2)



Hence, to reuse the Knowledge Pattern, PowerCopy provides an interface to select the corresponding geometric inputs for the creation of Knowledge Pattern code in the 'Destination Part'.

Student Notes:

General Process - Knowledge Pattern – UDF Instantiation

Following is the General Process followed for creating Knowledge Pattern that involves "Instantiation of a UDF".

0. Create the UDF(s) which you want to use in your Knowledge Pattern.
1. Generate CATGScript file(s) of the UDF(s) by using the 'Type' tab of the UDF definition.
2. Export the UDF(s) to a Catalog.
3. Create the required keywords in the UDF(s), which will be used to write the code in the Knowledge Pattern for instantiating the UDF(s).
4. Save the catalog file and store it in the KnowledgeResourcesCatalogs folder.
5. Create the Knowledge Pattern in your Source-CATPart and create the required 'Lists' for it.
6. In the code field, write the instantiation code for Knowledge Pattern.
7. If required, create a 'PowerCopy' in your Source-CATPart to provide interface for selecting inputs in context of the destination Part.

Student Notes:

General Process - Knowledge Pattern – Datum Creation

Following is the general process followed for creating the Knowledge Pattern that involves creation of 'Datums'.

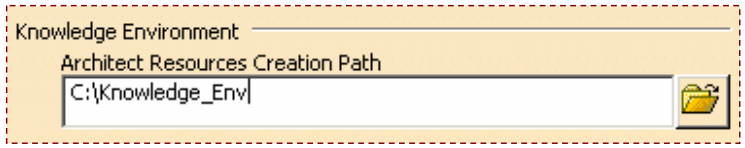
1. Create a new Knowledge Pattern feature using the Knowledge Pattern Tool.
2. Create the required lists for the Knowledge Pattern.
3. Write the code in the code field of the Knowledge Pattern and click OK on the Knowledge Pattern Creation Panel.

Student Notes:

How to Create Knowledge Pattern (1/9)

Before creating a Knowledge Pattern, you can set the folder for "Architect Resources Creation Path' folder.

This setting can be accessed in Tools > Options > General > Parameters and Measures > Knowledge Environment tab.



After doing this setting, the files which are created by Knowledge Pattern functionalities fall in this folder.

Otherwise, the files are created in the installation folder of CATIA V5. (…\intel_a\resources\Knowledge)

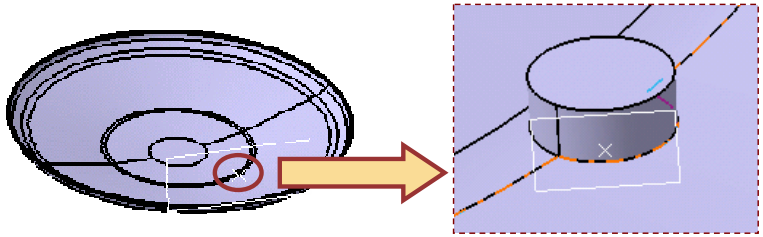
Note that you will have to restart CATIA for this setting to take effect.

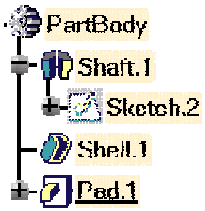
How to Create Knowledge Pattern (2/9)

Student Notes:

CATIA Data Used: Hole_UDF.CATPart

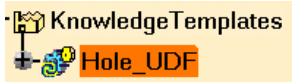
1 **Open Hole_UDF.CATPart and save it to some location.**

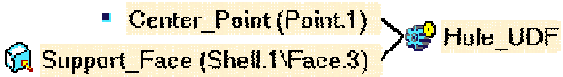




Notice Pad.1. In this example, you will instantiate Pad.1 into a destination body and use the 'Remove' Boolean operation to create a hole.

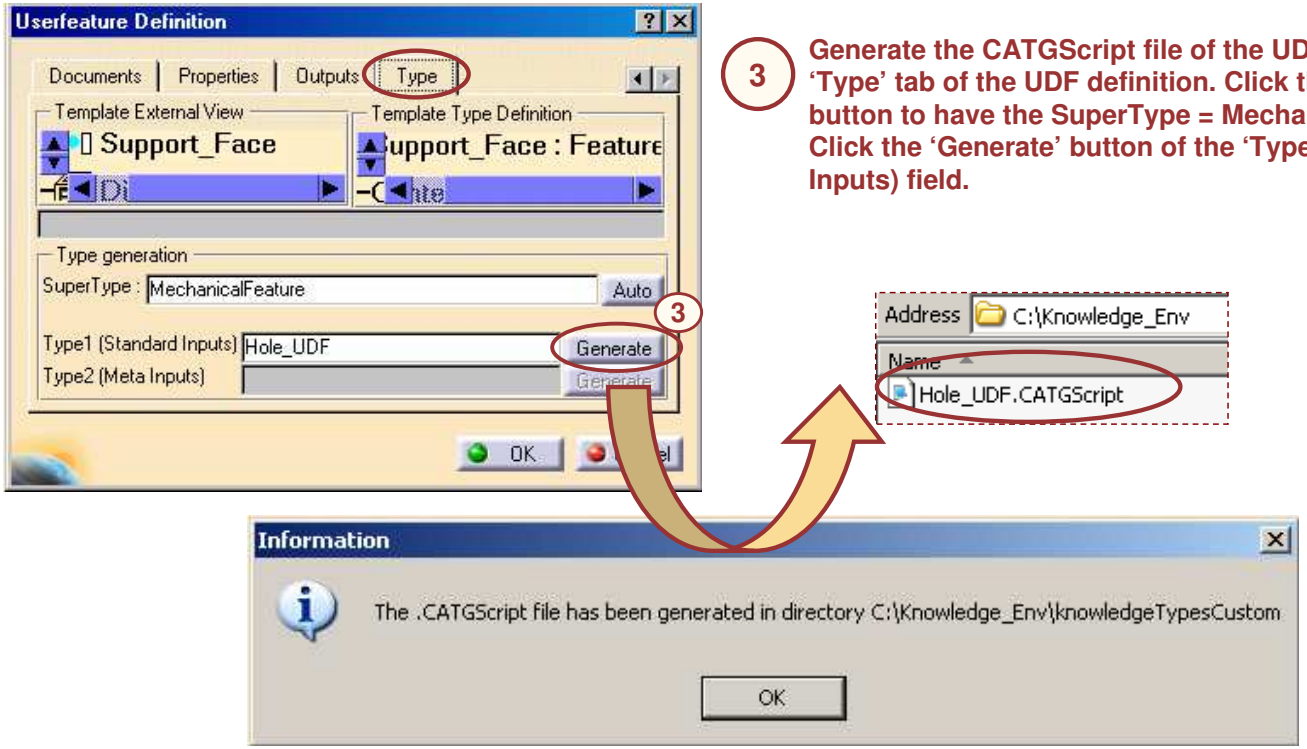
2 **Double-click the UDF – 'Hole_UDF' and notice the inputs of the UDF in the 'Inputs' tab.**





Student Notes:

How to Create Knowledge Pattern (3/9)



3 **Generate the CATGScript file of the UDF using the 'Type' tab of the UDF definition. Click the 'Auto' button to have the SuperType = MechanicalFeature. Click the 'Generate' button of the 'Type1' (Standard Inputs) field.**

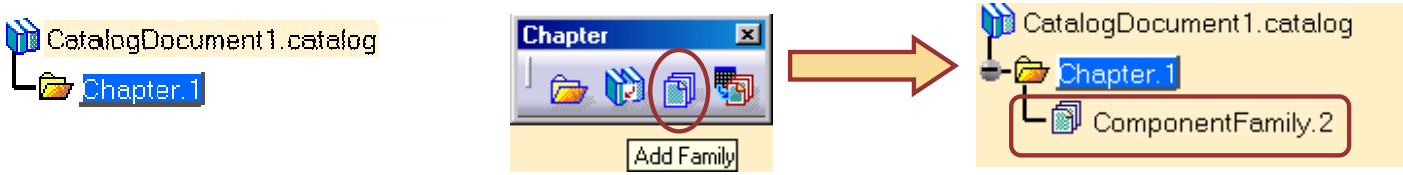
A CATGScript file will be created in the folder which is specified in the Knowledge Environment > Architect Resource Creation Path.

Student Notes:

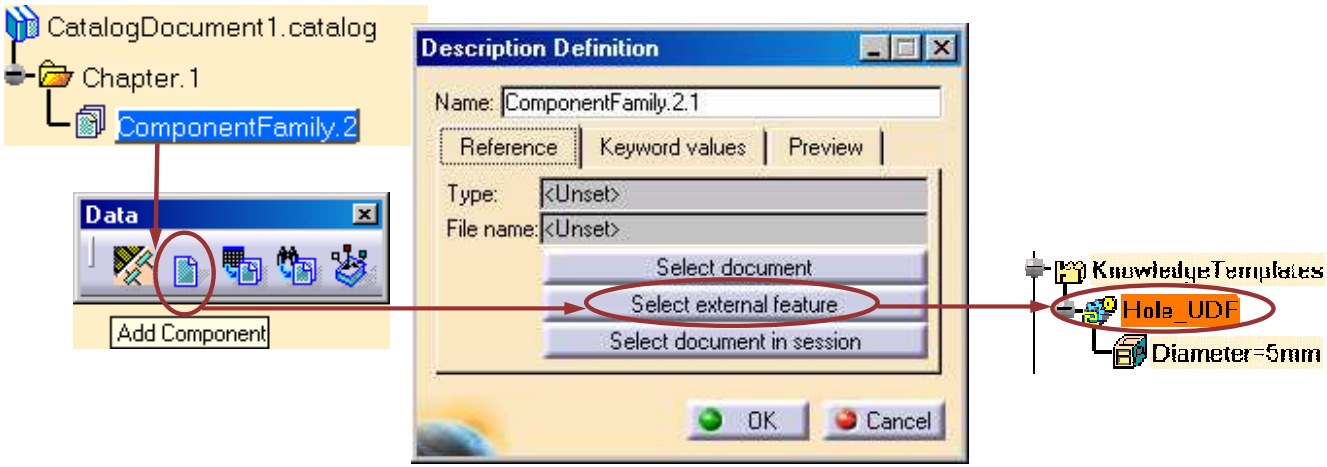
How to Create Knowledge Pattern (4/9)

4 **Export the Hole_UDF to a catalog file and create the required keywords in the catalog file.**

4a **Create a new Catalog Document and add a new family to the default Chapter.1.**



4b **Activate the component family and add the Hole_UDF feature as a component in this catalog file.**



Click 'Add Component'

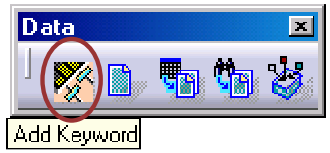
Click 'Select external feature'

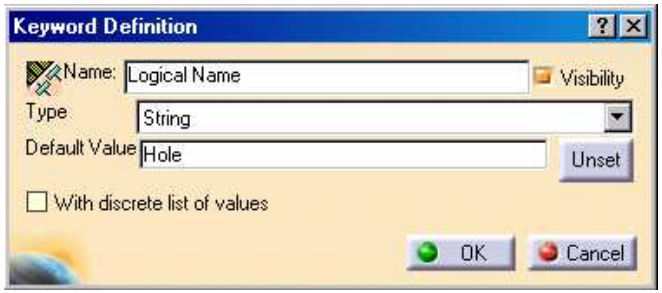
Select the UDF Hole_UDF from Hole_UDF.CATPart and click OK

Student Notes:

How to Create Knowledge Pattern (5/9)

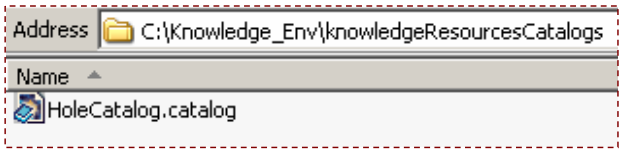
4a **Click the 'Add Keyword' tool, specify the name, type, and default value as shown, and click OK.**





The keyword name and its value is used as an identifier to write the instantiation code in the Knowledge Pattern Feature.

5 **Save this Catalog Document by the name 'HoleCatalog.Catalog' and copy it to the 'KnowledgeResourcesCatalog' folder of the "Architect Resource Creation Path"– Folder.**

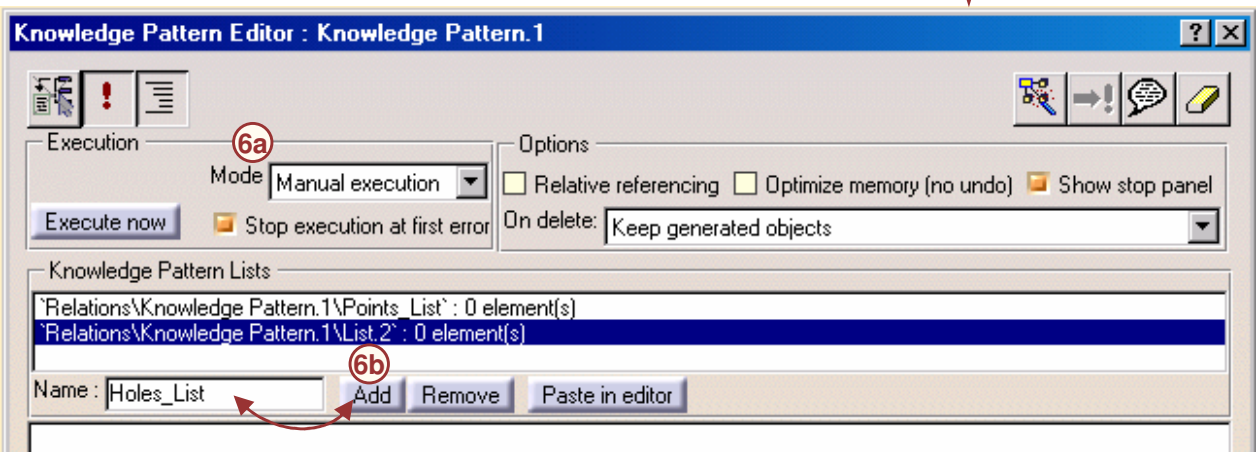


Student Notes:

How to Create Knowledge Pattern (6/9)

CATIA Data Used: KnowledgePatternSource.CATPart

6 **Open the KnowledgePatternSource.CATPart, and create a 'Knowledge Pattern Feature' by clicking the 'Knowledge Pattern Tool' of the Product Knowledge Template workbench.**



6a **In the dialog box, select the 'Manual Execution' mode.**

6b **Add the lists named 'Points_List' and 'Holes_List' for the Knowledge Pattern by clicking the 'Add' button.**

To add the lists, click the 'Add' button, type the name of the list, and again click the 'Add' button.

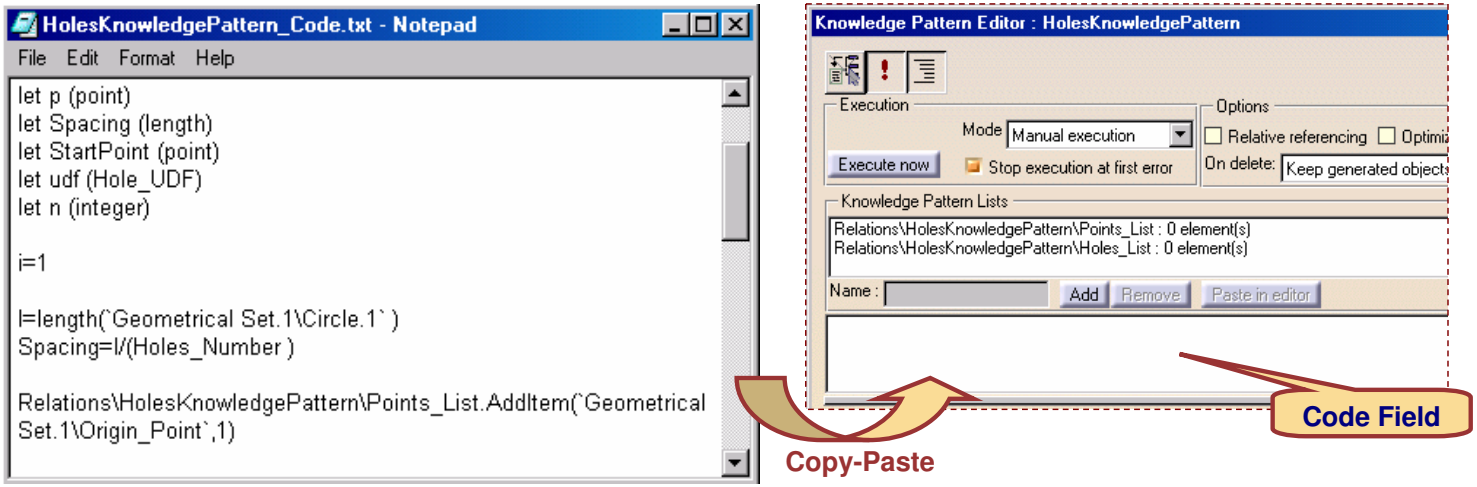
7 **After creating the required lists, click OK on the 'Knowledge Pattern Editor' dialog box, and rename the 'Knowledge Pattern Feature' to "HolesKnowledgePattern".**

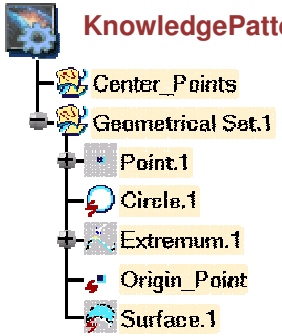
Student Notes:

How to Create Knowledge Pattern (7/9)

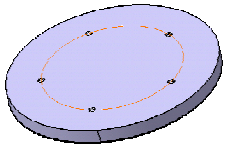
Data Used: HolesKnowledgePattern_Code.txt

8 **Double-click the 'Knowledge Pattern' feature in the specification tree, copy the code from the attached text file and paste it in the 'Code Field' of the 'Knowledge Pattern Feature' dialog box. Click OK on the dialog box.**





This code is written in context of the geometric inputs of the source part. In the above code, instructions are written to create equidistant points on the curve 'Circle.1', and to create a hole at every such point. If you run this feature, you will get the result as shown in the adjoining image.



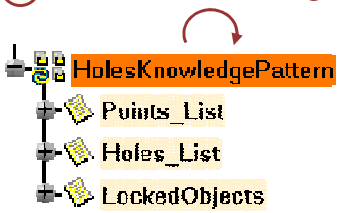
Student Notes:

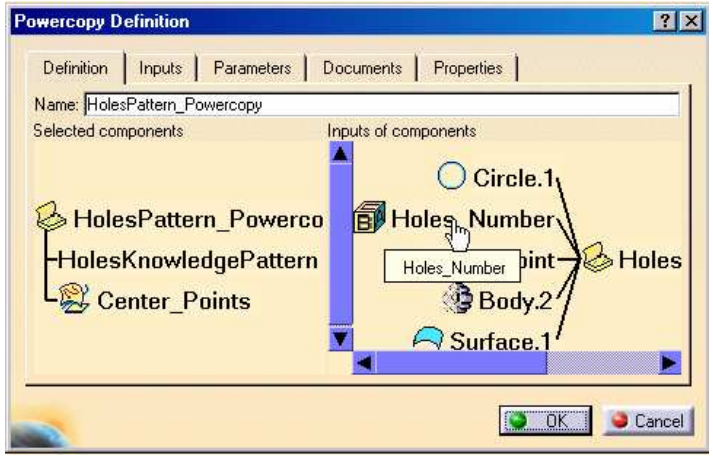
How to Create Knowledge Pattern (8/9)

9 Create a PowerCopy feature using this 'Knowledge Pattern' feature as shown.

9a Go to the Part Design workbench, from the menu, select 'Insert > Knowledge Templates > PowerCopy.

9b Select the Knowledge Pattern feature in the specification tree.





9c From the 'Inputs of components' list, click 'Center_Points' and 'Holes_Numbers' to include them in the 'Selected Components' list.

Creation of PowerCopy is not mandatory. However, if this knowledge pattern is to be reused, PowerCopy provides a convenient interface to select inputs in context of the destination part.

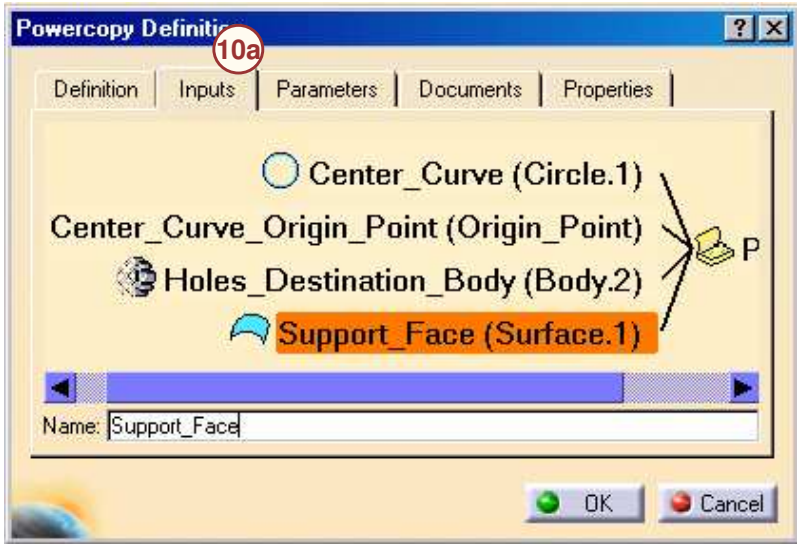
Student Notes:

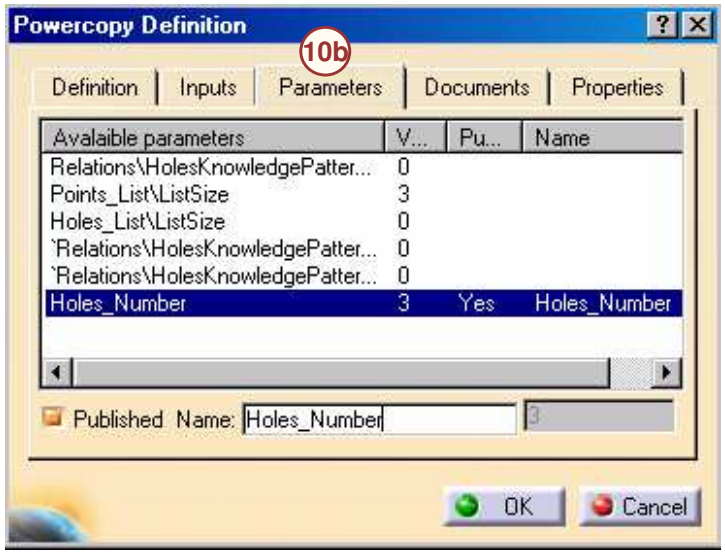
How to Create Knowledge Pattern (9/9)

10 Go to the 'Inputs' and 'Parameters' tab and make the changes as specified below.

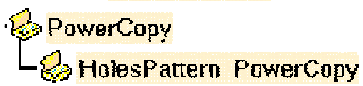
10a In the 'Inputs' tab, rename the inputs as shown.

10b Go to the Parameters Tab and publish the 'Holes_Number' parameter and click OK.





10c Rename the PowerCopy feature to 'HolesPattern_Powercopy'



Your 'KnowledgePatternSource.CATPart is ready. Save this part to some location.

Student Notes:

Knowledge Pattern – Script Explanations (1/2)

The following code will explain the method of writing the code for creation of Datums and instantiation of UDF.

```
let l (length)
let i (integer)
let j (integer)
let p (point)
let Spacing (length)
let StartPoint (point)
let udf (Hole_UDF)
let n (integer)

i=1

l=length(`Geometrical Set.1\Circle.1` )
Spacing=l/(Holes_Number )

Points_List.AddItem(`Geometrical Set.1\Origin_Point`,1)

j=2
For i while i <=Holes_Number -1
{
        set p = CreateOrModifyDatum("Point",Center_Points  ,Points_List   , j)
        p=pointoncurve(`Geometrical Set.1\Circle.1` ,`Geometrical Set.1\Origin_Point`      ,i*Spacing,True)
        p.Name = "Center_Point." + ToString(i)
        p.Show=True
        j=j+1
        i=i+1
}
n=1
For n while n<=Points_List  ->Size()
{
        udf=CreateOrModifyTemplate("HoleCatalog|Hole",Body.2  ,Holes_List, n)
        udf.Center_Point = Points_List  ->GetItem(n)
        udf.Support_Face=`Geometrical Set.1\Surface.1`
        EndModifyTemplate(udf)
        udf.Name="Hole."+ToString(n)
        n=n+1
}
```

Variables' Declaration

Use of UDF type defined in the CATGScript file

Loop creations with number of instances to generate

Code for generating Datums

Creation of Datum Point

Code for UDF – Template instantiation using the ARM catalog

Student Notes:

Knowledge Pattern – Script Explanations (2/2)

The following code explains the syntax of UDF instantiation and specification of the inputs for the UDF.

```
For n while n<=Points_List  ->Size()
{
        udf=CreateOrModifyTemplate("HoleCatalog|Hole",Body.2  ,Holes_List, n)
        udf.Center_Point = Points_List  ->GetItem(n)
        udf.Support_Face=`Geometrical Set.1\Surface.1`
        EndModifyTemplate(udf)
        udf.Name="Hole."+ToString(n)
        n=n+1
}
```

Instantiation of UDF from the HoleCatalog.catalog file

Specification of the inputs for the UDF

Student Notes:

How to Reuse Knowledge Pattern (1/3)

CATIA Data Used: Hole_UDF.CATPart, HoleCatalog.Catalog, KnowledgePatternSource_Ready.CATPart

Other Files Used: Hole_UDF.CATGScript

- **Prerequisites for reusing Knowledge Pattern involving UDF instantiation.**

For reusing a Knowledge Pattern, you should have created the knowledge pattern, its Power Copy, UDF part and respective catalogs. However, these inputs have been kept ready for you. Following are the prerequisites to reuse the Knowledge Pattern for the example used in this case.

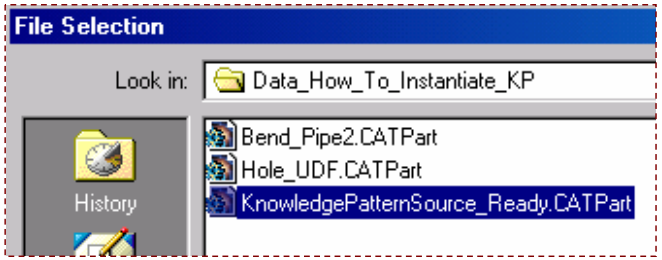
1. **Copy the Hole_UDF.CATPart file to your "c:\temp" directory.**
2. **Copy the HoleCatalog.Catalog file to the "\knowledgeResourcesCatalogs" folder of the Knowledge Environment – 'Architect Resources Creation Path' (which is specified in the User Settings)**
3. **Open the HoleCatalog.Catalog file and verify that it points to the Hole_UDF feature of Hole_UDF.CATPart that you have stored at "c:\temp"**
4. **Copy the Hole_UDF.CATGScript file to the "\knowledgeTypesCustom" folder of Knowledge Environment – 'Architect Resources Creation Path' (which is specified in the User Settings)**
5. **In this example, you will be instantiating the knowledge pattern from the source CATPart – KnowledgePatternSource_Ready.CATPart. Save this part to some location and close it.**

Student Notes:

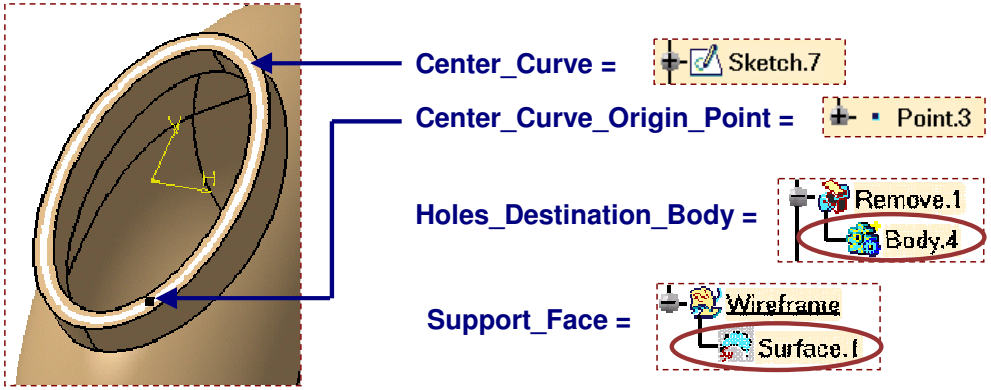
How to Reuse Knowledge Pattern (2/3)

CATIA Data Used: Bend_Pipe.CATPart

1. **Open the Bend_Pipe.CATPart and from the menu, select – Insert > Instantiate from Document and select the file "KnowledgePatternSource_Ready.CATPart".**



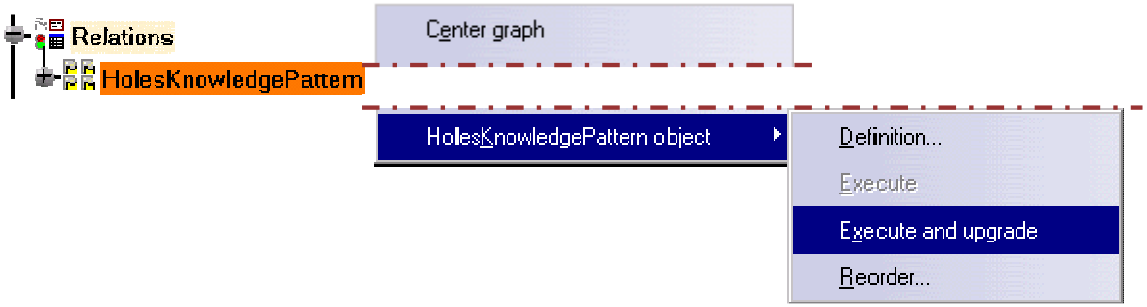
2. **Select the inputs in context of the destination part (Bend_Pipe.CATPart) as shown below.**



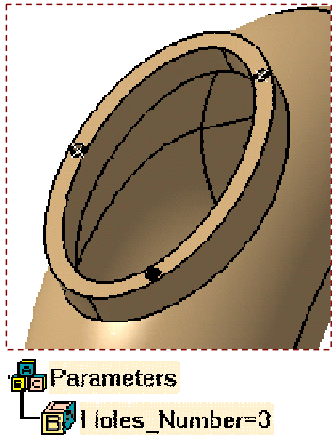
Student Notes:

How to Reuse Knowledge Pattern (3/3)

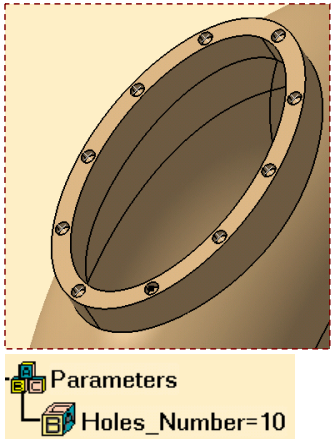
3 **Knowledge Pattern will be created in the specification tree. Click on the Knowledge Pattern in the specification tree, and from the contextual menu select 'Execute and upgrade'.**



4 **Update the part using the menu Edit > Update and notice the three points and holes that are instantiated.**



You can increase the number of holes by increasing the value of the parameter 'Holes_Number'.

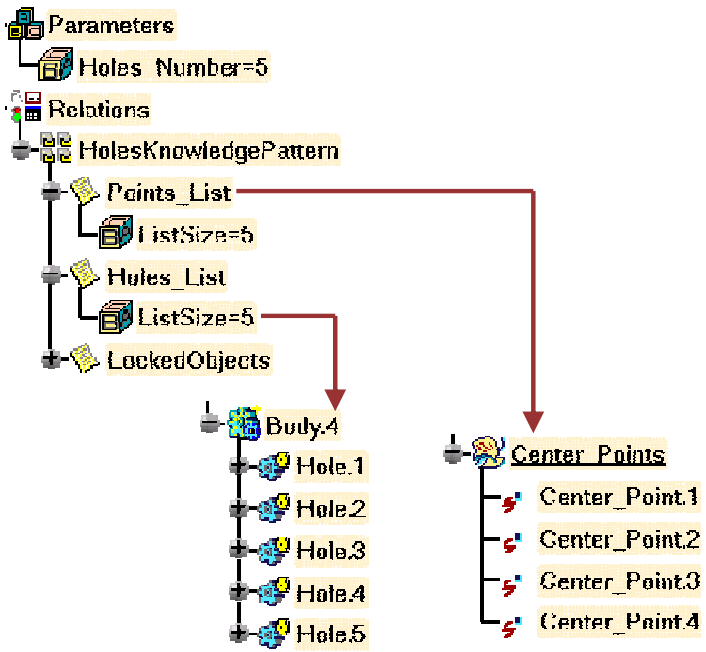


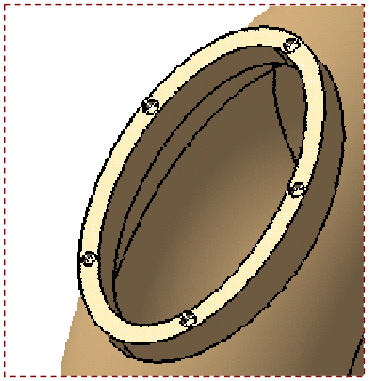
Student Notes:

Additional Information – Knowledge Pattern (1/3)

- **Knowledge pattern supports the instantiation / creation of the following objects:**
  - **User Features**
  - **Datums (Planes, Points, Lines, Circles, Curves, Surfaces, and Volumes)**

- **Lifecycle of the objects instantiated by Knowledge Pattern:**
  - **The Lifecycle of the objects instantiated by Knowledge Pattern is controlled through the contents of the 'Lists' that are created in the Knowledge Pattern Feature.**

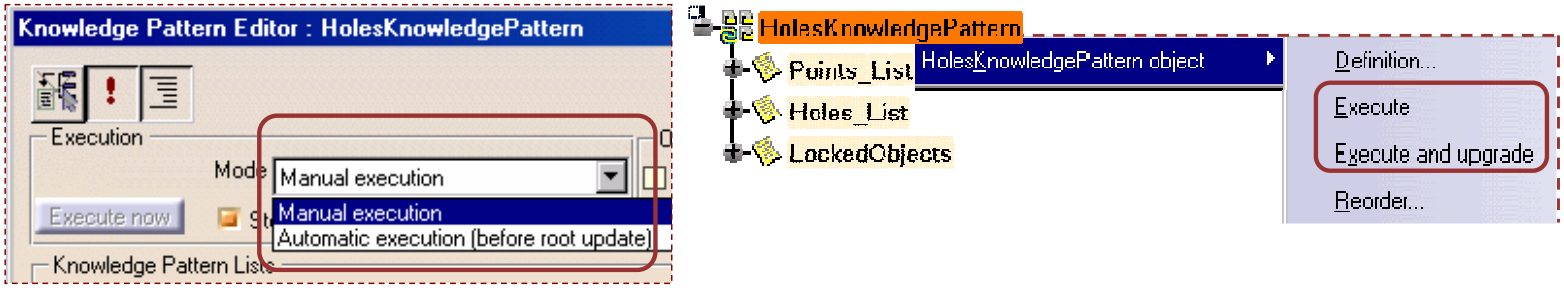




Student Notes:

Additional Information – Knowledge Pattern (2/3)

- **Execution Mode for Knowledge Pattern:**



- **In the 'Manual execution' mode, you have to execute the Knowledge Pattern every time you make changes in the code or any parameters related to the Knowledge Pattern.**

- **In the 'Automatic execution' mode, the Knowledge Pattern automatically gets executed when you click OK on the 'Knowledge Pattern Editor' dialog box.**

Student Notes:

Additional Information – Knowledge Pattern (3/3)

Knowledge Pattern assures associativity of the instantiated objects.

- **The instantiated objects are associative with respect to:**
  - **The inputs defining the Knowledge Pattern**
  - **The number of instances**
  - **The generated datums**
  - **Parameter values of the instances**

To Sum Up

In this lesson, you have learned:

- **The concept of 'Knowledge Pattern'**
- **Its applications**
- **Methods to create and reuse 'Knowledge Pattern'**
- **Guidelines to write the code of 'Knowledge Pattern'**

Student Notes:

Student Notes:

Summary

In this course you have learned the following topics:

- **Creating User Parameters, Formulae, Lists, Rules, Checks and Reactions.**
- **Creating and Managing Design Tables.**
- **Using Knowledgeware Advisor Tools like ‘Knowledge Inspector’, ‘Set of Equations’, ‘Laws’.**
- **Creating and re-using PowerCopies and User Defined Features.**
- **Creating and re-using Part and Assembly Templates.**
- **Using advanced replication and instantiation tools like Knowledge Pattern.**