Machining Knowledge Editor Training

Proprietary and restricted rights notice

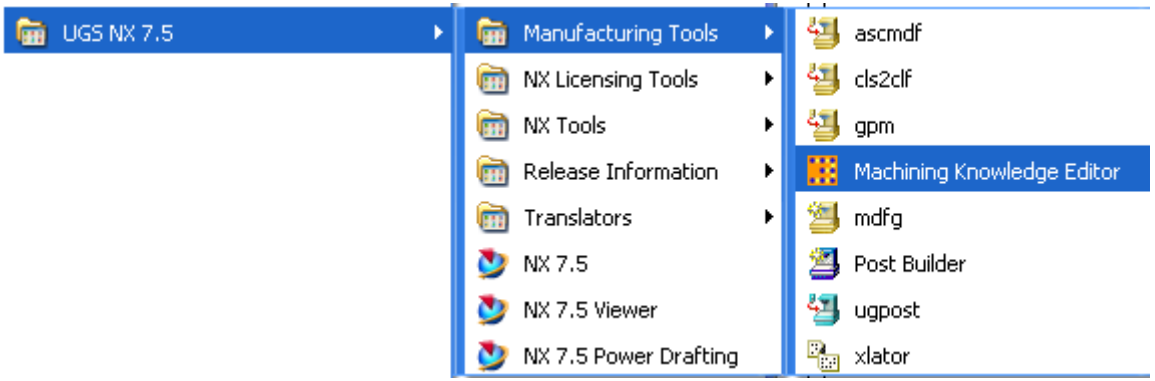
CHAPTER 1

Machining Knowledge Editor Concepts ***1***

CHAPTER 1

Machining Knowledge Editor Concepts

The first chapter explains the concepts behind the Machining Knowledge Editor.



Best practice definition. The example of a hole.

There are many ways to machine a hole. It will depend on the hole's diameter, its depth, the ratio depth/diameter, the quality of the cylindrical surface, the tolerance of the diameter, and more.

We can create a hole using a twist drill, a reamer, a boring bar, a mill, or some other type of tool. Sometimes we start directly from the surface. Sometimes we must start from an existing hole or a centerpoint.

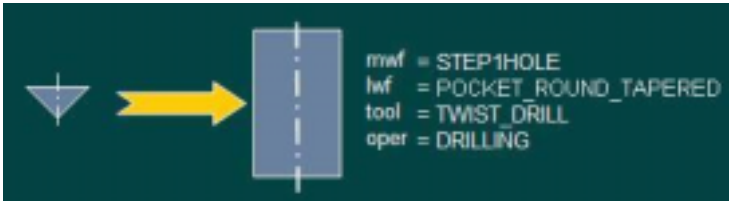
What applies to holes, also applies to most other features: generally there will be alternative ways to have them manufactured.

A company's best practice defines their methodology of machining certain manufacturing features. It defines, for a set of machining features, which operations should be used, which tools should be used, and in which order these should be applied. The best practice is defined by sets of Machining Rules.

In addition to the day-to-day advantage of generating fast, reliable and reproducible solutions, having a best practice also serves as a vehicle for standardization.

Machining Rule concept

A Machining Rule defines how a manufacturing feature (**M**ore **W**orked **F**eature) can be machined starting from another feature (**L**ess **W**orked **F**eature) using a tool of a certain class.



In general, a Machining Rule will correspond to a manufacturing operation (e.g. drilling a hole or milling a pocket) and will describe:

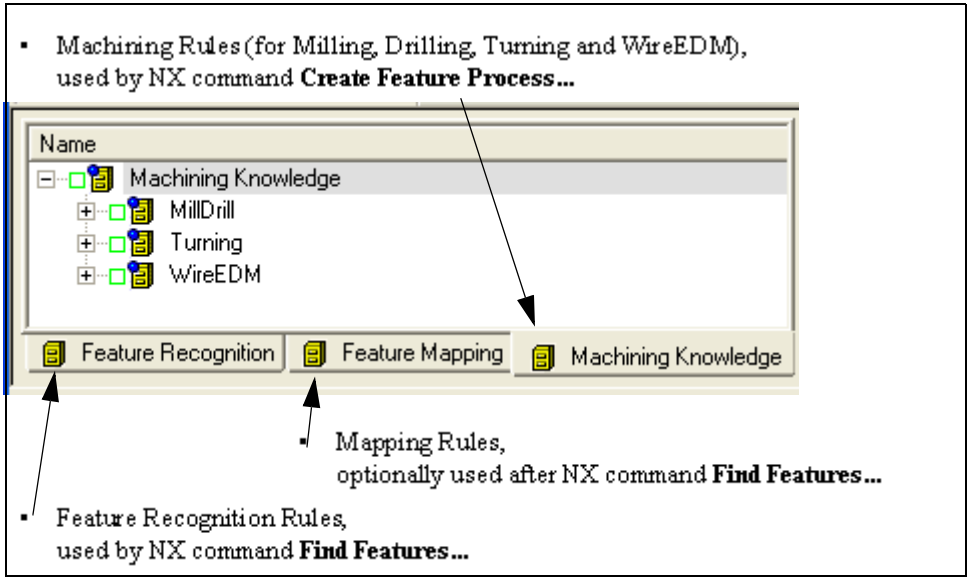
- when this operation is allowed
- what result can be achieved (which shape is created, which tolerances can be achieved)
- which tool is necessary (which type is required, what dimensions are required)
- which start shape needs to be there.

For brevity, in this training when we talk about a ‘Rule’ we actually mean a ‘Machining Rule’.

RuleLibrary

Rules are always grouped in a RuleLibrary. Three different RuleLibraries, with different purposes, are used in NX:

1. The ‘Machining Knowledge’ library, which is used by the ***Create Feature Process*** command. This RuleLibrary defines the best practice to manufacture features.
2. The ‘Feature Mapping’ library, which is used by the ***Find Features*** command. This RuleLibrary defines how a feature of one type can be changed (mapped, transformed) into a feature of another type. This functionality can be used to transform external or modeling features (User Defined Features) into manufacturing features and vice versa. See “Feature Mapping” on page 123.
3. ‘Feature Recognition’ rules. These are outside the scope of this tutorial.



In the Machining Knowledge Editor release 7.5 you will see 3 different Tabs for the 3 different Rules you can define. The names of the Tabs and the Tab order are fixed.

Editing machining and mapping rules does not require any knowledge or skills in a particular programming language like C or C++.

The Machining Rules are used by the inference engine which is dedicated to searching a solution to machine manufacturing features. Machining Rules are the elementary process steps and the application of a Machining Rule will generally result in a single NX operation object. The inference engine will be discussed in “How Operation Selection works” on page 39.

Starting the Machining Knowledge Editor

Rules are defined with the dedicated Machining Knowledge Editor application.

Every user can view and edit the Rules. Protection can be arranged by setting special file permissions on the relevant files/directories or are handled by Teamcenter when you are working in managed mode.

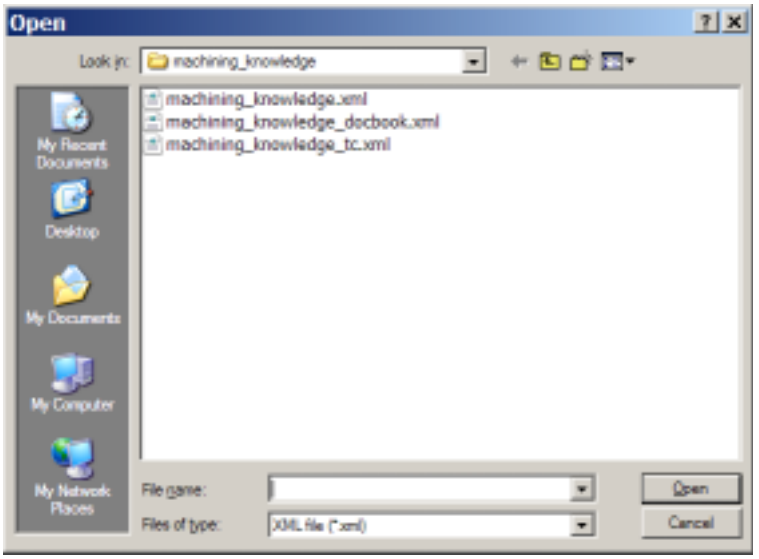
The Machining Knowledge Editor is opened from the start menu ***Programs > UGS NX Version > Manufacturing Tools > Machining Knowledge Editor***.

Activity: Open the Machining Knowledge Editor

- *Start the application*
- *From the menu bar, choose View > Options*
- *Inspect tab Exchange Adapter*
- *Inspect tab TCIN. When you work in managed mode, you must check active ‘Use Teamcenter Integration’. When not in managed mode, you will be reading and writing to the file system. In managed mode, the 2-tier or 4-tier connection parameters must be set. Further details can be found in the **NX** Help for the Machining Knowledge Editor.*

Loading the Rule Library

The first step is to load the RuleLibrary. Either use ***File*** **>** ***Open*** in the main menu, or click on the ***Open*** icon in the toolbar. This will open a dialog in which a .xml file can be selected and opened.



Note Screenshot is taken when working in native mode.

By default, the dialog will open in the directory defined by the system variable UGII_CAM_MACHINING_KNOWLEDGE_DIR. Recently used libraries can be found by clicking on the ‘My Recent Documents’ icon.

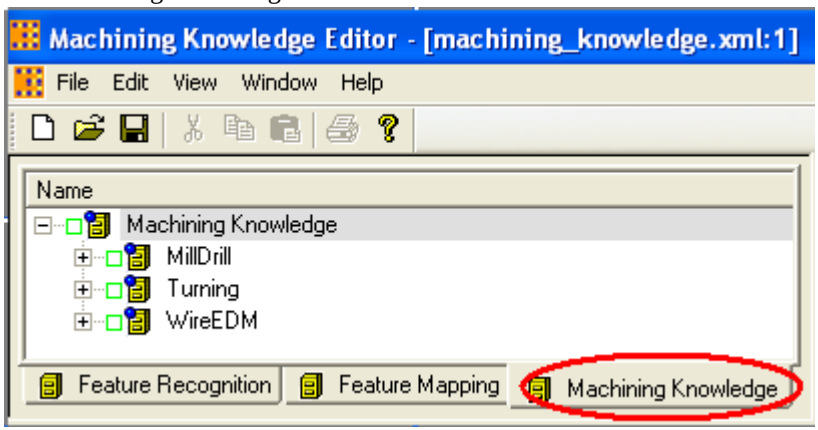
You can open multiple .xml files in the Machining Knowledge Editor. This allows for easy copying of data between libraries.

Activity: Load a Machining Knowledge Library

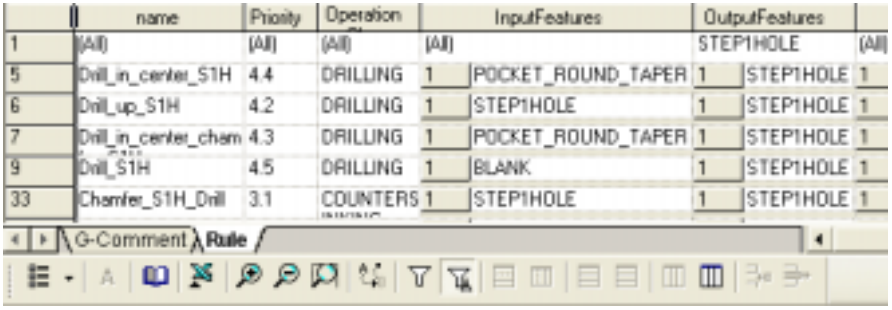
- *Open the ‘machining_knowledge.xml*

The Machining Knowledge Editor User Interface

The Machining Knowledge Editor User Interface consists of three sections:

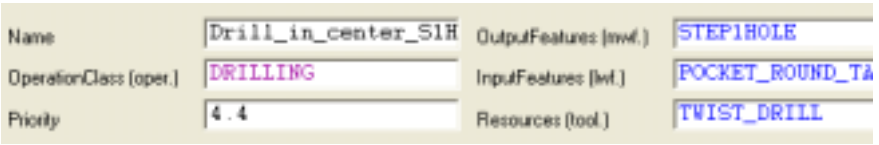


- The Tree View showing the RuleLibrary tree. This lists the names of the Rules. The Tree View is mainly used for finding and activating nodes, and for starting specific commands.
  The three Tabs (new in NX 7.5) allow to switch between modes for Machining Rules and Feature Recognition Rules and Feature Mapping Rules.
- The Table View showing the activated nodes.

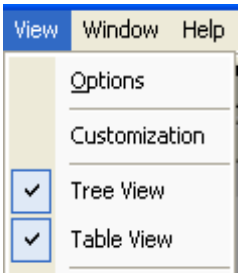


The Table View is mainly used for finding and filtering on Rules.

- The Rule View showing the component data of a selected Rule.



This view is used to define a Rule.



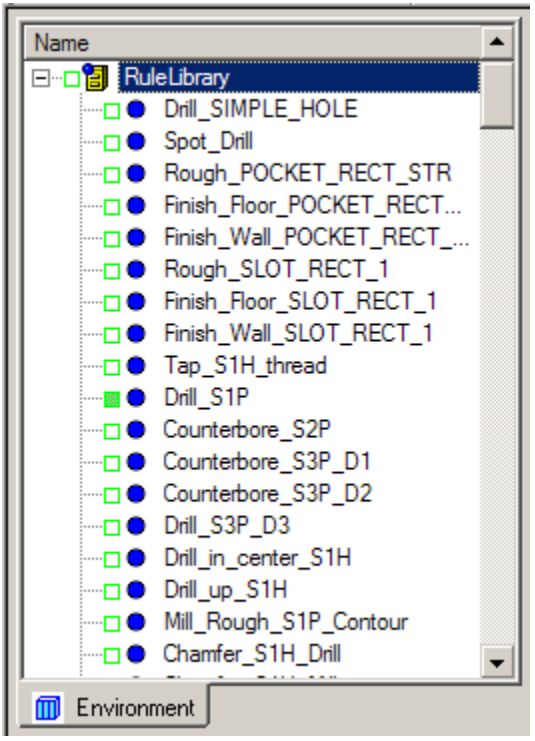
The display of the Tree View and Table View can be toggled from the pull-down menu ***View***. From the sub menu select the name of the view. This will toggle the display of the view. When checked active, it is displayed.

Activity: Change the visibility of the views

- *Toggle the visibility of the views with the **View** entry in the main menu.*
- *Display all views again.*

Tree View

The Tree View lists the Rules in a tree format. The Tree View is mainly used for finding and activating nodes, and for starting specific commands.



Activity: Navigating the tree

- *move through the tree use the arrow-up, arrow- down, Home, End, Page Up, and Page Down keys.*
- *Sections of the tree can be expanded or collapsed by clicking with the left mouse button on the +/- sign shown in front of the node name.*
- *Sections of the tree can be expanded or collapsed by using the left and the right keys on the keyboard.*

Selecting nodes

Most commands are executed on one or more nodes in the Tree View or Table View. If a command is started from the Tree View, only the selected nodes will be used as input for the commands.

In the Tree View all nodes of the project can be selected. In the Table View only the activated nodes can be selected (as these are the only nodes which are displayed in this view).

Selected nodes will be highlighted in the views where the nodes are visible. Nodes, which are (de)selected in one view (e.g. Tree View), are also (de)selected in the other view (Table View).

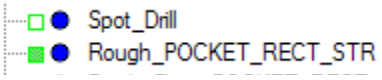
Activity: Selecting Nodes in the Tree View

- *click MB1 on a node (active or not active) for single select.*
- *Control + click MB1 to modify the selection. The node will be added or removed from the current selection.*
- *Shift+MB1 allows multiple selection. All nodes starting from the previously selected node will be added to the current selection.*
- *Shift+Up or Shift+Down allows multiple selection. A new selection will be started from the first node where the shift was pressed.*

Activating nodes

Only activated nodes are visible in the Table View.

The status (activated/deactivated) is shown with the check box in front of the node.



Activity: Activating Nodes in the Tree View

- *Click with the left mouse button on the check box in the Tree View to toggle the status of the node. If the node is part of a selection, all other nodes will be set to the same activation status regardless of their current status.*

- *Press the space bar to reverse the status of the node. If the node is part of a selection, the status of all other nodes will also be reversed from their current status.*
- *All children of a node can be activated from the Tree View popup menu by executing the **Activate Children** command.*
- *All nodes can be deactivated from the Tree View popup menu by executing the **Deactivate All** command.*

Renaming nodes

In the Tree View a node can be renamed by first selecting it, followed by another mouse click on the name of the node or by pressing F2. This will start the editing mode indicated by the box around the name. The changes can be cancelled by pressing the 'Esc' button. The editing mode will be closed when clicking outside the editing box.

Activity: Rename a node

- *Rename the 'RuleLibrary' node into 'Default RuleLibrary'.*

Ordering nodes

In the Tree View nodes can be moved by making a selection, dragging this selection with MB1 pressed within the Tree View on one of the other nodes, and releasing MB1. This will move all selected nodes to a position below the node that the selection was dragged on.

Activity: Moving nodes within the Tree View

- *Move some of the nodes within the Tree View.*

Copying nodes

In the Tree View nodes can be copied by making a selection, dragging this selection with MB1 pressed within the Tree View on the RuleLibrary node, and releasing MB1. This will copy all selected nodes and add them at the end of the tree. All data related to a node will be copied to the new node. This enables to quickly create a new Rule from an existing Rule.

Activity: Copy a node

- *Copy some of the nodes within the Tree View.*
- *Rename these nodes.*

Deleting nodes

In the Tree View nodes can be deleted by making a selection, and pressing the ‘Delete’ button or by selecting ‘Delete’ from the popup menu. This will delete all selected nodes.

Activity: Delete a node

- *Delete the nodes that were created in the previous exercise.*

Table View

The Table View shows the data of activated nodes in an 'Excel' style tabular representation. The Table View is mainly used for finding and filtering nodes.

Selecting nodes in Table View

Nodes in the Table View can be selected in one of the following ways:

- ***MB1*** on the beginning of a row is single select. If the mouse is moved while the MB1 is pressed (and you should see an arrow cursor), multiple lines will be selected.
- ***Control + MB1*** on the beginning of a not-selected row modifies the selection. The node will be added to the current selection.
- ***Control + MB1*** on the beginning of a selected row modifies the selection. The node will be removed from the current selection.
- ***Shift+MB1*** on the beginning of a row allows multi selection. All nodes from the previous selected node will be added to the current selection.
- ***MB1*** on the top left Cell will select all rows in the Table View.

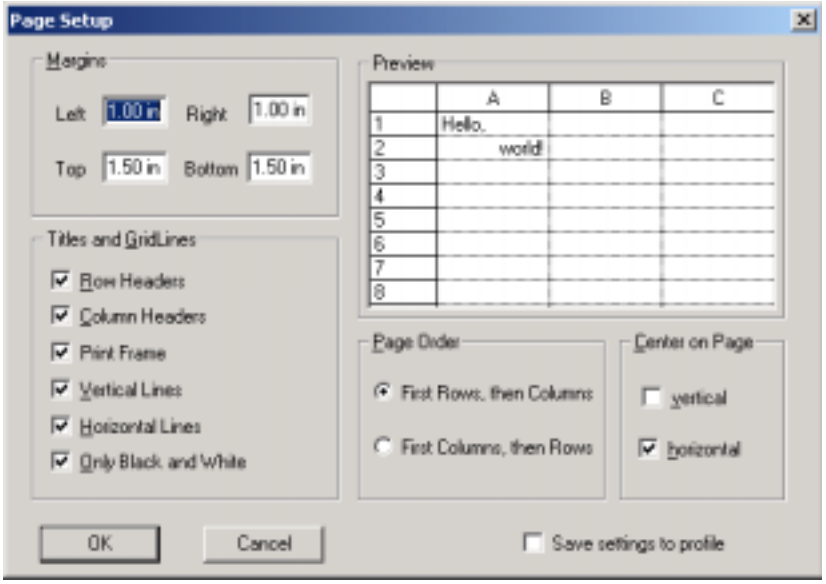
Commands in Table View

Node specific commands can be started from the popup menu of the Table View. This popup menu only appears when using MB3 on the beginning of the row.

Table View Toolbar

The Table View toolbar contains the following settings and commands for the Table View:

- ***Page Setup of Table View***: Pops up the Page Setup window for the Table View. In this window the setup of the Table View can be changed.



- ***Export Table View to Microsoft Excel***: Saves the content of the Table View to an MS Excel file format.
- ***Zoom in Table View by 20%***: does what it says.
- ***Zoom out Table View by 20%***: does what it says.
- ***Reset Zoom***: Resets the zoom of the content of the Table View to 100%
- ***Replace***: Searches the column for entries containing the search string and replaces the search string with the replacement string. The search string does not support wildcards.

- ***Filter***: Hides all rows that do not have data matching the search string in the selected column. To display all rows again, the filter button has to be pressed again. If a cell is selected instead of a column, then the data of this cell is used to preset the search string for filtering the column containing the cell. The search string supports wildcards:
  - `*` replacing zero or more characters
  - `?` replacing exactly one character
  - `;` separating elements of a vector

- ***Autofilter***: Activating the AutoFilter adds a new row to the TableView just below the top row with the column names. Clicking with the mouse in one of the fields in this AutoFilter row shows a menu with all different values in the column. Selecting one of the values will show only the rows with that value and filter out all rows with other values. Upon deactivating the AutoFilter the AutoFilter row is removed and the filters are reset, i.e. all rows are shown again.
- ***Toggle Freeze rows/ Toggle Freeze columns***: moves the selected rows/ columns to the first positions, where they will remain visible when scrolling through the Table View. Pressing the button again will restore the normal display of the rows/columns.
- ***Hide rows / Unhide rows***: Will hide the selected rows. Unhide rows will show all hidden rows again.
- ***Hide columns / Unhide columns***: Will hide the selected columns. Unhide columns will show all hidden columns again.

Activity: Using some Table View commands

- *Activate all Rules.*
- *In the 'name' column find the nodes starting with 'Tap' by using the filter 'Tap*'*
- *Use the filter wildcards command on the 'name' column to display all rows that have 'S1H' at the beginning, at the end, or somewhere in the string.*
- *Use the freeze/hide commands.*
- *Use the Autofilter command to filter on the OperationClass and the OutputFeatures column values.*

Sorting cells

In the Table View the rows can be sorted by attribute by clicking on the name of the column.

Editing cells

In the Table View the attributes of nodes can be edited. Cells are:

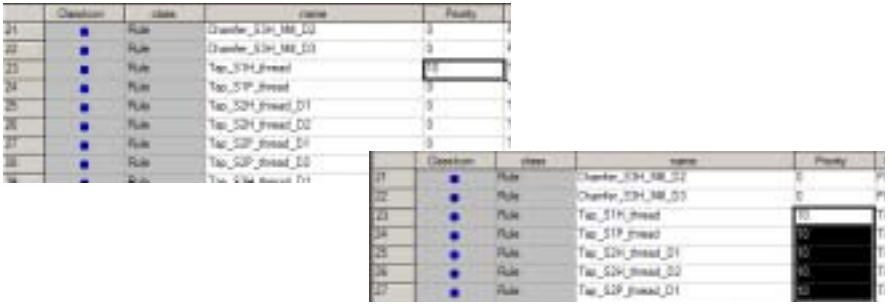
- white, if the value in the cell can be edited,

- light gray, if the value in the cell is read-only,
- dark gray, if the attribute does not exist for the node.

The Table View displays both single value attributes (for example the ‘name’ attribute), and vector attributes (for example the ‘OutputFeatures’ and the ‘Product-Materials’ attribute).

Several cells, either single value or vector values, in one column can be set to the same value by:

- select one Cell having the value you want to copy, 10 in the example screenshot.
- select a corner of that Cell such that the cursor shape changes to a small rectangular.
- drag that corner of the Cell over the other Cells



Several single value cells, not necessarily in one column, can be set to the same value by:

- Select several cells
- Ctrl-click in one of the fields that you want to change
- Edit the value of the last selected cell and press Enter; the values of all selected cells are changed to the new value.

Activity: Editing Cells

- *Activate all Rules.*
- *In the Table View click on the ‘name’ column to order the data.*
- *Change the priority of all the TAPPING operations to 10 using the two described methods.*

Rule View

The Rule View shows the data of a selected Rule. This is shown in a set of tabs, which are described later, and six Rule Definition Components. These are:

Name. Contains the name of the Rule.

OperationClass (oper.). Contains the class of the operation that is to be generated based on this Rule. Pressing the F8 key in the field will pop-up a scrollable list with operation classes. Select one item from the list, type the value manually, or use a copy/paste sequence. The operation class and its parameters determine the motions of the tool during the operation.

> **Note** Display Names (also known as Nice Names) for operation classes are not supported by the Machining Knowledge Editor. The class name will be shown. This also applies for the feature classes and the tool classes.

The display color will change when a valid OperationClass is defined. This also applies for the feature classes and the tool classes. The classes and related information are defined in the customization belonging to the RuleLibrary. This is explained in “Customization view” on page 37.

There is a special class DummyOperation that can be use to change one feature type into another feature type. Rules of this class have no tool class.

Priority. Contains a numerical value which determines the relative priority of the Rule, compared with other Rules producing the same More Worked Feature. The priority defines which Rule will be tried first, when there are more Rules that can machine the same feature. Generally, we give Rules that correspond to ‘cheap’ operations (like drill operations) a higher priority than Rules that correspond to ‘expensive’ operations (like reaming operations). This way we can be sure the system will try cheap Rules first and when these cannot be applied, try the next more expensive Rule etcetera.

OutputFeatures (mwf.). Contains the class of the Output Feature that is produced by the operation when the Rule is applied. This corresponds to the shape of the feature at the end of the operation. Pressing the F8 key in the field will pop-up a scrollable list with feature classes. Select one item from the list, type the value manually, or use a copy/paste sequence. When the Rule produces multiple features, they are separated by commas.

InputFeatures (lwf.). Contains the class of the Input Feature that is expected by the operation before the Rule is applied. This corresponds to the shape of the feature at the beginning of the operation. Pressing the F8 key in the field will pop-up a scrollable list with feature classes. Select one item from the list, type the value manually, or use a copy/paste sequence. When the Rule consumes multiple features, they are separated by commas.

Resources (tool.). Contains the class of the tool that is used by the operation when the Rule is applied. Pressing the F8 key in the field will pop-up a scrollable list with tool classes. Select one item from the list, type the value manually, or use a copy/paste sequence.

When you locate the mouse pointer in any of the above fields you get a short help text.

In addition to the six Rule Definition Components, there are seven Tabs containing more detailed aspects of the Rule Definition. These are described in the following sections.

Conditions Tab

This tab shows the conditions pertaining to the Rule. Conditions are used to describe specifics of the Rule, like when a Rule can be applied, which specific tool should be selected, operation parameters etc. All conditions are interpreted as logical expressions and must be true. If one or more conditions is false, the Rule will be rejected.

The conditions are described in an easy, Visual Basic style, text based editing environment. This allows for free text editing (manual typing, delete, insert, copy, paste, etc.) with support for standard windows shortcuts:

TABLE 1. Shortcuts for condition editing

Ctrl + A	Select all text.
Ctrl + X	Cut selected item.
Ctrl + C	Copy selected item.
Ctrl + V	Paste
Home	Goes to beginning of current line.
Ctrl + Home	Goes to beginning of document
End	Goes to end of current line.
Ctrl + End	Goes to end of document.
Shift + Home	Highlights from current position to beginning of line
Shift + End	Highlights from current position to end of line
Ctrl + Left arrow	Moves one word to the left at a time
Ctrl + Right arrow	Moves one word to the right at a time.

For more details please refer to “Conditions concept” on page 22.

Constants Tab

This tab shows the constants that can be used within conditions to improve consistency and readability. Constants can be used instead of hard-coded values in expressions. See “Constants concept” on page 30.

Materials Tab

This tab shows the list of materials that are defined as part materials within NX. A Rule can be made valid for selected materials. By checking the ‘All Materials’ check box, a Rule will be valid for any material, also for materials that will be added in the future.

Machines Tab

This tab shows the list of machines that are defined within NX. A Rule can be made valid for selected machines. By checking the ‘All Machines’ check box, a Rule will be valid for any machine, also for those that will be added in the future.

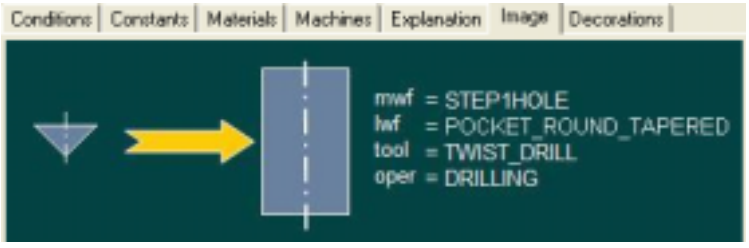
Explanations Tab

Contains a free textual explanation of the Rule.

Image Tab

You can add a picture to the Rule. The image can have as format .jpeg or .jpg. Use the command ***Add Image...*** that is available in the popup menu of the Image tab.

The picture appears in the Image Tab.



Add-on Tab

This Tab is new in NX 7.5. It enables the creation and editing of what we have called “Add-ons”. Depending on the Rule’s Operation Class, different types of Rule Add-ons can be defined.

Examples of Add-on Object Types for a **Drill** Rule are:

- Cycle - enables definition of the cycle type and its specific parameters
- Start of Path Events - like Approach Markers, opstop and many more
- End of Path Events -
- Knowledge Fusion (KF) - all parameters of the Rule’s Operation Class can be defined via KF if desired.

Example of Add-on-Types for a **Groove_OD** Rule for turning is:

- Geometry Parent, class Containment (is mandatory)

Example of Add-on-Types for an **Internal_Trim** Rule for Wire EDM is:

- Geometry Parent, class Wedm_Geom (is mandatory)

Examples of Add-on-Types for a **Cavity_Mill** Rule are:

- Geometry Parent, class Mill_Area
- CutLevel, class ZlevelBase
- NcmLocalEngage, class EngageRetract_Default

Examples of Add-on-Types for a **Thread_Milling** Rule are:

- Geometry Parent, class Hole_Boss (is mandatory)

Conditions concept

A condition is a logical expression which can evaluate to TRUE or FALSE.

There are four distinct categories of Conditions:

- ‘Application Criteria” see page 22.
- ‘Tool Attributes” see page 24.
- ‘Less Worked Feature Attributes” see page 26.
- ‘Operation Attributes” see page 27.

Although NX does not “know” these categories, it is good to keep them in mind and organize them into these four categories.

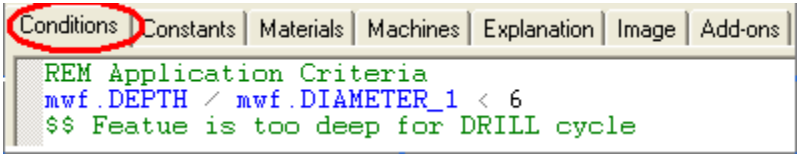
Application Criteria

Application criteria state when a Rule may be applied. In general, application criteria are related to the size and quality of the More Worked Feature. Examples are:

- The roughnesses that can be achieved: e.g. drilling is only allowed until a certain roughness, otherwise reaming/boring should be used.
- The size tolerances that can be achieved: e.g. drilling is only allowed until a certain diameter tolerance, otherwise boring or reaming should be used.
- The shape of the feature: e.g. drilling with a straight drill will finish the shape only when there are no chamfers.
- The size of the feature: e.g. drilling is only allowed when the feature is not too deep or too large, otherwise another operation type should be used.

A Rule can only be applied to a feature when all its *application criteria* are TRUE. When the expression evaluates to FALSE, evidently the Rule may NOT be applied.

An example of an application criterion for using a twist drill is
:



```
REM Application Criteria
mwf.DEPTH / mwf.DIAMETER_1 < 6
$$ Feature is too deep for DRILL cycle
```

Each line in this example has a different function. The first line starting with 'REM' is considered as remark and does not have any effect on the behavior of the program. The programmer can use remarks to include short explanations or observations. It is advised to add remarks so others can understand the behavior of the Rule and the reason for the conditions. There can be several lines of remarks following each other, but each line needs to start with 'REM '.

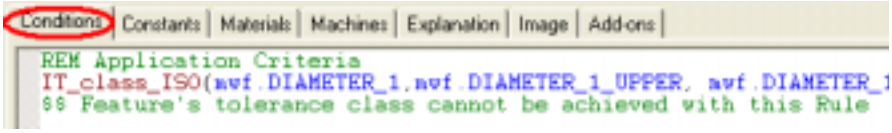
The second line
`mwf.DEPTH / mwf.DIAMETER_1 < 6`
is the actual condition. In this case defining that the DEPTH attribute of the mwf (More Worked Feature) should be smaller than 6 times the DIAMETER_1 of the mwf. When applied to a manufacturing feature, with its actual parameter values for DEPTH and DIAMETER, this condition is false, the Rule will be rejected and the system will continue evaluating the next Rule.

The third line starting with $$ defines the message that will be output when the condition is false and thus the rule is rejected. This message is part of the messages that users will see if they want to analyze why a certain solution was found. As a consequence, these messages should be clear enough for the users to be able to understand why a Rule was not used. When the Rule is denied because of this condition, the text prefixed with "$$" will be printed in the Information window (if displayed) and to the MSAnalysis.lis file on the $TEMP directory.

In the above examples, DEPTH and DIAMETER_1 are attributes of STEP1HOLE. In order to obtain a complete list of attributes you can use in expressions, type "mwf" followed by a dot "." and select one from the list.

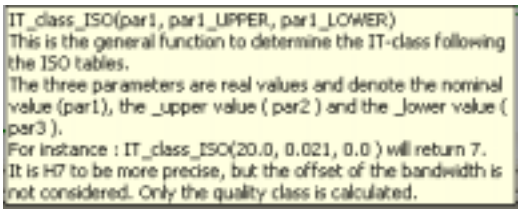
When the editor recognizes valid entries, it will display them in a different color. These colors are defined in the Preferences.xml that can be found in the MACH\machining_knowledge_editor directory.

Another example of an application criterion is:



```
IT_class_ISO(mwf.DIAMETER_1, mwf.DIAMETER_1_UPPER,
mwf.DIAMETER_1_LOWER) >= 11
$$ Feature's tolerance class cannot be achieved with this Rule
```

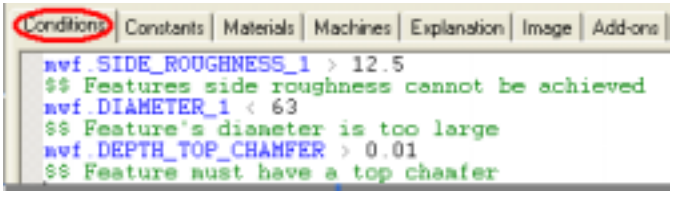
In the IT-class system for tolerancing dimensions, a low value denotes a tighter tolerance than a high value. So if the IT class of an actual feature is 7, it is too tightly toleranced to be drilled. Then this application criterion will evaluate to FALSE and the Rule will not be applied.



In the above example, IT_Class_ISO( ) is a function delivering an integer number as result. When placing the mouse over the function text, a tool tip will appear explaining the function, see left.

In order to obtain a complete list of functions you can use in expressions, press F8 and select one from the list.

Some other examples of rather common Application Criteria are:




```
mwf.SIDE_ROUGHNESS_1 > 12.5
$$ Features side roughness cannot be achieved
mwf.DIAMETER_1 < 63
$$ Feature's diameter is too large
mwf.DEPTH_TOP_CHAMFER > 0.01
$$ Feature must have a top chamfer
```

Tool Attributes

The Tool Query defines the size of the tool that will be used for the operation. Examples are:

- The diameter of the tool should be corresponding to the diameter of the mwf: e.g. the drill diameter should be smaller than the maximum toleranced size of the mwf diameter but should be larger than the minimum toleranced size.
- The tool should be long enough: e.g. the length of the drill should be larger than the depth of the hole.

A Rule will have one or more expressions to specify tool attributes like:

```
tool.Diameter >= mwf.DIAMETER_1 + mwf.DIAMETER_1_LOWER
tool.Diameter <= mwf.DIAMETER_1 + 0.5*(mwf.DIAMETER_1_UPPER +
mwf.DIAMETER_1_LOWER)
tool.FluteLength > mwf.DEPTH
```

The expressions above were entered using copy and paste, see screenshot below.

In the above example, Diameter, and FluteLength are attributes of TWIST_DRILL. In order to obtain a complete list of attributes you can use in expressions, type “tool” followed by a dot “.” and select one from the list.

The text printed, when one of these conditions evaluates to FALSE, says “Conflict in one of the tool size specifications”. That would happen, for instance, if we had mistakenly entered something conflicting like:

```
tool.POINT_ANGLE = 120 AND tool.POINT_ANGLE = 118
```

Note The operator AND is equivalent to putting the expressions on separate lines.

A Rule should not be rejected based on conditions defining tool Attributes. When the cutting tool of the class is not available, the Rule will not be applied and NX will try the next Rule. In that case, the $$text will **not** be displayed.

NX will try to use the tool with the biggest Cutter Diameter that is still smaller than the maximum as specified by the expressions. As a consequence, the lower limit only needs to be defined when it is essential for a correct tool selection, like for drill type operations. For milling operations a lower limit is most of the times not necessary. For this to function, NX must know which attribute holds the size of the Cutter Diameter. This is a trivial issue, but since the tool classification is free, it is necessary to appoint one of the tool’s attributes to be the Cutter Diameter. This is done in the Customization View with command ***Set as Cutter Diameter***...

Tool Sort Order

Generally, more than one tool of the desired class satisfies the tool query. It is important to avoid choosing a too long and therefore unstable tool. Tools are ordered by their length. For this to function, NX must know which attribute holds the size of the Cutter Length. This is a trivial issue, but since the tool classification is free, it is necessary to appoint one of the tool's attributes to be the Length. This is done in the Customization View with command ***Set as Cutter Length***.

Less Worked Feature Attributes

When defining Less Worked Feature attribute values we define the in-process geometry. The screenshot above shows the drill point that exists before drilling the hole.

If no Less Worked Feature is required, the class BLANK is used. This means the operation does not need a defined starting geometry.

> **Note** In general, there should be at least one operation with BLANK as LWF for every MWF feature type. Otherwise there will be no complete solution found for a specific feature type.

In order to calculate the dimensions we need to be familiar with the feature classes and their attributes.

To specify the dimensions of the drill point (class POCKET_ROUND_TAPERED) we need following expressions:

```
lwf.DIAMETER_1 = mwf.DIAMETER_1 * 0.5
lwf.DIAMETER_2 = 0
lwf.DEPTH = lwf.DIAMETER_1/2
```

In the above example, DIAMETER_1, DIAMETER_2 and DEPTH are attributes of Less Worked Feature POCKET_ROUND_TAPERED. In order to obtain a

complete list of attributes you can use in expressions, type “lwf” followed by a dot “.” and select one from the list.

When there is no expression for the LWF attribute, NX copies the corresponding value from the MWF. This works only if the attribute names match exactly. So, if the DEPTH of the feature does not change, you do **not** need to enter the expression:

```
lwf.DEPTH = mwf.DEPTH
```

Also, you do **not** need to enter expressions like:

```
lwf.X_POSITION = mwf.X_POSITION
lwf.Y_POSITION = mwf.Y_POSITION
lwf.Z_POSITION = mwf.Z_POSITION
```

When the LWF is of class BLANK, we do not care about the LWF class and hence do not need to specify any LWF attributes.

For readability, it is good practice to have lwf attributes on the left side of the equation.

Operation Attributes

The last category of conditions specifies attributes of the Rule’s operation. Examples are:

```
REM Operation Attributes

oper.Control_Point_Type ="Tool_Shoulder"
oper.Maximum_Depth_Distance = mwf.DEPTH + constant.Thru
oper.Through_Clearance  = constant.Thru_Hole_Clearance
```

- The operation should go deep enough
- What is the control point on the tool

Above a screenshot with some examples of assignments to operation attributes. In order to obtain a complete list of attributes you can use in expressions, type “oper” followed by a dot “.” and select one from the list.

Expression syntax | Functions

This section gives a short overview of the expression syntax including the operators and functions that can be used within the expressions.

Table 2—Operators

Operator	Explanation
=	equal
<	smaller than
<=	smaller than or equal
>	greater than
>=	greater than or equal
< >	not equal
+	addition
-	subtraction
*	multiplication
/	division
^	power
AND	logical
OR	logical
NOT	logical
(	left parenthesis
)	right parenthesis
IF THEN ELSE	conditional expression

Note a - b + c is evaluated as a - (b + c). It is advised to use brackets for clarity and correct evaluation.

Functions

Press F8 while in the Conditions Tab to obtain a list of functions:

TABLE 3. Functions to be used in expressions

output	function(argument)	explanation
[degrees]	ACOS (number)	returns the arc cosine.
[degrees]	ASIN (number)	returns the arc sine.
[degrees]	ATAN (number)	returns the arc tangent.
[number]	COS (degrees)	returns the cosine.
[number]	SIN (degrees)	returns the sine.
[number]	TAN (degrees)	returns the tangent.
[integer]	CEIL (number)	returns the smallest integer >= input value.
[integer]	CEIL_RANGE (number)	The input value can be a range, bound single or double sided. IF (a<=number<=b) THEN CEIL((a+b)/2) Otherwise same as CEIL(number)
[integer]	FLOOR (number)	returns the largest integer <= input value.
[integer]	FLOOR_RANGE (number)	Similar to CEIL_RANGE
[boolean]	is_defined(arg)	returns TRUE if arg is defined; FALSE otherwise. returns FALSE if arg does not exist; arg can be of type real, integer or string;
[integer]	IT_class_ISO(number , number,number)	Example: `IT_class(20,0.021,0) = 7`
[double]	roughness_value(string)	returns the roughness value of an attribute
[boolean]	on_same_axis(mwf1 , mwf2)	returns true if the two features are on the same axis
[double]	distance(mwf1 , mwf2)	returns the distance between two features

Constants concept

Constants are objects with a name, a type and a value. We use them in conditions instead of ‘hard-coded’ values, for instance:

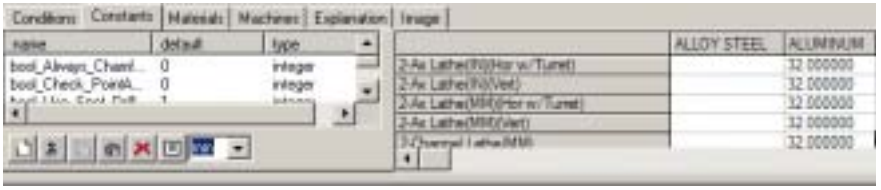
```
oper.Depth = mwf.DEPTH + constant.DEPTH_EXTEND
```

instead of

```
oper.Depth = mwf.DEPTH + 2
```

Using constants instead of hard-coded values eases the readability and maintainability of expressions.

Another advantage of using constants is the possibility to vary the values for any combination of machine / material. If there is no value specified then the default is used.



Above screenshot shows a **constant**.Pre_Drill_Limit with a different value for material ALUMINIUM.

A constant can be of datatype ‘double’, ‘integer’, or ‘string’.

Different constant values can be defined for metric and inch part files. Based on the units used in the part file, NX will automatically select the correct values of the constants.

- If you define a constant value in metric (mm) only then NX will calculate the english (inch) value by division by 25.4.
- If you define a constant value in english (inch) only then NX will calculate the metric (mm) value by multiplication with 25.4.
- If you define a value for both english (inch) and metric (mm) unit systems then -however- these values will be used by NX.

Scope of a Constant

Preferably we declare a Constant ‘globally’, that is on the RuleLibrary level, with its default value and if applicable its values per machine / material combination.

We use the Constant locally in a Rule’s Conditions and when the value should be different from the global value we can define the Constant locally at Rule level.

Constant Command Icons

- Opens a dialog window to declare a new Constant and its default value.
- Cuts selected Constant and saves it on the clipboard.
- Copies selected Constant and saves it on the clipboard.
- Paste from the Clipboard
- Delete selected Constant
- Edit selected Constant (alternative for double click)

These commands are also available in the pop-up menu of the list of constants.

Using Constants in Expressions

In order to obtain a complete list of Constants you can use in expressions, type “constant” followed by a dot “.” and select one from the list.

It is possible to use a constant even before it is declared. However, the ***Check Validity*** command will stumble over this and you can declare afterwards.

Finding Rules based on criteria

There are several ways of finding and searching within the Machining Knowledge Editor. These options are described in the following section.

Find in all Rule Conditions

Use command ***Find in all Rule Conditions*** in the condition Tab or on the RuleLibrary, to activate (so they're displayed in Table View) all Rules having the search string.

Find/Replace

To search for any text in the conditions and replace that text. When at the end of the Rule's condition, the command asks if it should continue to search in the next Rule unless the 'Wrap around search' is checked active.

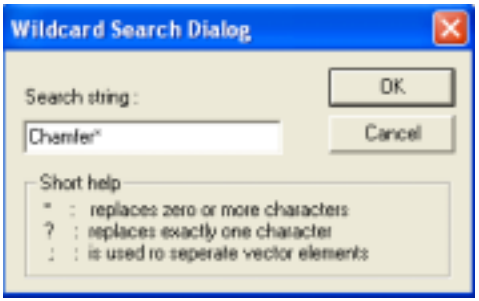
Using TableView autofilter

- Use Table View to filter the RuleLibrary on any criteria, like Rules for machining a feature of class STEP1HOLE.

InputFeatures		OutputFeatures	
(All)		STEP1HOLE	
1	BLANK	1	STEP1HOLE
1	STEP1HOLE	1	STEP1HOLE
1	POCKET_R	1	STEP1HOLE

Using TableView Filter Column

- You can search a selected column using the Wildcard search. For instance you can find all activated Rules who's name starts with Chamfer:



Activity: Searching the Rules

- *Use the table view to find all Rules with the TAPPING OperationClass.*
- *Find all Rules that use the condition 'tool.Diameter < mwf.DIAMETER_1'*

Testing the Rules

Check Validity

Once a Rule is complete, the Rule can be checked by compiling the Rule or some of the conditions. Compilation of the Rule conditions can be done on different levels. Typically this is done to test conditions on validity ‘on the fly’.

- Select one or more lines in the conditions Tab and execute ***Check Validity (selection)*** from the pop-up menu.
- Select the Rule node in the RuleLibrary and execute ***Check Validity*** from the pop-up menu.

If the compiler discovers mistakes these are indicated by bookmarks. Locate these bookmarks and correct the mistakes with the help of the compiler’s error message.

Activity: Check Validity

- *Select a Rule and check the validity of some of the conditions.*
- *Select several Rules and check the validity of the Rules.*

Building

Before the Rule Library can be used, the RuleLibrary has to be compiled. The command ***Build*** on the RuleLibrary will create a dll (dynamically linked library) which is used by command ***Create Feature Process***.

Normally the build ends successfully and there will be a dll created in the same directory as the library xml file. This dll contains the compiled and linked Rules.

If the build finishes with an error, the MachiningKnowledgeEditor.log contains all error messages. This file is saved in the $TEMP directory.

Activity: Building the RuleLibrary

- *Build the RuleLibrary*

Testing in NX with *Create Feature Process*

Once a library has been built, the ***Create Feature Process*** command in NX can be used to test the Rules. This can be done by selecting features in the NX Manufacturing Feature Navigator and starting the ***Create Feature Process*** from the popup menu.

Reading the Information window and the MSAnalysis.lis

After each execution of ***Create Feature Process*** there will be a report in the Information window and in the MSAnalysis.lis file on the $TEMP directory.

At the beginning of the file, the used dll is mentioned, like:

```
Knowledge library: C:\Program Files\UGS\NX
7.5\mach\resource\machining_knowledge\machining_knowledge.dll
```

When a Rule is denied because of a condition, the text after that condition prefixed with “$$” will be printed.

Activity: Testing the RuleLibrary

- *Start NX*
- *Load a test model*
- *Switch to Manufacturing.*
- *In the Feature Navigator, recognize features with the **Find Features** command*
- *Start the **Create Feature Process** command on a feature.*

Rules for multiple More Worked Features

It is allowed, for feature mapping rules only, to have multiple mwf's separated by commas.

Attribute Rule.type [*direct*,*composed*,*indirect*] determines the applicability of the Rule.

- *direct*

  mwf **OR** mwf **OR** mwf

  Attributes in conditions are prefixed with "mwf". Since only attributes can be used that are in all mwf classes there can be no confusion.

- *composed*

  mwf **AND** mwf **AND** mwf

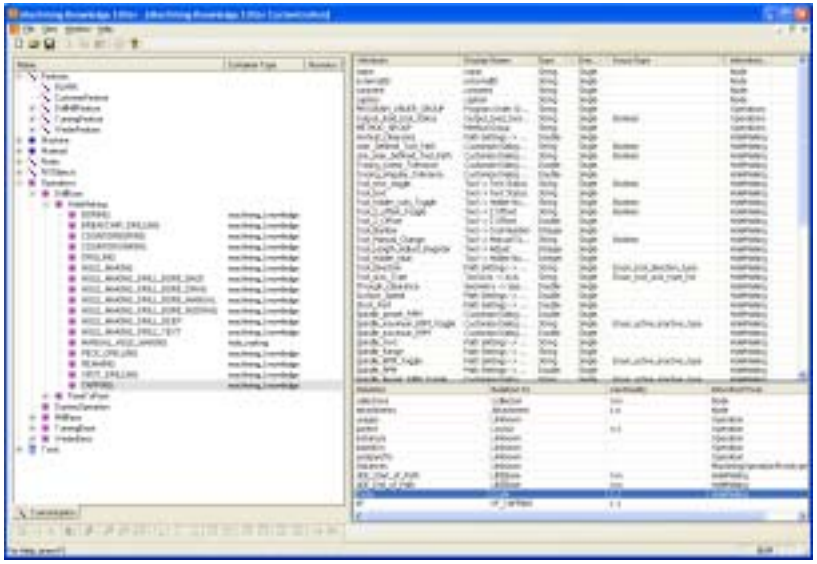
  Attributes in conditions must be pre-fixed with mwf_1, mwf_2 etcetera.
  This is not yet supported in for operation selection rules.
  It is supported in NX6 for feature mapping rules.

- *indirect*

  Rule will be filtered out in any case during ***Compute*** **>** ***Operations*** command.

Customization view

The customization defines which classes are available and which attributes can be used. Once a RuleLibrary is loaded, the customization can be viewed by selecting ***View > Customization*** from the main menu.



This opens the customization view consisting of a tree view on the left listing the classes. On the right hand side you find an attribute view listing all attributes of the selected class, and a relation which shows the relations of the selected class with other objects in the customization.

In the relation view, for instance, we find that all operations under HoleMaking have a 1:1 relation with the Cycle object which you can find back under the NXObjects node.

The tree views contains:

- Features, Operations and Tools classes. For the Operation classes, field Container Type holds the name of the operation template where the class is defined.
- Materials and Machines from the NX libraries.

The attributes view shows, for each attribute in the selected class:

- Attribute: the name of the attribute as it is used in the MKE.
- Display Name: the path to the parameter in the NX Operation Dialogue. It helps to find the purpose of the operation parameters.
- Type: the type of the attribute
- Enum Type: the type of enumerated defining the allowed values of the attribute.

Activity: Customization view

- *Open the Customization view and explore the customization.*

Making changes to the customization

Changing the customization is described in “Customization Files” on page 132.

CHAPTER 2

How Operation Selection works

This chapter deals with the Operation Selection that was introduced in NX6. It gives some background on how things work. NX CAM programmers do not necessarily have to be aware of this.

What is Operation Selection?

Rule–based operation selection is a proven technology seamlessly integrated into NX 6 that helps you automatically create operations such as milling, drilling and tapping from a generic template. Since NX 7.5 it is also applicable to Turning and Wire EDM operations.

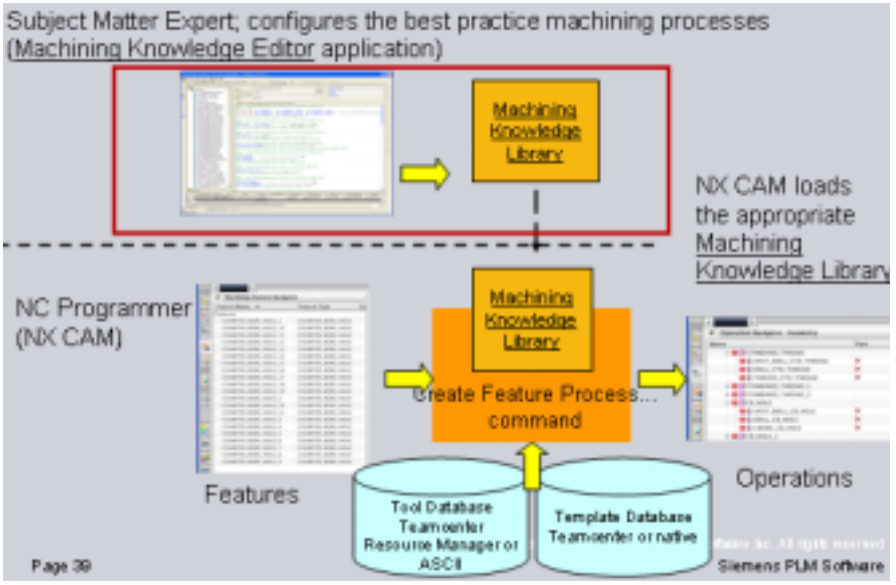
It lets you select features such as holes, slots, and pockets from any source, including features that are user defined, identified, recognized or tagged.

It applies best practice machining rules on the features while taking into account any defined PMI.

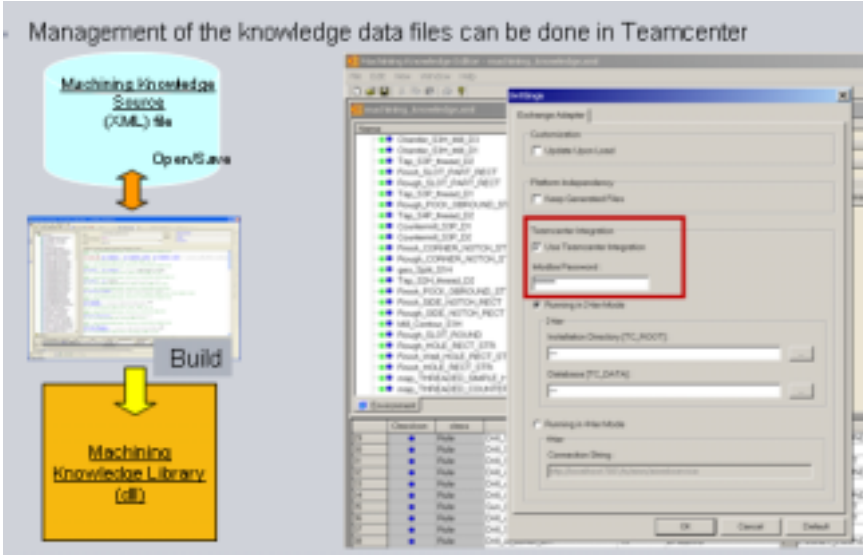
Benefits of feature based Operation Selection are:

- Standardize on best practice machining knowledge. The software finds the best solution for a machining task within a company's environment.
- Save time with process automation.

Machining Knowledge Editor and NX



There is a clear separation between the tool of the Subject Matter Expert, the Machining Knowledge Editor (above the dashed line) and the NX Programmer.

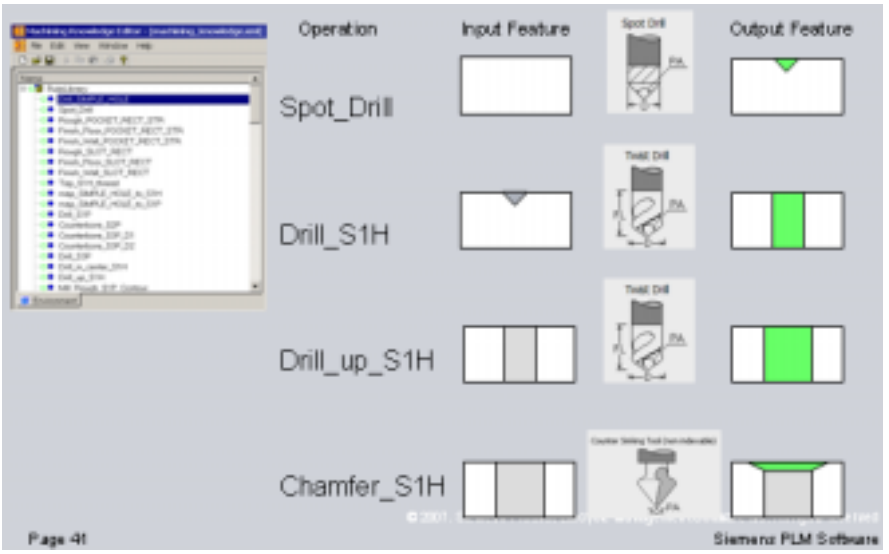


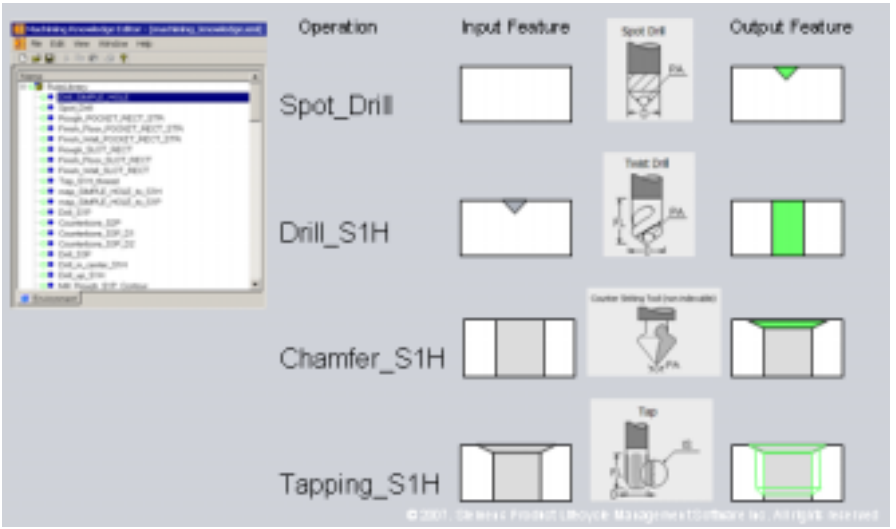
The Machining Knowledge Library dll is created using the MKE. NX can be configured to use this dll. For details on the configuration, See “Customization Files” on page 132.

Since, in NX 7.5, the dll is still created using a c-compiler, it is necessary to have a c-compiler installed.

Examples of operation sequences

STEP1HOLE and STEP1HOLE_THREAD in 4 operations.



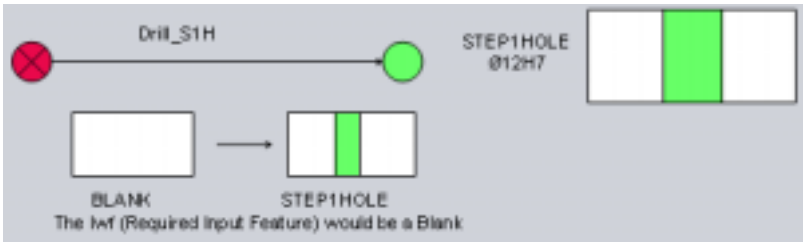


Observe that **Spot_Drill** and **Drill_S1H** and **Chamfer_S1H** are three elemental operations that appear in the solution for both features. This is an essential concept:

elementary rules are defined only once and are applied whenever appropriate. This sets aside the NX FBM solution to other systems or custom made solutions which find solutions by definining the complete set of operations for every case that needs to be automated. In NX FBM, a modification to a Rule will have effect in all instances where the rule will be applied.

Finding a solution using elementary rules

In this example we will follow the reasoning process that is followed when searching for a solution for a feature of type STEP1HOLE with a diameter 12H7.

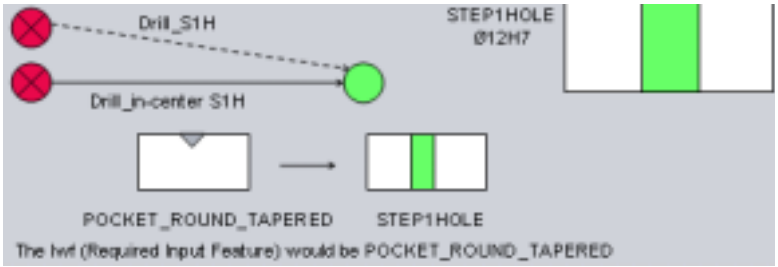


name	Priority	OperationClass	Input		Output		Resources	
(All)	(All)	(All)	(All)		STEP1HOLE		(All)	
Ream_S1H	1	REAMING	1	STEP1HOLE	1	STEP1HOLE	1	CHUCKING_REAMER
Bore_S1H	2.1	BORING	1	STEP1HOLE	1	STEP1HOLE	1	BORE
Chamfer_S1H_Mill	3	HOLE_MILL	1	STEP1HOLE	1	STEP1HOLE	1	COUNTER_SINKING
Chamfer_S1H_Drill	3.1	COUNTERSINKING	1	STEP1HOLE	1	STEP1HOLE	1	COUNTER_SINKING
Drill_up_S1H	4.2	DRILLING	1	STEP1HOLE	1	STEP1HOLE	1	TWIST_DRILL
Drill_in_center_chamfer_S1H	4.3	DRILLING	1	POCKET_ROUND_TAPERED	1	STEP1HOLE	1	TWIST_DRILL
Drill_in_center_S1H	4.4	DRILLING	1	POCKET_ROUND_TAPERED	1	STEP1HOLE	1	TWIST_DRILL
Drill_S1H	4.5	DRILLING	1	BLANK	1	STEP1HOLE	1	TWIST_DRILL

First the candidate rules will be selected. These are the rules that produce a feature of type STEP1HOLE. By filtering the Table View on STEP1HOLE we can create that list in the MKE. NX Operation Selection will create the same list internally.

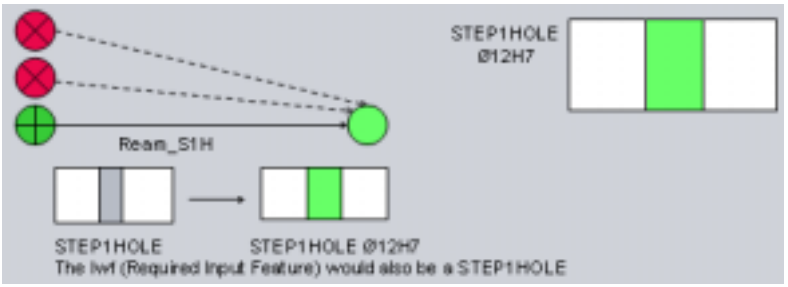
Operation Selection will first try the candiate Rule with highest priority. We have given the rules priorities such that the least expensive has the highest priority. In the example this means that Drill_S1H is tried first. It cannot be applied because a diameter tolerance of H7 is not achievable when using a twist drill.

That is one of the Rule's application criteria. This road ends which is denoted by: 



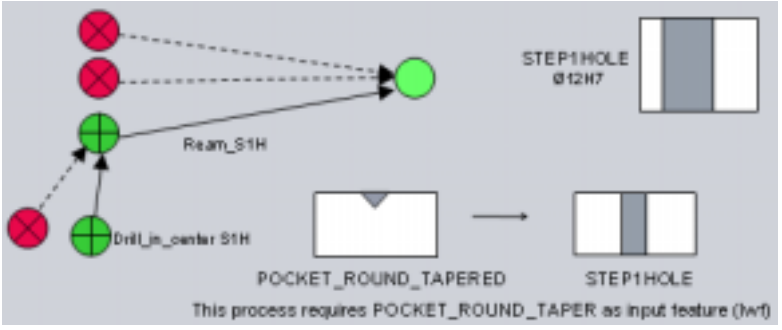
name	Priority	OperationClass	Input		Output		Ressources	
(All)	(All)	(All)	(All)		STEP1HOLE		(All)	
Ream_S1H	1	REAMING	1	STEP1HOLE	1	STEP1HOLE	1	CHUCKING_REAMER
Bore_S1H	2.1	BORING	1	STEP1HOLE	1	STEP1HOLE	1	BORE
Chamfer_S1H_Mill	3	HOLE_MILL	1	STEP1HOLE	1	STEP1HOLE	1	COUNTER_SINKING
Chamfer_S1H_Drill	3.1	COUNTERSINKING	1	STEP1HOLE	1	STEP1HOLE	1	COUNTER_SINKING
Drill_up_S1H	4.2	DRILLING	1	STEP1HOLE	1	STEP1HOLE	1	TWIST_DRILL
Drill_in_center_chamfer_S1H	4.3	DRILLING	1	POCKET_ROUND_TAPERED	1	STEP1HOLE	1	TWIST_DRILL
Drill_in_center_S1H	4.4	DRILLING	1	POCKET_ROUND_TAPERED	1	STEP1HOLE	1	TWIST_DRILL
Drill_S1H	4.5	DRILLING	1	BLANK	1	STEP1HOLE	1	TWIST_DRILL

Then the next expensive Rule, Drill_in_center_S1H is tried. Also this Rule will fail for the same reason as Drill_S1H : a tolerance of H7 cannot be achieved with a twist drill.



name	Priority	OperationClass	Input	Output	Resources
[All]	[All]	[All]	[All]	STEP1HOLE	[All]
Ream_S1H	1	REAMING	1 STEP1HOLE	1 STEP1HOLE	1 CHUCKING_REAMER
Bore_S1H	2	BORING	1 STEP1HOLE	1 STEP1HOLE	1 BORE
Chamfer_S1H_Mill	3	HOLE_MILL	1 STEP1HOLE	1 STEP1HOLE	1 COUNTER_SINKING
Chamfer_S1H_Drill	3.1	COUNTERSINKING	1 STEP1HOLE	1 STEP1HOLE	1 COUNTER_SINKING
Drill_up_S1H	4.2	DRILLING	1 STEP1HOLE	1 STEP1HOLE	1 TWIST_DRILL
Drill_in_center_chamfer_S1H	4.3	DRILLING	1 POCKET_ROUND_TAPERED	1 STEP1HOLE	1 TWIST_DRILL
Drill_in_center_S1H	4.4	DRILLING	1 POCKET_ROUND_TAPERED	1 STEP1HOLE	1 TWIST_DRILL
Drill_S1H	4.5	DRILLING	1 BLANK	1 STEP1HOLE	1 TWIST_DRILL

We skip a few candiate Rules that are rejected and arrive at Ream_S1H. For this Rule all conditions are true, and it is really applied which is denoted by a new node: 



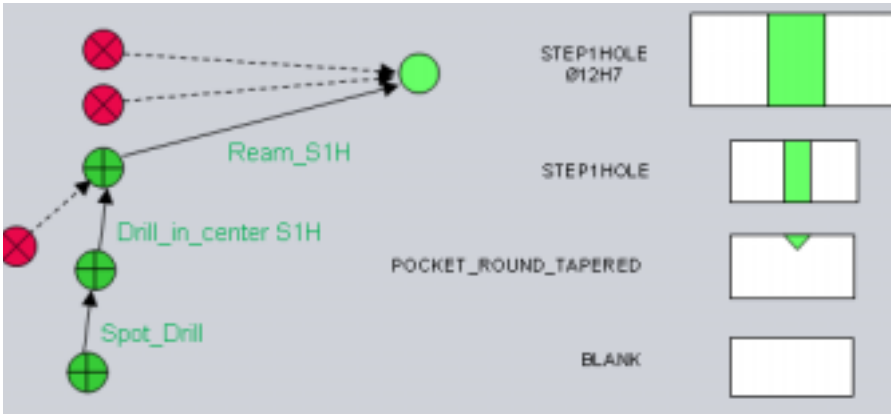
name	Priority	OperationClass	Input		Output		Resources	
(All)	(All)	(All)	(All)		STEP1HOLE		(All)	
Ream_S1H	1	REAMING	1	STEP1HOLE	1	STEP1HOLE	1	CHUCKING_REAMER
Bore_S1H	2.1	BORING	1	STEP1HOLE	1	STEP1HOLE	1	BORE
Chamfer_S1H_Mill	3	HOLE_MILL	1	STEP1HOLE	1	STEP1HOLE	1	COUNTER_SINKING
Chamfer_S1H_Drill	3.1	COUNTERSINKING	1	STEP1HOLE	1	STEP1HOLE	1	COUNTER_SINKING
Drill_up_S1H	4.2	DRILLING	1	STEP1HOLE	1	STEP1HOLE	1	TWIST_DRILL
Drill_in_center_chamfer_S1H	4.3	DRILLING	1	POCKET_ROUND_TAPERED	1	STEP1HOLE	1	TWIST_DRILL
Drill_in_center_S1H	4.4	DRILLING	1	POCKET_ROUND_TAPERED	1	STEP1HOLE	1	TWIST_DRILL
Drill_S1H	4.5	DRILLING	1	BLANK	1	STEP1HOLE	1	TWIST_DRILL

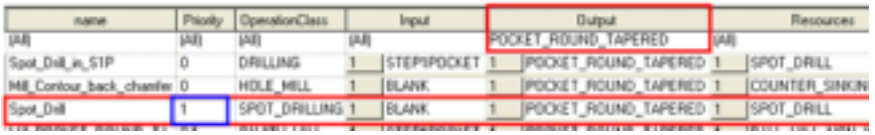
The new node is an in-process feature of, again, type STEP1HOLE. The Ream_S1H conditions will have loosened the tolerance on its diameter.

The in-process STEP1HOLE is the new target for the Operation Selection. And what we have seen before happens again: the candidate list with Rules is tried in order of priority, highest first.

We see that Drill_S1H is rejected because of a general application settings that rejects drilling directly without centering first.

The second Rule is Drill_in_center_S1H. For this Rule all conditions are true, and it is really applied which is denoted by a new node:





name	Priority	OperationClass	Input		Output		Resources	
(All)	(All)	(All)	(All)		POCKET_ROUND_TAPERED		(All)	
Spot_Drill_in_S1P	0	DRILLING	1	STEP1POCKET	1	POCKET_ROUND_TAPERED	1	SPOT_DRILL
Mill_Contour_back_chamfer	0	HOLE_MILL	1	BLANK	1	POCKET_ROUND_TAPERED	1	COUNTER_SINKIN
Spot_Drill	1	SPOT_DRILLING	1	BLANK	1	POCKET_ROUND_TAPERED	1	SPOT_DRILL

The new node is an in-process feature of type POCKET_ROUND_TAPERED.

The in-process POCKET_ROUND_TAPERED is the new target for the Operation Selection. And what we have seen before happens again: the candidate list with Rules is tried in order of priority, highest first.

The applied Rule is Spot_Drill. For this Rule all conditions are true, and it is really applied which is denoted by a new node:

The input feature of Spot_Drill is of type BLANK.

The Operation Selection has successfully found a suitable set of rules to completely machine this feature.

The resulting process in this simplified example is : Spot_Drill => Drill => Ream.

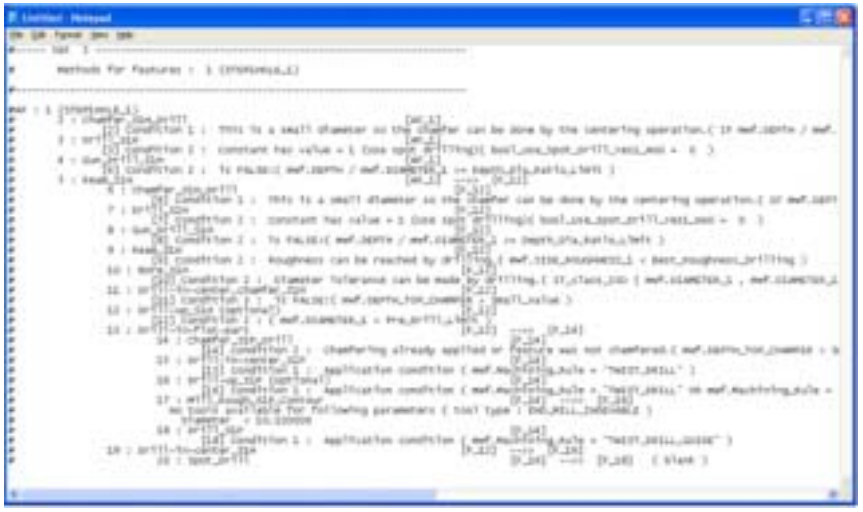
NX will now create the operations in the Operation Navigator.

As you have surely noticed, the result is found in the reversed order, reasoning backward from the final feature to the blank feature. Please take a few moments to consider the human approach in finding such a solution. Do we solve this sort of problems much different ?

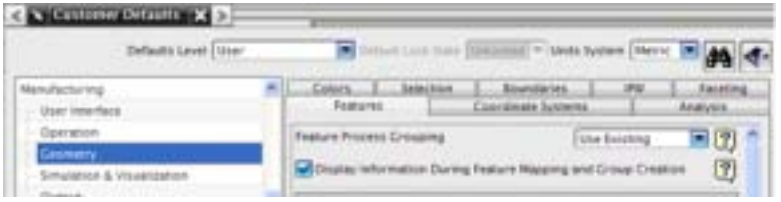
Information Window

On your %temp% directory you will find a file MSAnalysis.lis. It has the analysis of the reasoning process including candidate Rules that were rejected.

For a full understanding of how the solution came about, this file is essential.



With File > Utilities > Customer Defaults...you can activate the check box “Display Information...etc” to have the file popping up after each **Create Feature Process...**



CHAPTER 3

Machining Knowledge Editor Exercises Part 1: Hole Making

Before proceeding, please make sure you have completed reading “Machining Knowledge Editor Concepts”.

In this chapter you find exercises to create a number of example MachiningRules:

- Drill
- Tap
- Ream
- Drill-up
- Thread Mill (to be done)

After completing these exercises you are well equiped to ‘translate’ your company’s best manufacturing practices into MachiningRules that can be used by NX.

The exercises provide detailed instructions. It should be possible to follow this training off-line, self-paced.

Each major subject has a time estimation to complete. In total these add up to almost 12 hours.

Creating a new RuleLibrary file

10 minutes

The Rule Library file is an xml source file containing

- MachiningRules
- FeatureRecognitionRules
- MappingRules

We will focus on MachiningRules in this part of the Tutorial.

There are two options to create a new RuleLibrary file:

1. Copy an existing RuleLibrary file. Advised when your new MachiningRuleLibrary will have a lot in common with the old MachiningRuleLibrary. This is for example when you are going to add extra MachiningRules to an existing MachiningRuleLibrary.
2. Create a new RuleLibrary file from scratch.

Copying an existing RuleLibrary file

This can be done in the following steps:

- Go to the UGII_CAM_MACHINING_KNOWLEDGE_DIR configuration directory.
- Copy the configuration file that is linked to the existing RuleLibrary.
- Rename the copy.
- Edit the copy of the configuration file by assigning a new library name to the MACHINING_KNOWLEDGE_LIBRARY entry.
- Go to the configuration directory. Depending on the installation this is either the UGII_CAM_CONFIG_DIR directory or the UGII_CAM_CUSTOM_DIR directory.
- Copy the configuration file that is linked to the existing RuleLibrary.
- Rename the copy.
- Edit the copy of the configuration file. Defining the MACHINING_KNOWLEDGE entry to point to the new file that was created in the UGII_CAM_MACHINING_KNOWLEDGE_DIR configuration directory.
- Start the Machining Knowledge Editor.

- In the main menu select ***View*** **>** ***Options***.
- Check the ‘Update Upon Load’ box in the ‘Customization’ box. Close the dialog with the ‘OK’ button.
- Open the existing RuleLibrary file. When the ‘NX Configuration Files’ dialog appears, select the .dat file that you created and press the OK button.
- Save the file with the new library name using the ***File*** **>** ***Save As*** command.

Creating a new RuleLibrary file.

This requires the following steps:

- Go to the UGII_CAM_MACHINING_KNOWLEDGE_DIR configuration directory.
- Copy an existing configuration file.
- Rename the copy.
- Edit the copy of the configuration file. Assign a new library name to the MACHINING_KNOWLEDGE_LIBRARY entry.
- Go to the configuration directory. Depending on the installation this is either the UGII_CAM_CONFIG_DIR directory or the UGII_CAM_CUSTOM_DIR directory.
- Copy the configuration file that pointed to the original configuration file in the UGII_CAM_MACHINING_KNOWLEDGE_DIR.
- Rename the copy.
- Edit the copy of the configuration file by defining as the MACHINING_KNOWLEDGE entry to point to the new file that was created in the UGII_CAM_MACHINING_KNOWLEDGE_DIR configuration directory.
- Start the Machining Knowledge Editor.
- Choose ***File >New***.
- In the NX Configuration Files dialog box, from the Configuration Folders list, select the location for the configuration file:

  - If you use a standard NX configuration, select UGII_CAM_CONFIG_DIR.

  - If you have custom configurations, select UGII_CAM_CUSTOM_DIR.

  - Or you can use Browse to select a configuration file from anywhere on the file system.
- From the Configuration Files list, select the file you copied and renamed.

- Select the unit system and click OK.
- Save the file with a new name using the ***File > Save As*** command.

As we will start from scratch we need to create a new RuleLibrary file.

Activity: Creating a new RuleLibrary file

- *Create a file 'my_feature_machining.dat' on the UGII_CAM_CONFIG_DIR pointing to MyRules.dat.*
- *Create a file 'MyRules.dat' on the* UGII_CAM_MACHINING_KNOWLEDGE_DIR *pointing to MyRules.*
- *Launch the MKE from the Start menu > UGS NX 7.5 > Manufacturing Tools > Machining Knowledge Editor.*
- *Create a new RuleLibrary with File > New.*
- *Use 'my_feature_machining.dat' from the UGII_CAM_CONFIG_DIR.*
- *Save the initial source file as MyRules.xml*

Activity: Setting the Cutter Diameter and Cutter Length

Since we are working on a new source file, we must appoint one of each tool's attributes to be the Cutter Diameter.

- *Open the Customization View by selecting* ***View > Customization*** *from the main menu.*
- *Expand the Tools tree.*
- *Right-click TWIST_DRILL in the tree and select* ***Set as Cutter Diameter ....***
- *Select the attribute 'Diameter' and press the > arrow to move it to the 'Group by' box.*
- *Press OK.*
- *Repeat this for classes TAP , CHUCKING_REAMER and BORE.*
- *Repeat with* ***Set as Cutter Length....***

Defining a first Rule

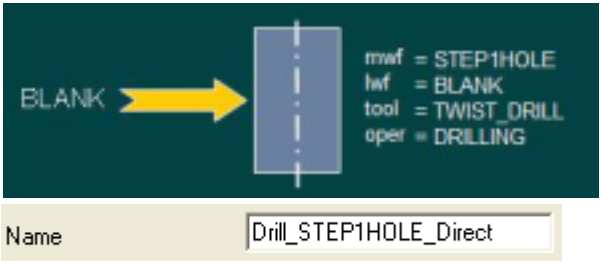
What this example intends to explain

- The basics of creating a single Machiningrule
- How to ***Build*** the MachiningRuleLibrary
- How to *test* the MachiningRule

All conditions for this MachiningRule are presented here. Please keep in mind that these are just examples, as is the MachiningRule itself. Especially conditions of category “Application Criteria” are always customer specific.


60 minutes.

Explanatory picture



We start by defining a MachiningRule which describes how to drill a straight hole in full material using a common drill. When creating a new MachiningRule, we have to define the following items:

- The **More Worked Feature** class, describing the shape of the end geometry.
- The **Less Worked Feature** class, describing the shape of the starting geometry.
- The cutting **tool** class.
- The **operation** class describing the machining strategy.

In our example, this maps to the following:

- More Worked Feature: a straight through hole corresponds to the feature class ‘STEP1HOLE’.
- Less Worked Feature: when starting from full material this corresponds to the feature class ‘BLANK’.

- Tool: the common drill corresponds to the ‘TWIST_DRILL’ cutting tool class.
- Operation: the drill motion is defined by the ‘DRILLING’ strategy.

Activity: Define a MachiningRule for Drilling a Step1Hole

- *Activate the Machining Knowledge tab in the TreeView and expand the tree.*
- *Move the mouse over ‘MachiningRuleLibrary’, right-click and choose* ***New...*** *from the menu.*
- *Check class* MachiningRule *and leave the amount to 1.*
- *Expand ‘MachiningRuleLibrary’ by clicking the + sign and select MachiningRule*.*
- *Rename MachiningRule* to Drill_STEP1HOLE_Direct. You can either do this in the Name field or by pressing F2 on the node name in the Tree.*
- *Press F8 in the OperationClass field and select DRILLING from the list.*
- *Press F8 in the More Worked Features field and select STEP1HOLE from the list.*
- *Press F8 in the Less Worked Features field and select BLANK from the list.*
- *Press F8 in the Tool field and select TWIST_DRILL from the list*
- *Set the Priority = 3.*

Conditions

By creating the MachiningRule, we only defined that you can drill a straight hole with a drill. But we did not yet define any such details as:

- when we want to use this MachiningRule,
- what size of tool do we want to use,
- or how deep we want to drill.

As a result, the system will use this operation for any hole independent of size or quality, and will select any drill without regard of the size. In order to create a useful MachiningRule, this information needs to be added. This is done by defining conditions for the MachiningRule. In general, the conditions can be grouped into 4 categories:

- “(1) Application Criteria” describing when the MachiningRule is allowed.
- “(2) Tool attributes” describing the details of the tool.

- "(3) Less Worked Feature attributes" describing the details of the starting geometry.
- "(4) Operation Attributes" describing the details of the operation.

The next sections will detail these conditions.

(1) Application Criteria

Application criteria define when a MachiningRule is allowed. In general, these conditions are related to the size and quality of the More Worked Feature. Examples are:

- The roughnesses that can be achieved: e.g. drilling is only allowed until a certain roughness, otherwise reaming/boring should be used.
- The size tolerances that can be achieved: e.g. drilling is only allowed until a certain diameter tolerance, otherwise boring or reaming should be used.
- The shape of the feature: e.g. drilling with a straight drill will finish the shape only when there are no chamfers.
- The size of the feature: e.g. drilling is only allowed when the feature is not too deep or too large, otherwise another operation type should be used.

Conditions are defined as text in the Conditions tab of the Machining Knowledge Editor. An example of a condition is the following group of lines:

```
REM This condition checks for the achievable side roughness
roughness_value( mwf.SIDE_ROUGHNESS_1 ) >= 1.6
$$$$ Cannot achieve roughness on side surface
```

Each line in this example has a different function. The first line starting with 'REM' is considered as remark and do not have any effect on the behavior of the program. The programmer can use remarks to include short explanations or observations. It is advised to add remarks so others can understand the behavior of the MachiningRule and the reason for the conditions. There can be several lines of remarks following each other, but each line needs to start with 'REM '.

The second line

```
roughness_value(mwf.SIDE_ROUGHNESS_1) >= 1.6
```

is the actual condition defining in this case that the SIDE_ROUGHNESS_1 attribute of the mwf (More Worked Feature) should be larger or equal than 1.6. The

MachiningRule is allowed if the condition is true. In this example, the MachiningRule will be allowed if the SIDE_ROUGHNESS_1 = 1.6 or larger but will not be allowed if SIDE_ROUGHNESS_1 = 1.5. If a condition is false, the whole MachiningRule will be invalid and the system will continue with evaluating the next MachiningRule.

The third line starting with $$ defines the message that will be displayed when the previous condition is false. This message is part of the messages that users will see if they want to analyze why a certain solution was found. Therefore these messages should be clear enough for the users to understand why a MachiningRule was not used.

Another example of an application criteria condition is:

```
mwf.DIAMETER_1 <= 25
$$ Diameter_1 is too big to drill at once
```

(2) Tool attributes

Tool attributes criteria define the shape of the tool that will be used for the operation. Examples are:

- The diameter of the tool should be corresponding to the diameter of the mwf: e.g. the drill diameter should be smaller than the maximum toleranced size of the mwf diameter but should be larger than the minimum toleranced size.
- The tool should be long enough: e.g. the length of the drill should be larger than the depth of the hole.

This corresponds to criteria like:

```
tool.Diameter >= mwf.DIAMETER_1 + mwf.DIAMETER_1_LOWER
tool.Diameter <= mwf.DIAMETER_1 + mwf.DIAMETER_1_UPPER
tool.FluteLength > mwf.DEPTH + constant.Thru_Hole_Clearance
```

(3) Less Worked Feature attributes

When defining Less Worked Feature attribute values we define the in-process geometry. Since we start drilling from blank, which means we do not bother about the starting geometry, nothing needs to be specified here.

(4) Operation Attributes

The last category of conditions specifies attributes of the MachiningRule's operation. Examples are:

- The operation should be deep enough: e.g. the depth of the drill motion should be equal to the depth of the mwf feature, or in case of a through hole should be larger than the depth of the hole.
- The tool diameter can be larger than the modeled diameter, as long as it stays within the tolerance range.

```
oper.Maximum_Depth_Distance = mwf.DEPTH +
constant.Thru_Hole_Clearance
oper.Through_Clearance  = constant.Thru_Hole_Clearance
oper.Allow_Oversize_Tool = "true"
oper.Oversize_Tool_Percent = 100.0 * mwf.DIAMETER_1_UPPER /
mwf.DIAMETER_1
```

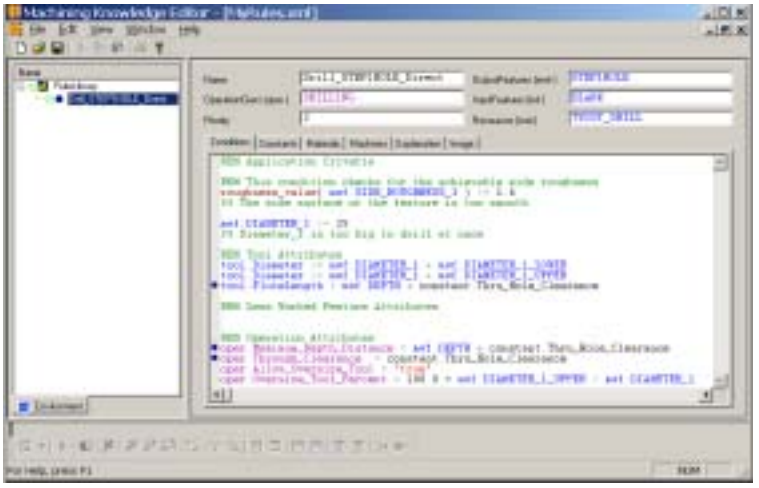
Output Load Tool. When you activate this option, the system outputs a LOAD or TURRET command in the CLSF file, even if the tool has not changed. This option can be defined as an operation parameter.

```
oper.Output_load_tool_Status ="true"
```

Activity: Define Conditions for Drilling a Step1Hole

- *Click the Conditions Tab of MachiningRule Drill_STEP1HOLE_Direct*
- *Create or Copy the conditions from the previous sections into the conditions area. Add the REM in front of the remarks. You can copy from a PDF document if you choose* ***Tools*** *>* ***Basic*** *>* ***Select*** *in the Adobe Reader.*

- *Move the mouse into the conditions area, click MB3 and choose* ***Compile (selection)***



- *The result should look as depicted above. The lines containing an undefined Constant are bookmarked.*

Constants

The only constant used in the Conditions, Thru_Hole_Clearance, should be declared and have a default value. It is good practice to declare a constant on the MachiningRuleLibrary level, so its default value is shared by all MachiningRules in the MachiningRuleLibrary.

Activity: Defining a Constant

- *Select 'MachiningRuleLibrary' and click the Constants Tab.*
- *Define a new constant, either by clicking* ***MB3 > New*** *in the constant area or by clicking the New icon.*
- *Define as Name = Thru_Hole_Clearance and as Default Value = 3.*
- *OK in the Options Dialog window.*
- *In the Machine/Material matrix, define a value = 4 mm for when the product material is CARBON_STEEL.*

- *In the tree, select node “Drill_STEP1HOLE_Direct”, right-click and choose **Compile**.*
- *There should be no more compiler errors now.*

Materials and Machines

Activity: Select for which Materials and Machines the MachiningRule is valid

- *Select “Drill_STEP1HOLE_Direct” and click the Materials tab.*
- *Unselect All Materials and select only “ALUMINIUM”.*
- *Click on the Machines tab.*
- *Unselect All Machines and mark only some of them.*
- *Undo your previous changes by checking “All Materials” as well as “All Machines”.*

Activity: Learn to use the explanation field

- *Click the Explanation tab*
- *Enter a clear textual explanation of this MachiningRule. Be sure it will be understandable for another person.*
- *Ask your tutor to review your explanation.*

Activity: Adding an Image file to a MachiningRule

- *Click the Image tab*
- *From the popup menu select **Add Image...**.*
- *On the Training CD, locate file \training_data\Machining Knowledge Editor\Drill_STEP1HOLE_Direct.JPG and press OK.*

Building the MachiningRuleLibrary

The MachiningRuleLibrary ‘contains only one MachiningRule up to now and that’s enough to do a first test. Before we can test we must build the MachiningRuleLibrary.

Activity: Building the MachiningRuleLibrary

- *Choose **File > Build***
- *This command will create a file MyRules.dll on the same location as the source file MyRules.xml*

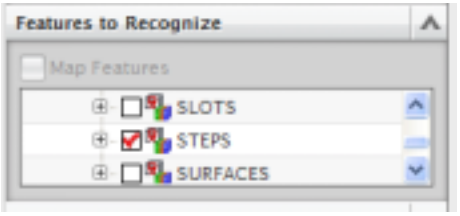
Testing the MachiningRule

In order to test the MachiningRule we must have feature instances. In real MachiningRuleLibrary development projects it's crucial to collect 'families' of feature instances in one reference product.

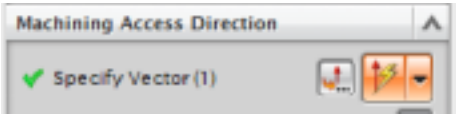
For testing the MachiningRuleLibrary we use the testmodel.prt .

Activity: Testing the MachiningRule

- *Load the testmodel.prt in NX.*
- *Start the Manufacturing Application.*
- *Select the configuration that you created in the beginning of this chapter.*
- *Switch to the 'Operation Navigator -Geometry'. Define the MCS and the WORKPIECE.*
- *Switch to the Feature Navigator. Start the **Find Features** command.*
- *Recognize only the STEP1 feature types.*

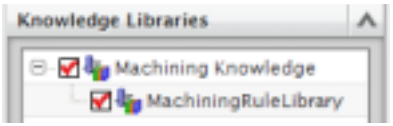


- *Use the top face of the product to define the Machining Access Direction.*



- *Press the Find Features command icon.* 
- *Execute **Create Feature Process...** on the recognized features*

- *Select the checkbox for MachiningRuleLibrary.*



- *Leave the allocation to Geometry objects at Automatic.*
- *Press **OK***

- *The rule must have been applied on 6 features. 3 Feature Geometry Groups each hold 2 features.*
- *Check the selected tools, 2 different drills are used.*
- ***Verify** the tool path of Drill_STEP1HOLE_Direct.*

Detailing the MachiningRule with Add-ons

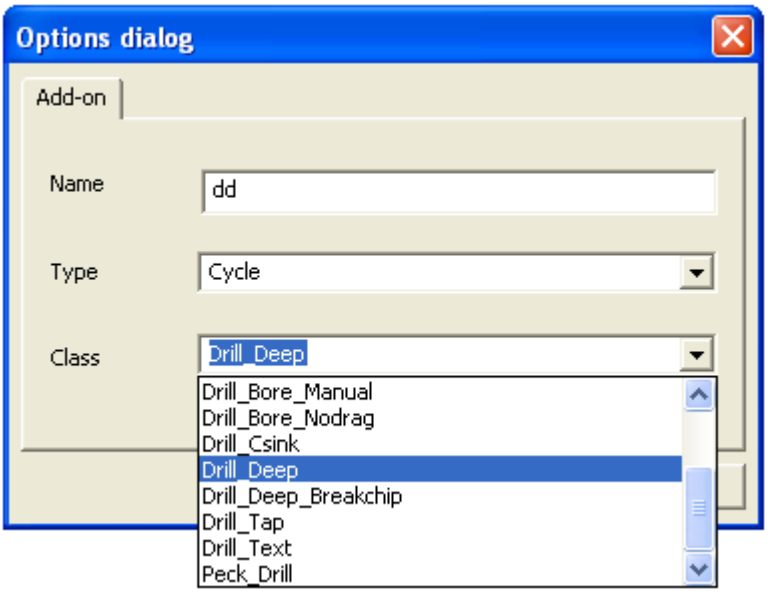
In these activities we will further

- "(1)Define the Cycle Type and Parameters" This is done by creating an Add-on to the rule.
- "(2) Define a Start/End UDE."
- "(3) Define Markers"

Activity: (1)Define the Cycle Type and Parameters

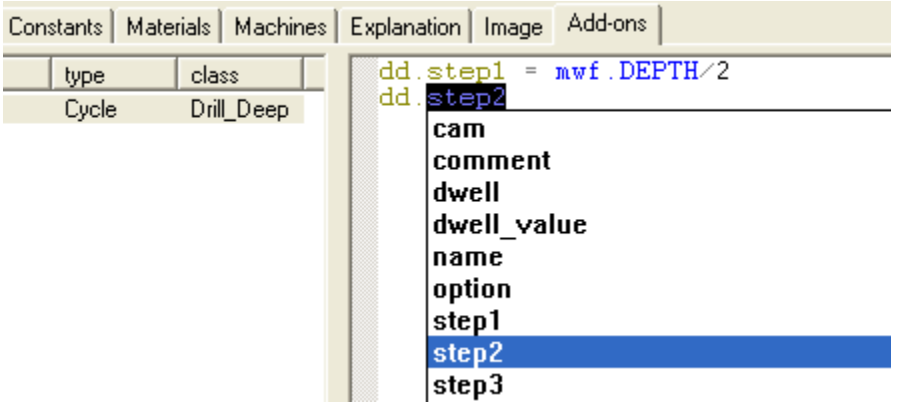
- *Click the Add-ons Tab*
- *Click New Add-on*

- You can choose any name for the object, like dd in this example.



- The Type of the Add-on object is Cycle.
- Choose the class of Cycle, like Drill_Deep in this example.

In the expression area you can enter expressions by typing the name of the object followed by a dot. You get a list of attributes that can be used in your expressions.

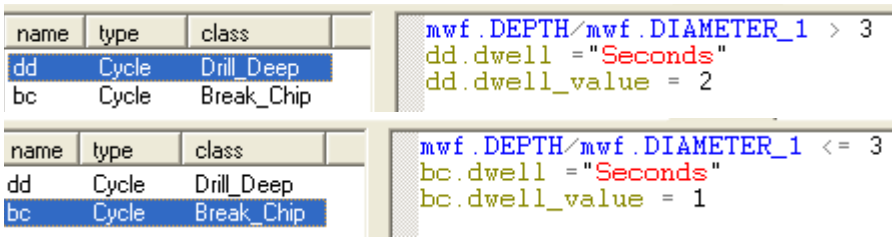


In your expressions you can use more worked feature attributes (mwf.) and tool attributes (tool.) to calculate the values of the cycle attributes. Also you can use conditional constructions with IF THEN ELSE. Use of less worked features attributes (lwf.) and operation attributes (oper.) is allowed but this seems not very likely in practice.

```
dd.step1 = mwf.DEPTH/2
dd.step1 = mwf.DEPTH/3
dd.step1 = mwf.DEPTH/6

dd.dwell ="Seconds"

IF mwf.DEPTH/ mwf.DIAMETER_1 > 3 THEN
dd.dwell_value = 2 ELSE
dd.dwell_value = 1

dd.comment = tool.comment
```

- *Define expressions for 'dd' as in screenshot above.*

On a single MachiningRule you can define multiple Cycle Add-on objects as shown below.



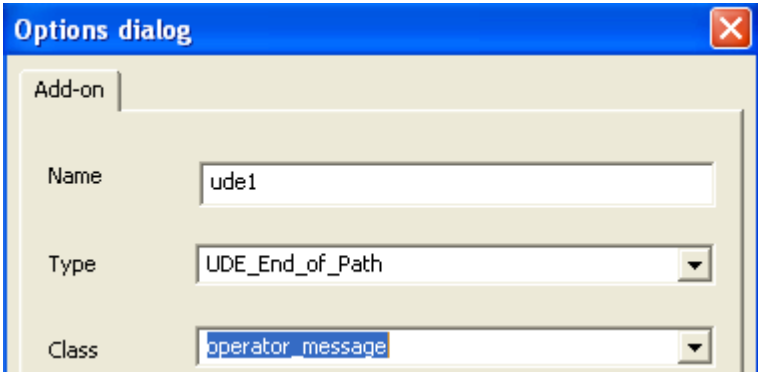
The system will create the first Cycle object of which all conditions are True. In this example the criterion is the depth/diameter ratio of 3.

- *Create a second Add-on named 'bc' with expressions as in screenshot above.*

Activity: (2) Define a Start/End UDE.

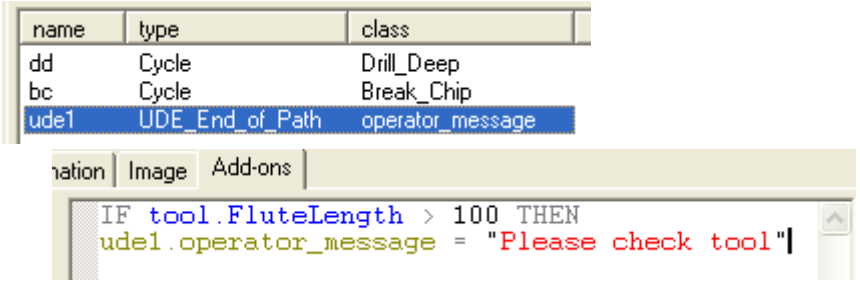
If you define multiple UDE's, the system will create all UDE objects of which all conditions are True. This is different from Cycle type objects where only one object will be created.

To create a User Defined Event (UDE) you first create an Add-on object of type UDE_Start_of_Path or UDE_End_of_Path. Choose a name and the class, like ude1 below.



In the expression area you type the name of the Add-on Object followed by a dot. Then you can choose from the list of available attributes.

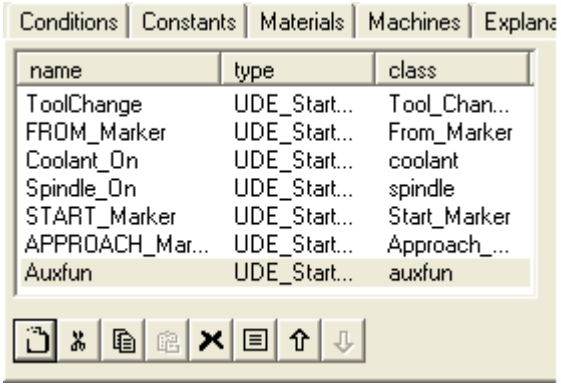
UDEs can be conditional using IF THEN ELSE constructs in the expression.



The sequence of the UDEs in the 'Defined list' is the order in the list of Add-ons. UDEs can be moved up and down using the arrow buttons.

- *Create an Add-on like 'ude1' with expressions as in screenshots above.*

Activity: (3) Define Markers



Markers will generally not have any conditions. Markers serve to tell the system where to output an event. For instance, the Tool Change Marker allows you to specify where you want the system to output the tool change event. When you insert this marker into the list of Add-ons, the system writes the commands above the marker to the CLSF before the LOAD or TURRET command, and it writes the commands below the marker to the CLSF after the LOAD or TURRET command.

- *Create a few markers like in this screenshot:*

```
fm      UDE_Start_of_Path   From_Marker
tc      UDE_Start_of_Path   Tool_Change_Marker
c       UDE_Start_of_Path   coolant
```

Defining a second MachiningRule

What this example intends to explain

We will define a MachiningRule to tap a STEP1HOLE_THREAD by cutting the thread. This will help to explain two important things:

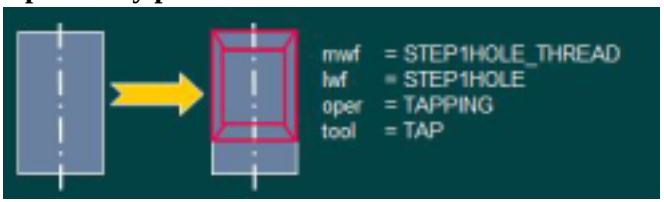
1. In addition to the previous example, you learn the basics of in-process geometry by specifying Less Worked Feature attribute values.
2. Tapping a feature will rarely be the only operation. There will usually be a drillling operation before. And we will see that *Drill_STEP1HOLE_Direct*, which we defined earlier, can do that job too. We only need to define the Tapping MachiningRule.

Note The drilling and tapping could be combined into a single tool, yes, but let's assume our company does not have these tools so they're not in our customization.

Note The instructions for defining the MachiningRule will be less detailed now.


30 minutes.

Explanatory picture



Conditions

Application Criteria

For now we do not define any Application Criteria. We always want to use this MachiningRule whenever a thread has been defined in the model.

Tool attributes

Since machine taps are usually standard tools from a catalogue, it is enough we specify just the selection criteria to select the correct tool. Most importantly these are the diameter, pitch and length.

```
tool.Diameter = mwf.THREAD_MAJOR_DIAMETER
tool.Pitch = mwf.THREAD_PITCH

tool.FluteLength > CEIL(mwf.THREAD_LENGTH)
```

The function CEIL gives us the smallest whole number that is larger then its argument.

Less Worked Feature attributes

We need to define the diameter of the less worked feature to match the core of the threaded hole. This is done with the following conditions:

REMARK The core diameter is usually calculated according to a simple formula:

```
lwf.DIAMETER_1 = mwf.THREAD_TAPPED_DRILL_SIZE
```

```
lwf.DIAMETER_1_UPPER = 0
```

```
lwf.DIAMETER_1_LOWER = mwf.DIAMETER_1_LOWER
```

Operation Attributes

REMARK the tapping tool should go as deep as the thread length.

```
oper.Maximum_Depth_Distance = mwf.THREAD_LENGTH
```

Constants

We do not make use of a constant parameter in this MachiningRule.

Materials and Machines

We will make the MachiningRule valid for Aluminum only. We make the MachiningRule valid for all machine tool type names.

Explanation

Tapping by cutting is done whenever a thread feature has been defined.

Image

See “Adding an Image file to a MachiningRule” on page 61.

Activity: Tapping a Step1Hole_Thread

- *Create the MachiningRule and change the name to “Tap_STEP1HOLE_THREAD”.*
- *Define the MachiningRule definition components to match the explanatory picture.*
- *Create or Copy the above conditions into the conditions area.*
- *In the Materials tab, select ALUMINIUM.*

- *Provide an Explanation in your own words.*
- *Add an Image to the MachiningRule. There’s an image on the training CD for this MachiningRule. Create one yourself if you have time.*
- *Check if the Cutter Diameter and Cutter Length are defined for the TAP tool class in the customization. (Use command Set as Cutter Diameter/ Length...)*

Activity: Build the MachiningRuleLibrary and test the tapping Rule

- *Move the mouse over ‘MachiningRuleLibrary’, click MB3 and choose **Build**.*
- *Test the MachiningRules with **Create Feature Process**.*
- *Check the selected tools and operations*
- ***Verify** the tool paths*

At this point it is important to understand that Rule Drill_STEP1HOLE_Direct is used for machining a feature that is a ‘final’ manufacturing feature, as well as for an in-process feature. The MachiningRule is the same in both cases.

Defining a ‘competing’ MachiningRule

What this example intends to explain

We will define a MachiningRule for reaming a STEP1HOLE.

In addition to the previous examples, you learn the basics of ‘competition’ between two or more MachiningRules having the same More Worked Feature class. The Reaming Rule you are going to create in this exercise is an alternative for Drilling and can be seen as a ‘competitor’ for Drilling.

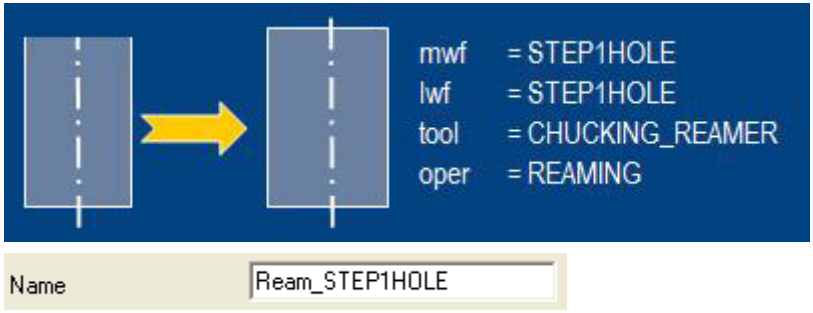
The competition between MachiningRules is controlled using the MachiningRule.Priority together with the MachiningRule’s conditions.

The MachiningRule.Priority is a numerical value which determines the relative priority of the MachiningRule, compared with other MachiningRules, producing the same More Worked Feature. The priority defines which MachiningRule will be tried first, when there are alternative MachiningRules to machine the same feature. Generally, we recommend to give cheaper MachiningRules a higher priority than

expensive MachiningRules. This way we can be sure the system will try cheap MachiningRules first and only when these cannot be applied (because of Application Criteria) try the next more expensive MachiningRule etcetera. See “How Operation Selection works” on page 39.

30 minutes.

Explanatory picture



Conditions

Application Criteria

We will test on the side roughness of the STEP1HOLE, and on the diameter tolerance bandwidth.

```
REM Application Criteria
roughness_value(mwf.SIDE_ROUGHNESS_1) >= 0.8
$$ Feature has a too smooth surface for reaming.
IT_class_ISO(mwf.DIAMETER_1,mwf.DIAMETER_1_UPPER,mwf.DIAMETER_1_LOWER) >= 6
$$ IT class Diameter_1 cannot be achieved.
```

Tool attributes

tool.Diameter >= mwf.DIAMETER_1 + mwf.DIAMETER_1_LOWER

tool.Diameter <= mwf.DIAMETER_1 + mwf.DIAMETER_1_UPPER

tool.FluteLength > CEIL(mwf.DEPTH + constant.Thru_Hole_Clearance)

Less Worked Feature attributes

```
lwf.DIAMETER_1 >= mwf.DIAMETER_1-
constant.ALLOWANCE_REAM_MAX

lwf.DIAMETER_1 <= mwf.DIAMETER_1-
constant.ALLOWANCE_REAM_MIN

lwf.DIAMETER_1_UPPER = 0.2

lwf.DIAMETER_1_LOWER = -0.2

lwf.SIDE_ROUGHNESS_1 = “12.5”
```

Operation Attributes

oper.Maximum_Depth_Distance = mwf.DEPTH + constant.Thru_Hole_Clearance

oper.Through_Clearance = constant.Thru_Hole_Clearance

Constants

We introduced a new `constant`.ALLOWANCE_REAM_MAX in the Less Worked Feature Attributes conditions above. So we must declare this one, with a default value = 2.

Similarly, declare `constant.`ALLOWANCE_REAM_MIN with a default value = 1.

Materials and Machines

MachiningRule is valid for all Materials and all Machines.

Explanation

Please enter an explanation text in your own words.

Image

Also for this MachiningRule you can find a picture in the training CD.

Activity: Reaming a Step1Hole

- *Copy Rule "Drill_STEP1HOLE_Direct" by dragging it onto its parent'MachiningRuleLibrary'.*
- *Change the MachiningRule definition components to match the explanatory picture.*
- *Since Reaming is more expensive than Drilling we should give Reaming a lower priority. Give this MachiningRule a priority = 1 and verify this is lower than the priority of Drill_STEP1HOLE_Direct.*
- *Check if the cutter diameter is defined for the CHUCKING_REAMER tool class in the Customization view.*
- *Work on all Tabs, starting with Conditions, and enter the expressions provided above.*
- *Replace the image and delete the Add-ons (all were copied).*
- *When finished, move the mouse over 'MachiningRuleLibrary', click MB3 and choose* ***Build****.*
- *Test the rules on all features with* ***Create Feature Process****.(First delete all previously generated operations, tools and geometry groups.)*
- *Locate file MSAnalysis.lis on your %TEMP% directory and try to understand how the reasoning process found a solution for this feature. Observe that MachiningRule "Drill_STEP1HOLE_Direct" is rejected the first time because it cannot achieve the required surface roughness.*

```
AF : 11 (STEP1HOLE_2)
        4 : Drill_STEP1HOLE_Direct
                [4] Condition 1 :  Cannot achieve roughness on s
        5 : Ream_STEP1HOLE
                6 : Drill_STEP1HOLE_Direct
```

Then Ream_STEP1HOLE is the next in the priority list and it is applied. The in-process geometry is then made by "Drill_STEP1HOLE_Direct".

Minimizing the number of different tools

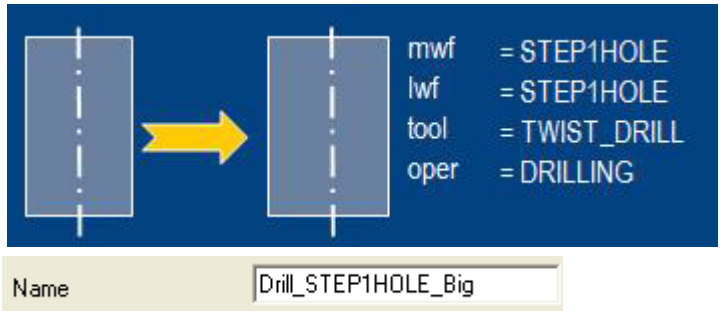
What this example intends to explain

We will define a MachiningRule **Drill_STEP1HOLE_Big** for drilling up a big STEP1HOLE with a Diameter larger than 25. Such STEP1HOLE needs to be pre-drilled. The size of the pre-drill has to be within a range with a lower- and upper boundary.

In addition to the previous examples, you learn the basics of working with ranges for in-process geometry and the NX capability to choose the in-process dimensions such that the total number of different tools is minimized. We will see that the in-process dimension is chosen such that we can use a cutting tool that is already used by another operation.


30 minutes.

Explanatory picture



Conditions

Starting with a copy of “Drill_STEP1HOLE_Direct”, only a few conditions need to be changed, these are:

Application Criteria

mwf.DIAMETER_1 > 25

$$ Rejected because Diameter_1 is too small

Tool Attributes

No changes.

Less Worked Feature attributes

It is good practice for the in-process diameter to be within 20% and 40% of the feature diameter. So:

```
lwf.DIAMETER_1 >= 0.2 * mwf.DIAMETER_1
lwf.DIAMETER_1 <= 0.4 * mwf.DIAMETER_1

lwf.DIAMETER_1_UPPER = 0.2
lwf.DIAMETER_1_LOWER = -0.2
```

Operation Attributes

No changes.

Constants

No new constant used here. But if you want you can replace the 0.2 and 0.4 by Constants.

Materials and Machines

Valid for all.

Explanation

Please enter a text in your own words.

The in-process diameter must be within 20% and 40% of the **M**ore **W**orked **F**eature diameter.
For a mwf.DIAMETER_1 = 30 this means:

6 <= lwf.DIAMETER_1 <= 12

Image

Also here we provided a .jpg image on the CD.

Activity: Create MachiningRule for drilling a big hole

- *Drag "Drill_STEP1HOLE_Direct" on its parent'MachiningRuleLibrary'. This will create a copy.*
- *Rename the copy to "Drill_STEP1HOLE_Big".*
- *Change the Less Worked Feature to* STEP1HOLE.
- *Set the priority to 2*
- *Work on all Tabs, starting with Conditions, and enter the expressions provided above. Replace the Image and delete the Add-ons that were copied.*
- *When finished, move the mouse over 'MachiningRuleLibrary', click MB3 and choose* ***Build****.*
- *Test the rules with* ***Create Feature Process****.*

Thread Milling

Thread Milling is new in NX 7.5

Training material is still under development.

Using functions

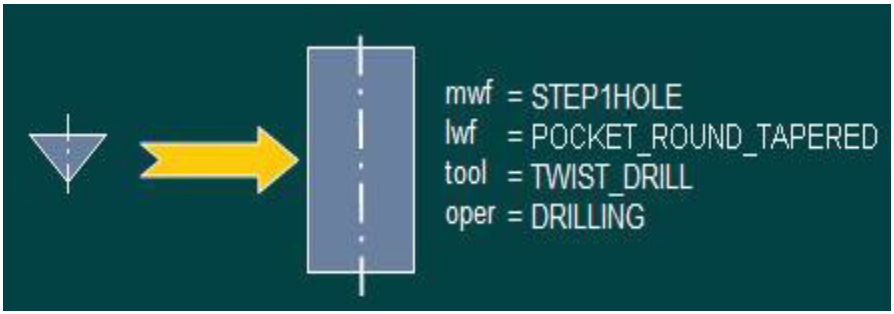
What this example intends to explain

We will define two MachiningRules: **Drill_chamfered_STEP1HOLE** for drilling a STEP1HOLE with a chamfer starting from a spot drilled pocket, and **Spot_drill** for drilling the spot drilled pocket.

In addition to the previous examples, you learn to use some functions available in the Machining Knowledge Editor.

30 minutes.

Explanatory picture



We first will define a new MachiningRule **Drill_chamfered_STEP1HOLE** for drilling a STEP1HOLE with a chamfer starting from a spot drilled pocket. This MachiningRule can be started as copy from the existing Drill_STEP1HOLE_Direct. The main changes will be in the less worked feature, which will be changed to a POCKET_ROUND_TAPERED and we will give the rule a higher priority.

Conditions

Starting with a copy of "Drill_STEP1HOLE_Direct", only a few conditions need to be changed, these are:

Application Criteria

We need to add 2 additional conditions to check for the existance of a top chamfer:

```
mwf.DEPTH_TOP_CHAMFER > 0

$$ There is no top chamfer defined.

mwf.ANGLE_TOP_CHAMFER > 0

$$ Feature has no top chamfer angle.
```

Tool Attributes

No changes in the copied MachiningRule.

Less Worked Feature attributes

We need to define the shape of the pocket_round_tapered. This shape depends on the top chamfer parameters and can be described with:

```
lwf.DIAMETER_1 = mwf.DIAMETER_1 +
2*mwf.DEPTH_TOP_CHAMFER*TAN(mwf.ANGLE_TOP_CHAMFER)

lwf.DIAMETER_2 = 0

lwf.DEPTH=lwf.DIAMETER_1/2 * TAN(mwf.ANGLE_TOP_CHAMFER)
```

Operation Attributes

No changes.

Constants

No changes.

Materials and Machines

Valid for all.

Explanation

Please enter a text in your own words.

Image

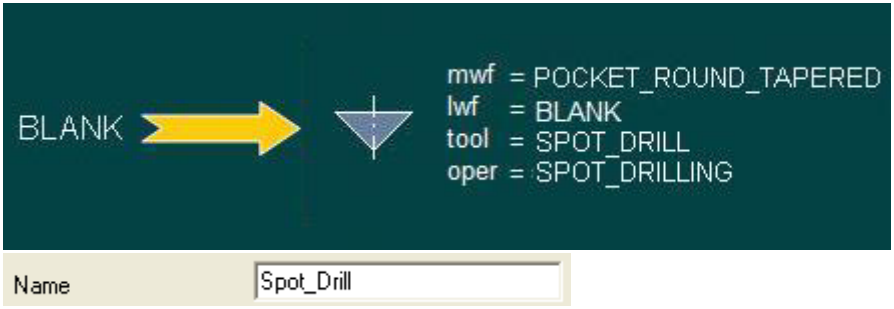
None provided.

Activity: Create MachiningRule for drilling a chamfered hole

- *Drag "Drill_STEP1HOLE_Direct" on 'MachiningRuleLibrary'. This will create a copy.*
- *Rename the copy to "Drill_Chamfered_STEP1HOLE".*

- *Change the Less Worked Feature to* POCKET_ROUND_TAPERED.
- *Set the priority to 4*
- *Work on all Tabs, starting with Conditions, and enter the expressions provided above.*

Explanatory picture



Next we define a new MachiningRule **Spot_drill** for drilling the POCKET_ROUND_TAPERED. This requires a new MachiningRule with as more worked feature a POCKET_ROUND_TAPERED feature, a less worked feature BLANK, a tool SPOT_DRILL and an operation class SPOT_DRILLING.

Conditions

All conditions need to be defined since we are not copying an existing rule.

Application Criteria

We will not need any application criteria as the operation should be used whenever there is a POCKET_ROUND_TAPERED .

Tool Attributes

The tool should be large enough for the diameter and have the correct point angle. This is defined with:

tool.Diameter > mwf.DIAMETER_1

tool.PointAngle = 2*ATAN(0.5*mwf.DIAMETER_1/mwf.DEPTH)

Less Worked Feature attributes

The less worked feature is a BLANK feature. We do not need to define any conditions.

Operation Attributes

No extra conditions, so we will use the template settings.

Constants

No constants needed.

Materials and Machines

Valid for all.

Explanation

Please enter a text in your own words.

Image

None provided.

Activity: Create MachiningRule for spot drilling

- *Create a new Rule 'Spot_drill' according to the previous description.*
- *When finished, move the mouse over 'MachiningRuleLibrary', click MB3 and choose **Build**.*
- *Test the rules with **Create Feature Process**.*

Using TableView to query the MachiningRuleLibrary

What this example intends to explain

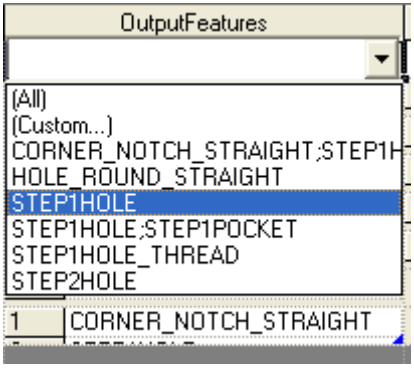
We are able to search MachiningRules by filtering on the Rule Definition components.


Estimated 10 minutes.

Activity: Searching MachiningRules with MWF = STEP1HOLE

- *Select TableView configuration Knowledge > Rule*
- *Activate all Rules of your own Library*
- *Press TableView toolbar button Autofilter*

- *Select* STEP1HOLE *from the drop down list.*



- *Verify there are 3 MachiningRules with that output feature.*

Using Find and Replace functions


Estimated 30 minutes.

Activity: Using Find in all Rule Conditions

- *Deactivate all nodes*
- *Press MB3 while the cursor is in the Conditions Tab of the first MachiningRule of your Library.*
- *Execute Find in all Rule Conditions...*
- *Enter something as search string, like DIAMETER_2 and press OK.*
- *Verify if the activated Rule(s) have a condition on the search string.*

Activity: Using Find/Replace

- *Replace REM by REMARK and find out that both compile OK.*

Additional stuff

What this example intends to explain

This exercise prepares us for ‘knowledge acquisition’ that is part of each customization project and further elaborated in “Machining Knowledge Customization Project” on page 131.


Estimated 90 minutes.

In case there’s still time, create a set of Rules for machining a POCKET_RECTANGULAR_STRAIGHT feature. In general, defining a RuleLibrary requires:

- identifying the feature types that should be covered
- defining possible operation sequences for each feature type
- defining the decision criteria to select between the sequences
- defining MachiningRules for each step in the sequence

This analysis always starts by questioning the experts on common work practices. During the process, iterations are often necessary as common work practices contain decision criteria or sequences that are not always clearly and consistently defined ('I always do it that way') and sometimes implicitly ('Of course, you never use the same tool for roughing and finishing'). This normally leads to problems during testing and to the need of additional criteria to get the right sequences. The task of the knowledge engineer is to identify possible problem areas and to ask for additional information in these areas.

An example of the definition of a set of sequences can be the following (based on the feature type POCKET_RECTANGULAR_STRAIGHT):

- Finishing is only necessary if one of the surfaces has a roughness < 6.3
- Only the surfaces with a roughness < 6.3 should be finished. This finishing requires a stock of 0.5 mm.
- Bottom and side surfaces can only be finished together if the depth <= 5 mm
- If roughing requires more than two passes along the side surfaces, then roughing should be split in to two operations: one with the largest possible mill removing the bulk of the material, followed by a second operation removing the remaining material in the corners.

Activity: Create a set of MachiningRules for POCKET_RECTANGULAR_STRAIGHT features (1)

- *Based on the description in the previous section, identify how many MachiningRules are necessary.*
- *Analyze if the data is complete*
- *What should a test product look like?*

In this specification there are already lots of unspecified items where additional information is necessary, like:

- What operation types should be used?
- Which tool classes should be used?
- Can the same tool be used for roughing and finishing? If not: what is the difference between a roughing tool and a finishing tool?
- Are the specified limits general or specific for a certain material?

- When only a section of the feature is finished, what is the required distance to the other surfaces that are not going to be finished in the operation?
- When the operation is manufacturing the corners: can the tool diameter be equal to the corner radius of the feature or should the tool diameter be smaller?
- When does roughing require more than two passes?

Activity: Create a set of MachiningRules for POCKET_RECTANGULAR_STRAIGHT features (2)

- *Define the MachiningRules. Use OperationClass='CAVITY_MILL' for roughing operations and OperationClass='FACE_MILLING_AREA' for finishing operations. Use as Resources='END_MILL_NON_INDEXABLE'.*
- *Test the MachiningRules on a product covering all situations.*

CHAPTER 4

Machining Features from opposite directions

When both sides of a hole need chamfering

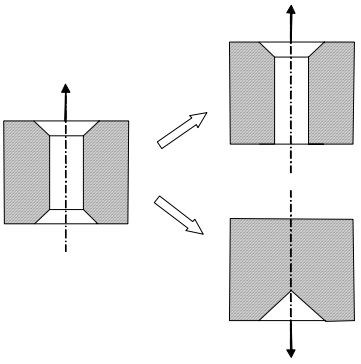
What this example intends to explain

Estimated 60 minutes.

It can be necessary to machine a feature from opposite directions, for instance when chamfering both sides of a through hole. That requires an operation chamfering the top of the feature and an operation chamfering the bottom of the feature. These operations can come from opposite directions.

Currently, with the Machining Knowledge Editor, we cannot define the operation direction. This is done implicitly through the vector attribute orientation_d of the feature. Consequently, when a feature needs to be machined from two directions, the feature should be split in two features each with their own (opposite) direction. Once the feature has been split, the created features will be solved using normal MachiningRules.

Creating a rule to split features



This section explains the definition of a MachiningRule which splits a feature into two features each with their own (opposite) direction. This is described with the example of chamfering a through hole from both sides. In this case the feature will be split in two:

- A section which will be used to drill the hole and to chamfer the top of the hole.
- A section which will be used to chamfer the bottom of the hole.

Splitting the original feature in two sections is accomplished by defining a MachiningRule in the knowledge editor with the following definition components (see screenshot below):

- OperationClass: DummyOperation. This type of operation does not require a tool and will not show up in the final operation sequence.
- OutputFeatures: the complete original feature. In this example the hole with chamfers on both sides is recognized as a STEP1HOLE.
- InputFeatures: the different features in which the original feature is split. All features should be separated by commas. In this example the top section is a STEP1HOLE and the bottom section a POCKET_ROUND_TAPERED. The sequence of classes is important for the conditions: the attributes of the first InputFeature can be accessed with lwf_1., the attributes of the second with lwf_2., etc.
- Resources: Should remain empty with a DummyOperation operation.
- Priority: If this rule should be considered as first operation then the priority must necessarily be high.

Defining the conditions

The decision when to use this MachiningRule is at first sight simple: there has to be a bottom chamfer, otherwise it is not necessary to chamfer from the back side of the feature. This can be defined with the following Application Criteria conditions:

```
mwf.DEPTH_BOTTOM_CHAMFER_1 > 0
$$ This feature does not have a chamfer on the back side

mwf.ANGLE_BOTTOM_CHAMFER_1 > 0
$$ This method does not support rounded chamfers
```

The main portion of the conditions will be the definition of the InputFeatures. The definition uses a notation using lwf_1, lwf_2, etc where the number is the position of the feature in the InputFeatures field. In the example lwf_1 will be the STEP1HOLE, lwf_2 is the POCKET_ROUND_TAPERED.

In the example, the position and orientation of the STEP1HOLE are the same as the position and orientation of the original feature. As this operation has multiple InputFeatures, the software will **not** automatically copy the attributes from Output-Feature to InputFeature. Therefore all InputFeatures attributes have to be explicitly defined. The position and orientation of the STEP1HOLE have to be defined explicitly with the following conditions:

```
REM lwf top feature has the same position as mwf feature
lwf_1.X_POSITION = mwf.X_POSITION
lwf_1.Y_POSITION = mwf.Y_POSITION
lwf_1.Z_POSITION = mwf.Z_POSITION

REM lwf top feature has the same direction as mwf feature
lwf_1.X_ORIENTATION_D = mwf.X_ORIENTATION_D
lwf_1.Y_ORIENTATION_D = mwf.Y_ORIENTATION_D
lwf_1.Z_ORIENTATION_D = mwf.Z_ORIENTATION_D
```

Also the other attributes of the STEP1HOLE have to be defined as nothing is copied automatically:

```
REM set the depth of the lwf top feature
lwf_1.DEPTH = mwf.DEPTH
lwf_1.DEPTH_LOWER = mwf.DEPTH_LOWER
lwf_1.DEPTH_UPPER = mwf.DEPTH_UPPER
lwf_1.DIAMETER_1 = mwf.DIAMETER_1
lwf_1.DIAMETER_1_LOWER = mwf.DIAMETER_1_LOWER
```

```
lwf_1.DIAMETER_1_UPPER = mwf.DIAMETER_1_UPPER
.....
```

The exception is the definition of the lower chamfer. The new feature should not have a bottom chamfer. If it had, the reasoning engine would get in an infinite loop. Not having a bottom chamfer is defined by:

```
REM remove the bottom chamfer from the lwf top feature
lwf_1.ANGLE_BOTTOM_CHAMFER_1 = 0
lwf_1.DEPTH_BOTTOM_CHAMFER_1 = 0
lwf_1.RADIUS_BOTTOM_CHAMFER_1 = 0
```

The position of the POCKET_ROUND_TAPERED is at the opposite side of the original feature. This position is defined with the following conditions:

```
REM the position of the lwf bottom feature is on the opposite side of the mwf feature
lwf_2.X_POSITION = mwf.X_POSITION + mwf.DEPTH * mwf.X_ORIENTATION_D
lwf_2.Y_POSITION = mwf.Y_POSITION + mwf.DEPTH * mwf.Y_ORIENTATION_D
lwf_2.Z_POSITION = mwf.Z_POSITION + mwf.DEPTH * mwf.Z_ORIENTATION_D
```

The orientation of the POCKET_ROUND_TAPERED is opposite from the original feature. This orientation is defined with the following conditions:

```
REM lwf bottom feature has the opposite direction as mwf feature
lwf_2.X_ORIENTATION_D = - mwf.X_ORIENTATION_D
lwf_2.Y_ORIENTATION_D = - mwf.Y_ORIENTATION_D
lwf_2.Z_ORIENTATION_D = - mwf.Z_ORIENTATION_D
```

Also the other attributes of the POCKET_ROUND_TAPERED have to be defined:

```
REM define depth of lwf bottom feature
lwf_2.DEPTH = mwf.DEPTH_BOTTOM_CHAMFER_1 +
(mwf.DIAMETER_1 / (2 * TAN(mwf.ANGLE_BOTTOM_CHAMFER_1)))

REM define top diameter of lwf bottom feature
lwf_2.DIAMETER_1 = mwf.DIAMETER_1 + (2 *
mwf.DEPTH_BOTTOM_CHAMFER_1 *
TAN(mwf.ANGLE_BOTTOM_CHAMFER_1) )

REM define bottom diameter of lwf bottom feature
lwf_2.DIAMETER_2 = 0
```

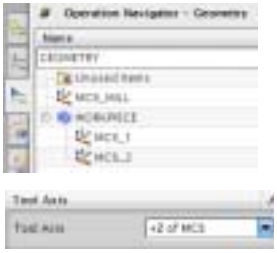
Activity: Creating a MachiningRule for splitting a STEP1HOLE feature with back chamfer

- *Create a MachiningRule with Rule definition components as described above. Complete the conditions for all InputFeature attributes.*
- ***File > Build...** when you're ready.*

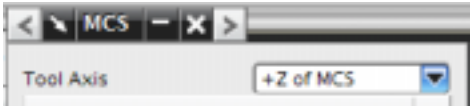
Testing the rule

You may have to design your own model for testing this rule. A simple block with a thru hole that is chamfered on both ends will do. Create a manufacturing setup for this part and perform **Find Features...**

The product has to have an MCS for every machining direction. Each MCS should be defined with **Tool Axis = +Z of MCS**. In this screenshot, two MCS's have been defined with opposite directions.

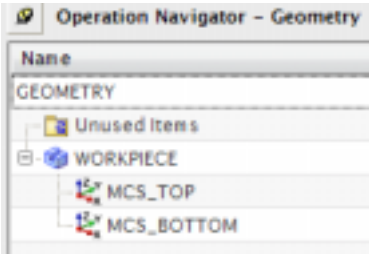


In case the dialog on the MCS does not show the Tool Axis box as in picture below, you must use mb3 **Object > Customize...** on the MCS object and add ToolAxis to the dialog.

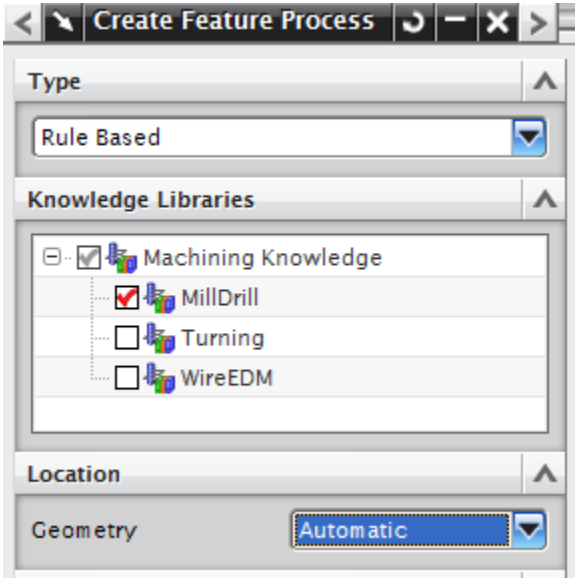


Activity: Test the Rule for splitting a STEP1HOLE feature with back chamfer

- *Define the necessary MCS objects for the test model, as below.*



- *Run **Create Feature Process...***



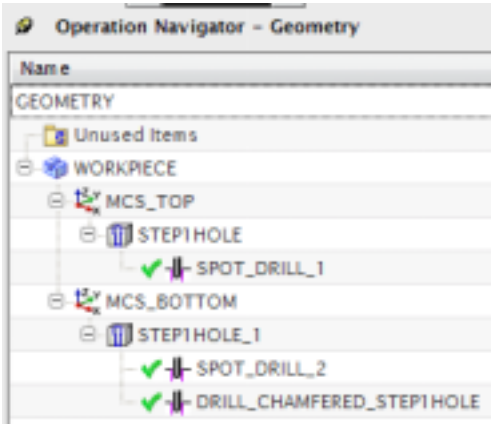
- *In the **Location** box, choose Geometry = Automatic.*

The **Automatic** allocation to Geometry moves the operations under an MCS if they match the MCS direction. The MCS's are tried in a top-down order. Operations that cannot be moved under MCS_1 are subsequently tried under MCS_2 etc.

- *Press **OK***

The result for a feature that is split looks like right picture. (With operations generated.)

Note that the dummy operation Split_back_Chamfer_S1H is not a real operation and will therefore not appear in the Operation Navigator.



Drilling a deep hole from opposite directions

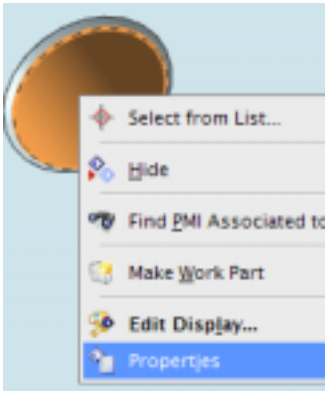
Deep holes can be too deep to be drilled from a single direction. A solution can be to drill from two directions. This section elaborates on that.

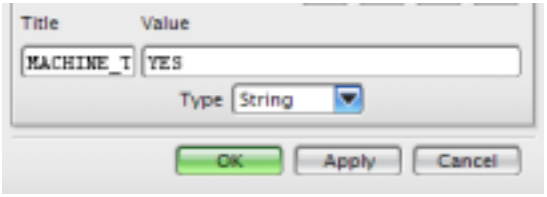
Adding a face attribute prior to Find Features...

You can manually add an attribute to a model face. This attribute and its value will be picked up by command ***Find Features...*** and will appear as a feature attribute.

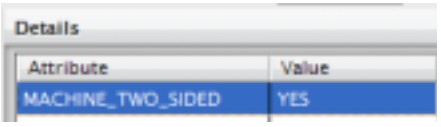
- *Choose selection class Face*

- *Click MB3 on a model face and choose Properties.*





- *Enter the Title and Value fields as shown above and press Apply or OK.*
- *Execute command **Find Features** and find back the face attribute as a feature attribute.*



Another convenient way of adding attributes to model faces and features is by feature tagging. See the NX help and command finder with keyword tag.

Testing the value of a Feature Attribute

We introduced a feature attribute MACHINE_TWO_SIDED, type String, with value *Yes* or *No*.

This attribute and its value can be assigned to a model face prior to command **Find Features...** or via tagging.

It can also be added automatically using a Knowledge Fusion function. See “Automating 2-sided machining using the KF functions added in NX6.0.3” on page 95.

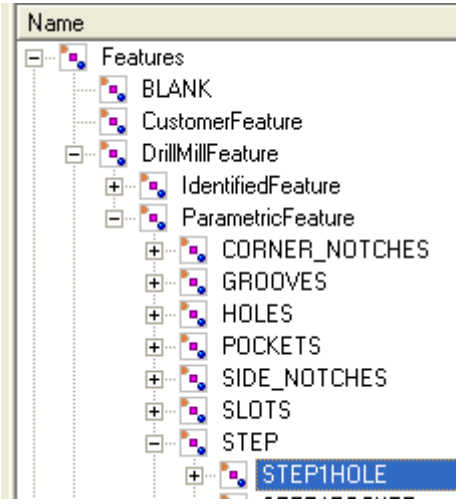
The value of the attribute (*Yes* or *No*) can be tested by a dummy rule which splits the feature when *Yes*. When *No*, the rule will be rejected and there will be no splitting.

Adding MACHINE_TWO_SIDED to the customization

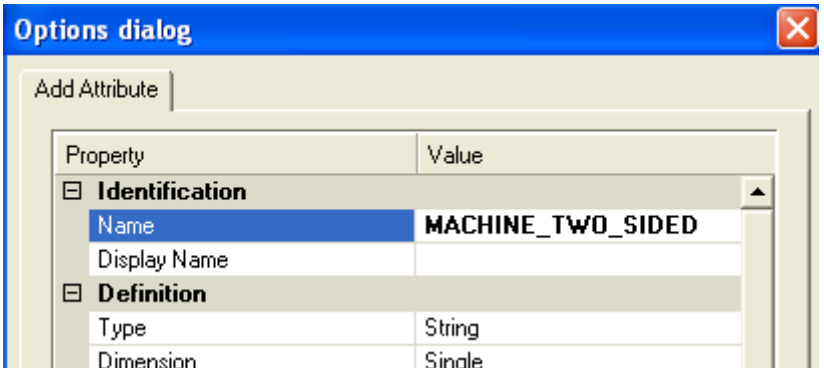
When we want to test the attribute value in the MKE, of course the attribute must be available. This is accomplished by adding it to the customization. (MACHINE_TWO_SIDED is not a standard attribute of the features.)

Activity: Adding an Attribute to the Customization

- *With MyRules.xml loaded in the MKE, choose View > Customization*
- *Drill down to STEP1HOLE*

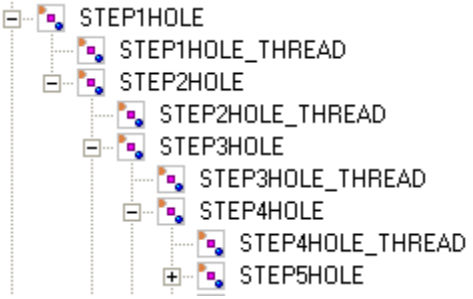


- *In the attributes window, click MB3 and choose **Add...***
- *Enter the Name and press **OK**.*



- *You see the custom attribute underlined and in a different color.*

- *Note that the attribute will be inherited by all children of the STEP1HOLE*



- *Save MyRules.xml*

With the attribute available, we can test it in a MachiningRule.

Creating a rule to split a feature

This MachiningRule can be a copy of the Split_back_Chamfer_S1H we created earlier.

However, we need to change a few things. The lwf features are both STEP1HOLE. And in the conditions we need to have the following:

REM Application Criteria

is_defined(mwf.MACHINE_TWO_SIDED)

$$ Attribute MACHINE_TWO_SIDED has no value or does not exist on the feature

mwf.MACHINE_TWO_SIDED = "*YES*"

$$ Attribute MACHINE_TWO_SIDED exists with value other than YES.

REM Less Worked Feature Attributes

These conditions are a bit different too.

- Both STEP1HOLE features need to keep the same diameter as the original.
- The position of lwf_2 will be at the bottom and its orientation reversed.
- The depths will be half the original mwf depth.
- The top chamfer attributes of lwf_2 are the bottom_chamfer attributes of the mwf.
- Both lwf_1 and lwf_2 must get MACHINE_TWO_SIDED = "*NO* " to prevent another round of splitting.

Activity: Creating a rule to split a *STEP1HOLE*

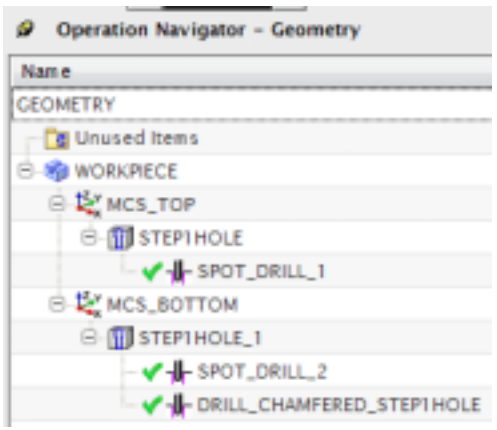
- *Copy Split_back_Chamfer_S1H and rename the copy to Split_S1H_into_2S1H*
- *In the Conditions Tab, remove existing application criteria and enter the application criteria given above.*
- *Work out the LWF assignments as outlined above.*
- *When in doubt you can import the MachiningRule "Split_S1H_into_2S1H" from the machining_knowledge.xml into your source file. (Use the Export/ Import eBOP functions)*

Activity: Testing a rule to split a *STEP1HOLE*

- *You can use the same steps as described in* ***"Test the Rule for splitting a STEP1HOLE feature with back chamfer" on page 89***

The result for a feature that is split looks like right picture. (With operations generated.)

Note that the dummy operation Split_back_Chamfer_S1H is not a real operation and will therefore not appear in the Operation Navigator.



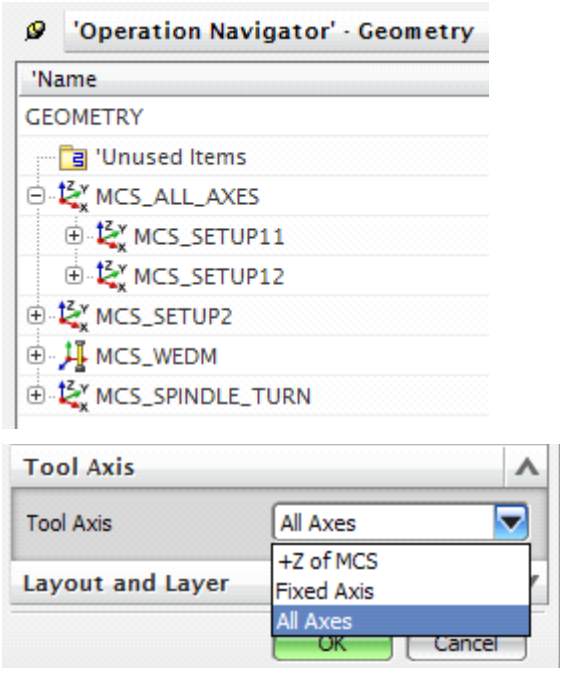
Decision making through multiple MCS's and Knowledge Fusion

Automating 2-sided machining using the KF functions added in NX6.0.3

The following KF functions were added to this purpose in NX 6.0.3:

ug_cam_askOrderedMcsNamesAndToolAxes() [ug_cam_func.dfa]

- This function retrieves an ordered list of pairs of MCS names and tool axes. The tool axis of an MCS is based on the tool axis parameter, if the tool axis parameter is 'All axes' or when the tool axis parameter is not available, like in MCS's of turning or wire EDM, the vector (0,0,0) is returned.
- Sample input:



- Sample output:
  { {"MCS_ALL_AXES",Vector(0,0,0) },
  {"MCS_SETUP11", Vector(0,0,-1) },
  {"MCS_SETUP12", Vector(0,0,1)},
  {"MCS_SETUP2", Vector(0,1,0) },
  {"MCS_WEDM",Vector(0,0,0) },
  {"MCS_SPINDLE_TURN",Vector(0,0,0) }

ug_fbm_postCreateFeaturesCustomFunction_MachineFeaturesFromTwoSides() [ug_cam_samples.dfa]

- This postCreateFeatures sample function demonstrates how you could decide if a feature has to be machined from two sides. The function sets a

`MACHINE_TWO_SIDES` attribute, which can be used in the Machining Knowledge to create a process accordingly.

- This sample function uses the new KF function `ug_cam_askOrderedMcsNamesAndToolAxes` (see 1). This sample only uses the tool axes information and the ordered MCS names are only printed (see d).

Note Make sure that this function is renamed to `ug_fbm_postCreateFeaturesCustomFunction()`, so it will be called upon the OK button of the ***Find Features...*** dialog.

The sample code:

```
#+

-------------------------------------------------------------------------------
-
Description:
   This sample function shows how to decide if a just recognized feature
   has to be machined from two sides and sets the MACHINE_TWO_SIDED attribute.

   Make sure you rename the function to
    ug_fbm_postCreateFeaturesCustomFunction(),
   so it will be called on the OK button of the Find Features dialog.
-------------------------------------------------------------------------------
-
#-
Defun: ug_fbm_postCreateFeaturesCustomFunction_MachineFeaturesFromTwoSides(
Instance $instance )
#Defun: ug_fbm_postCreateFeaturesCustomFunction( Instance $instance )
@{
  # Get the features to process
  $list_features << ug_fbm_askPostCreateFeatures();

  # Get the ordered MCS pairs
  $ordered_mcs_pairs << ug_cam_askOrderedMcsNamesAndToolAxes();

  # Collect the tool axes from the MCS pairs
  $ordered_mcs_axes << Loop
  {
    for $mcs_pair in $ordered_mcs_pairs;

    do printValue("MCS> " + First($mcs_pair) + " - " + ug_vectorToString(Second(
$mcs_pair ) ) );

         collect Second( $mcs_pair );
  };

  $result << if ( length( $list_features ) > 0 ) & ( length( $ordered_mcs_axes )
> 0 ) then
    Loop
    {
      # Process all features
```

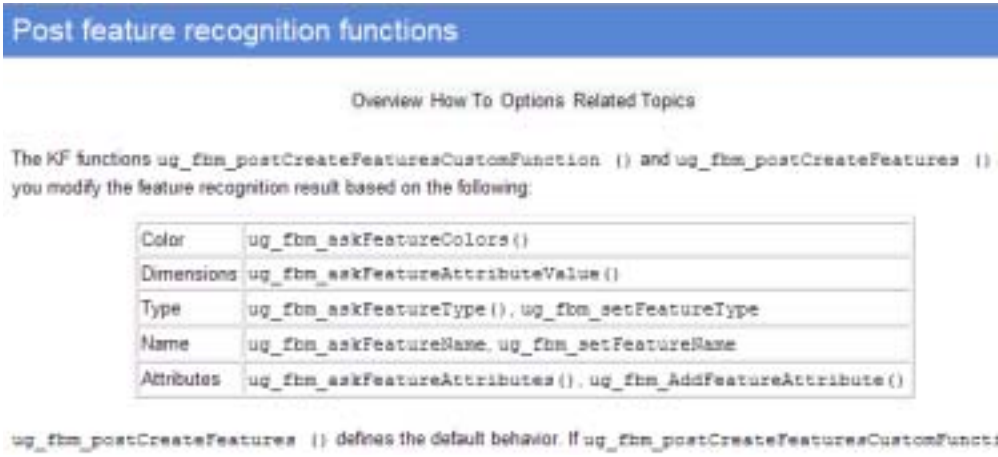
```
      for $feature in $list_features;

           # Check if the feature needs to be machined from two sides
           for $split_feat is ug_cam_askMachineFromTwoSides( $feature,
$ordered_mcs_axes );

           # Set the split feature attribute
                    do if ( $split_feat ) then  ug_fbm_AddfeatureAttribute($fea-
        ture,"MACHINE_TWO_SIDED", "YES")
               else ug_fbm_AddfeatureAttribute( $feature, "MACHINE_TWO_SIDED",
"NO" );

           return is true;
    }
  else false;


  $result;
} Boolean;
```

The logic of `ug_cam_askMachineFromTwoSides` is also defined in *ug_cam_samples.dfa*. It states that the attribute MACHINE_TWO_SIDED is set to "YES" when the feature has an alternative machining direction and an MCS with that alternative machining direction appears in the ordered list of MCS's before the main orientation. This is just a convention, so you must create the list of MCS's with this behaviour in mind.

```
Defun: ug_cam_askMachineFromTwoSides( Any $feature, List $tool_axes )

@{

# Context information

$feat_name << ug_fbm_askFeatureName( $feature );

# Get the list of accessible feature vectors

$feat_normals << ug_cam_askAccessibleVectorsOfNcFeature( $feature );

    # This example only supports two alternative feature directions, default is
no split

$do_split << if length( $feat_normals ) = 2 then

    @{

  $major_dir << First( $feat_normals );

  $alt_dir << Last( $feat_normals );

  printValue( $feat_name + "> Major direction - " + ug_vectorToString(
$major_dir ) );
```

```
  printValue( $feat_name + "> Alternative direction - " + ug_vectorToString(
$alt_dir ) );

    # First check if the alternative feature direction is available, if not do
not split

  if member( $alt_dir, $tool_axes ) & member( $major_dir, $tool_axes ) then

  @{

    $major_index << Position( $major_dir, $tool_axes );

    $alt_index << Position( $alt_dir, $tool_axes );

# Check if the alternative feature direction is before the major feature direc-
tion,

# if so split the feature otherwise do not split the feature

    if ( $alt_index < $major_index ) then

  true

else false;

  }

  else false;

}

else false;

  # Function result

  printValue( $feat_name + "> Split feature = " + StringValue( $do_split ) );

  $do_split;

} Boolean;
```

- Post_feature_recognition functions from the current online help:



Post feature recognition functions

Overview How To Options Related Topics

The KF functions ug_fbm_postCreateFeaturesCustomFunction () and ug_fbm_postCreateFeatures () you modify the feature recognition result based on the following:

Color	ug_fbm_askFeatureColors()
Dimensions	ug_fbm_askFeatureAttributeValue()
Type	ug_fbm_askFeatureType(), ug_fbm_setFeatureType
Name	ug_fbm_askFeatureName, ug_fbm_setFeatureName
Attributes	ug_fbm_askFeatureAttributes(), ug_fbm_AddFeatureAttribute()

ug_fbm_postCreateFeatures () defines the default behavior. If ug_fbm_postCreateFeaturesCustomFunct

Checking for tool availability

Another, perhaps more tricky, use of the DummyOperation type is when it can help us to decide if we can machine a very deep thru hole from both sides. When

the hole is very deep we can split it up in 2 holes. This only makes sense when we know there is a tool long enough to machine at least half the depth. If no such tool exists we do not need to split up, since we will not find a good solution anyway.

For this purpose we can create a DummyOperation with a tool. When there is a drill with a diameter equal to the feature diameter and a length greater than half the feature depth, the rule will set lwf.machine_two_sided to *Yes*. So, the actual splitting of the mwf into 2 lwf's is not done by this rule, it merely sets the lwf attribute which means "I can be split up in two".

CHAPTER 5

Machining Knowledge Editor Exercises Part 2: Turning

Feature Based Turning

This chapter shows the new capabilities of NX 7.5 in Feature Based Turning.

You can now create Feature Based Machining processes for Turning. The workflow is identical to feature based milling and drilling. The standard Feature Based Machining installation includes an initial set of predefined Turning features that are recognized, and the Manufacturing Rules for these features.

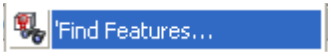
The set of predefined Turning features includes:

- TURNING_GROOVE_ID
- TURNING_GROOVE_OD
- TURNING_GROOVE_FACE

Future versions of NX will offer tools for easy definition of additional types.

Feature Recognition

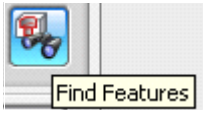
- *Load the cam_sample_fbm_2_mm.prt*
- *Start Manufacturing if necessary.*
- *Use Ctrl_Shift+K or the command icon*  *in the Tool Bar to make the workpiece solid model visible.*
- *Click on the Machining Feature Navigator icon.*  *. The MFN comes up. (hint: when you double click the icon, you can dock the MFN)*
- *In the MFN, right click in the background and choose*

- *In dialog box Type, select Parametric Recognition.*
- *In dialog box Geometry to Search, select Workpiece or All Geometry. Note that the number of entities changes from 1 to 39. Both will work.*

- *In dialog box Features to Recognize,*
  first deselect all by deselecting the top node, then select only Turning in the Fdl-RecognitionRuleLibrary.

- *Click the Find Features icon in the lower right corner of the dialogue.*



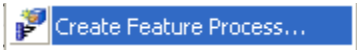
- *The Recognized Features shows 6 features (2 of each type that is currently recognized automatically) …..*

Features	Feature Type
'Features	
turning_part_demo	
TURNING_GROOVE_FACE_1	TURNING_GROOVE_FACE
TURNING_GROOVE_FACE_2	TURNING_GROOVE_FACE
TURNING_GROOVE_ID_3	TURNING_GROOVE_ID
TURNING_GROOVE_ID_4	TURNING_GROOVE_ID
TURNING_GROOVE_OD_5	TURNING_GROOVE_OD
TURNING_GROOVE_OD_6	TURNING_GROOVE_OD

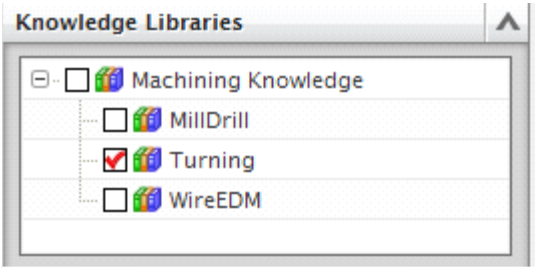
- *Click OK to accept the result.*
- *In addition to the turning features, you will find a STEP1HOLE that was already recognized before.*
- *Browse through the new TURNING features one by one. When you select a Feature in the MFN, its faces are highlighted in the selection color. Note: the TURNING_GROOVE_ID faces are inside the model.*

Operation Selection

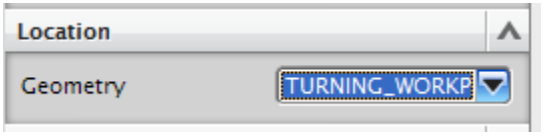
- *Hide the model geometry by pressing Ctrl + B and select the body, or use the Show/Hide command icon*  *in the Tool Bar.*
- *Press Ctrl + Alt + T for an orthogonal view on the Workplane.*
- *In the Machining Feature Navigator, select all Features, right click and execute*



- *In the Create Feature Process dialog, verify the Type field is set to* ***Rule Based****.*
- *In the Knowledge Libaries dialog box, select Turning and click OK.*



- *In the location dialog box, choose turning_workpiece.*



- *Press OK in the Create Feature Process dialog. NX will now automatically select operations for the selected turning features.*
- *Select the TURNING_GROOVE_FACE feature on the left side of the product. This feature cannot be machined in the current setup.*
- *Execute command Find Related Operations on this feature.*

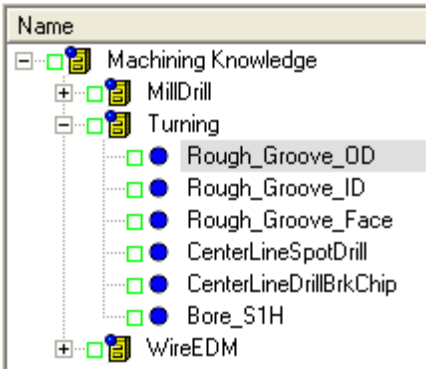
- *In the Operation Navigatir, switch to Program Order View and move the operation to the Unused Items folder.*
- *Select the Program folder and execute command Generate Tool Path.*
- *Verify the Tool Path with 2D Material Removal switched on.*

Machining Knowledge Editor

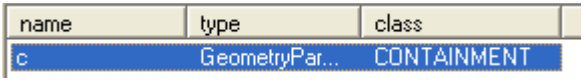
Part 2 of this demo lifts the curtain on how these automatic results came into existence, and where you could edit them if desired.

- *Start > UGS NX 7.5 > Manufacturing Tools > Machining Knowledge Editor*
- *File > Open.*
- *Select machining_knowledge.xml and press Open.*
- *Click on the Machining Knowledge Tab* 

- *Expand Turning library and select rule Rough_Groove_OD.*



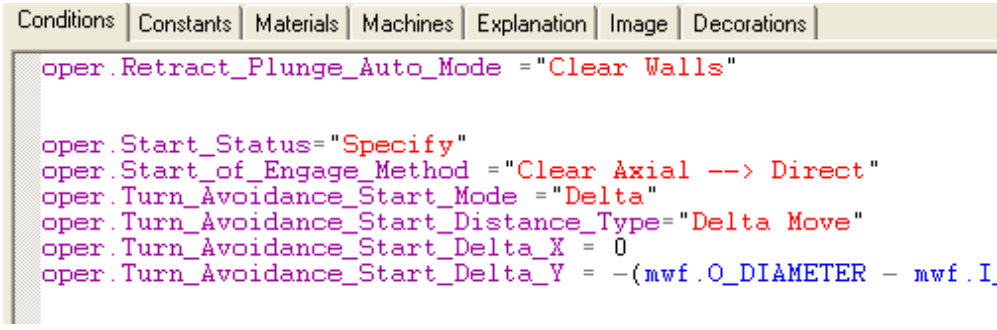
- *Click on the tab Add-ons.*
- *Select the add-on of type GeometryParent and class CONTAINMENT.*



- *On the right hand side you see the definition of the axial and radial containment, which can be expressed using the parameters of the TURNING_GROOVE_OD feature.*

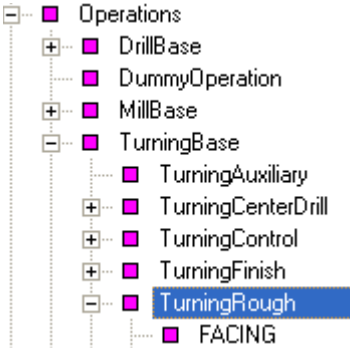
- *This specifies the axial and radial containment limits for the turning operation:*

- *Click on the Conditions tab to inspect the conditions that were defined for Rule Rough_Groove_OD.*

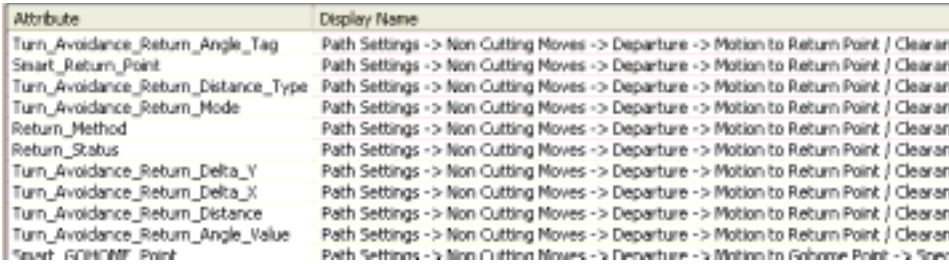


Note that the Non Cutting Moves (NCM) for the turning operation are attributes of the GROOVE_OD operation type.

- *Choose View > Customization in the menu bar.*
- *Expand the customization tree until TurningRough like this:*



- *The right-hand side of the customization view displays all attributes for the TurningRough operations which include GROOVE_OD.*
  Click once on the Display Name column header to sort alphabetically on Display Name.



Attribute	Display Name
Turn_Avoidance_Return_Angle_Tag	Path Settings -> Non Cutting Moves -> Departure -> Motion to Return Point / Clearan
Smart_Return_Point	Path Settings -> Non Cutting Moves -> Departure -> Motion to Return Point / Clearan
Turn_Avoidance_Return_Distance_Type	Path Settings -> Non Cutting Moves -> Departure -> Motion to Return Point / Clearan
Turn_Avoidance_Return_Mode	Path Settings -> Non Cutting Moves -> Departure -> Motion to Return Point / Clearan
Return_Method	Path Settings -> Non Cutting Moves -> Departure -> Motion to Return Point / Clearan
Return_Status	Path Settings -> Non Cutting Moves -> Departure -> Motion to Return Point / Clearan
Turn_Avoidance_Return_Delta_Y	Path Settings -> Non Cutting Moves -> Departure -> Motion to Return Point / Clearan
Turn_Avoidance_Return_Delta_X	Path Settings -> Non Cutting Moves -> Departure -> Motion to Return Point / Clearan
Turn_Avoidance_Return_Distance	Path Settings -> Non Cutting Moves -> Departure -> Motion to Return Point / Clearan
Turn_Avoidance_Return_Angle_Value	Path Settings -> Non Cutting Moves -> Departure -> Motion to Return Point / Clearan

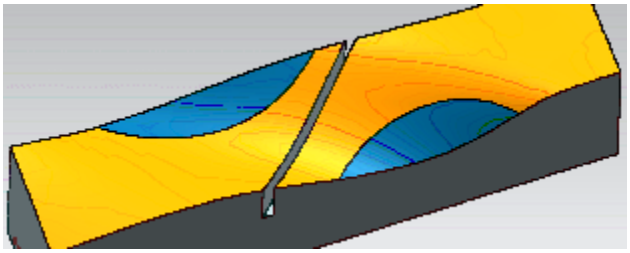
  This field is used to display the path in the OOTB NX user interface dialog for each operation Attribute. It helps the knowledge engineer in finding the NX operation Attribute when the place in NX user interface is known and vice versa.

- *Close the customization view by clicking the*  .
- *Exit the Machining Knowledge Editor.*

End of short demo of Feature Based Turning.

CHAPTER 6

Color and Attribute Features



Face color and attribute recognition

What is it?

You can specify any combination of face colors and face attributes to specify regions of a solid body or sheet body for feature recognition. Face attributes may also include a value that could determine or influence the recognition result.

Why should I use it?

You can use color and attribute recognition to help specify regions for machining that may be difficult to recognize by topology or shape.

Where do I find it?

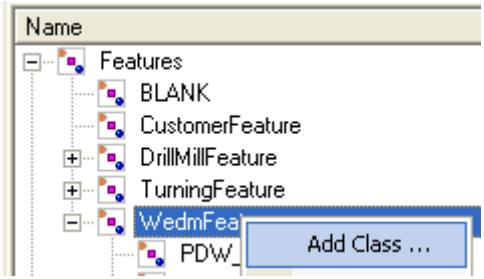
Machining Knowledge Editor > Customization view.

Below you find an example of

- how a feature is declared,
- how to define a recognition rule and
- how to use these in NX 7.5 Feature Based Machining.

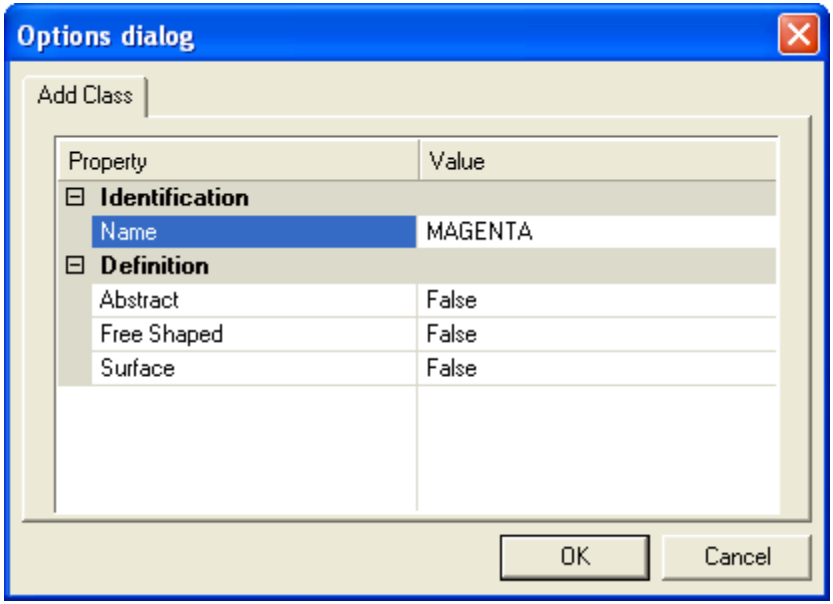
Declaring a new "Color&Attribute" Feature Type

- *Start > UGS NX 7.5 > Manufacturing Tools > Machining Knowledge Editor*
- *File > Open*
- *Select machining_knowledge.xml and press Open.*
- *Choose **View > Customization***
- *Expand Wedm Feature, right click and choose **Add Class…***



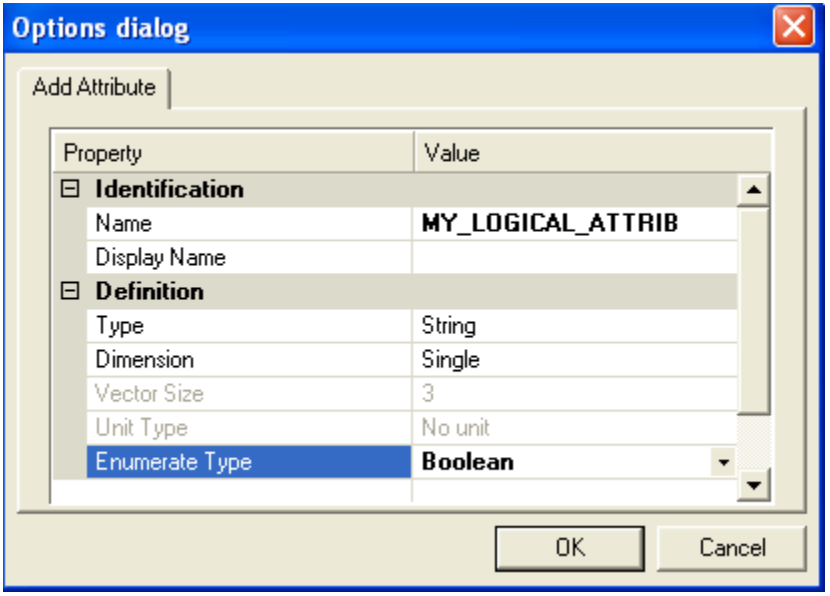
Note you can Add Class anywhere in the tree, WedmFeature is just an example.

- *In the Add Class dialog, enter Name = MAGENTA and press OK.*



- *A feature of type MAGENTA is declared. You recognize that it's a customer defined feature by the underlined type name and the dark blue color.*

- *Right click in the attributes area (right part of the screen) and choose **Add…** to add an extra attribute to the feature class. Give it any name you like, and choose the rest of the fields as shown in this example:*



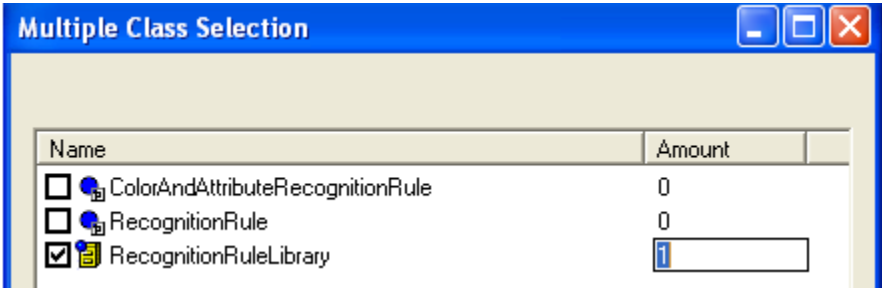
- *Press the Save icon* 
- *Dismiss the Customization view window by clicking the*  *in the upper right corner of the screen.*

This concludes the declaration of a new Color&Attributes feature type.

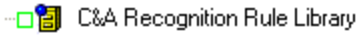
Defining a Recognition Rule for the new Feature Type

- *Click on the Recognition Tab* 

- *Create a new Library node by right clicking the upper node and choose **New…***

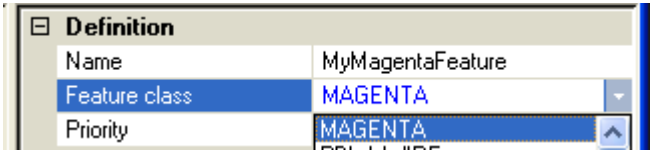


- *Rename the new Library to "C&A Recognition Rule Library"*

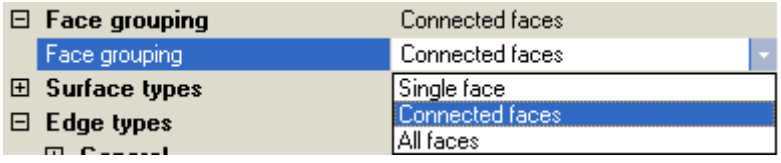


- *Create a new ColorAndAttributeRecognitionRule under the new library.*
- *Rename the ColorAndAttributeRecognitionRule* to MyMagentaFeature. (You are free to choose any name.)*

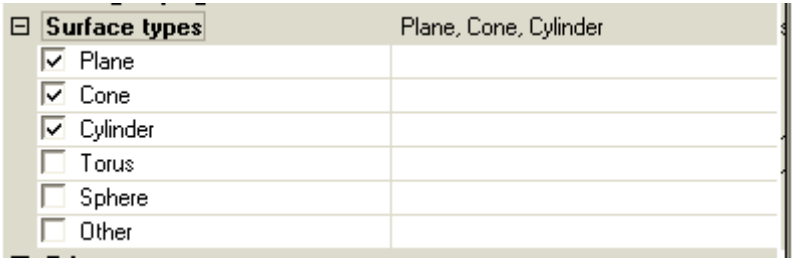
- *In the Definition Block, select the previously declared Feature class MAGENTA from the drop down list.*



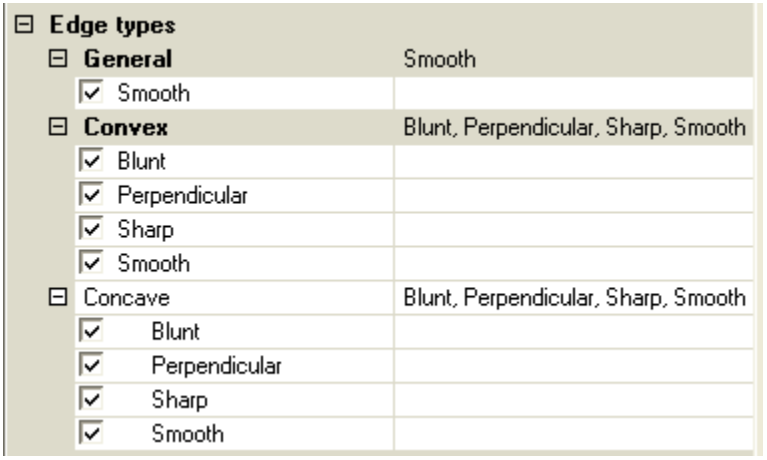
- *In the Face grouping block, leave Face grouping at Connected faces. This means that the faces of a recognized feature of type MAGENTA must be connected.*



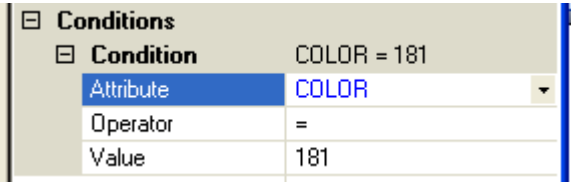
- *In the Surface types block, check Plane, Cone and Cylinder. This means that the faces of a recognized feature of type MAGENTA must be of one of these Surface types.*



- *In the Edge types block, mark all connection types. This means that all connection types are allowed. (This is only meaningful when option 'Connected faces' was chosen previously, which we did.)*



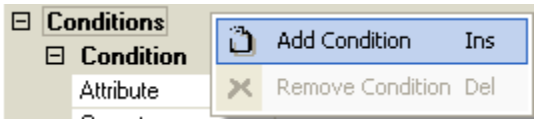
- *Create a condition that requires the feature's faces to be of color magenta..*



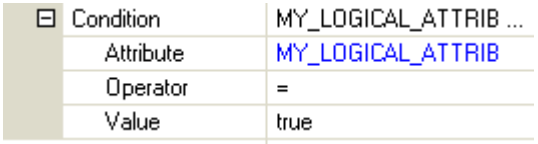
This condition means that each face belonging to the feature must have display color = 181.

Note The value (=181 in this case) corresponds to an RGB value set and a color name as defined in the Color Definition File used by the Part.

- *Right click in the header area or hit the Insert key to add a second condition.*



- *State that MY_LOGICAL_ATTRIBUTE must be 'true' by selecting from the drop down lists.*



This condition means that each face belonging to the feature must have a Face Attribute named MY_LOGICAL_ATTRIB having value = "true".

Note If a face has no such attribute or it has but its value is not "true" then it will not be in the recognized feature's face list.

- *Save the xml source file by pressing* 

This concludes the definition of the recognition recipe named MyMagenta for a feature of type MAGENTA.

Recognizing MAGENTA

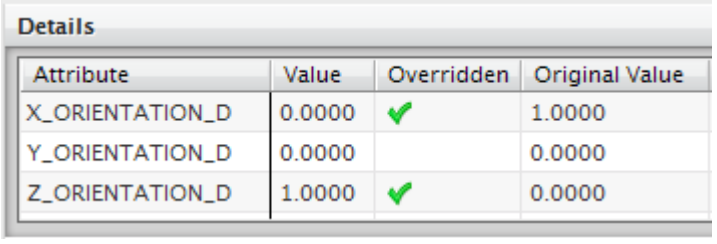
CA_Slide.prt can be downloaded from the FBM mycommunity site. When you have no access, send an e-mail to machining.support.plm@siemens.com requesting CA_slide.prt and you will receive the part via e-mail. In fact every model with color attributes can serve to recognize a feature of type MAGENTA.

- *Start NX*
- *Load CA_Slide.prt*
- *Start Manufacturing if necessary.*
- *Open the feature navigator*
- *Right-click and choose Find Features…*

- *In the Type box, choose Parametric Recognition*
- *In the Geometry to Search box, choose All Geometry*
- *In the Features to Recognize window, check MAGENTA*
- *Press Find Features*
- *Press OK and select the recognized feature.*
- *Note that MY_LOGICAL_ATTRIB is one of the feature's attributes.*
- *Also note that one of the cavity's faces is not part of the feature: when you select the feature, this face is not highlighted in the selection color.*
- *Select this face, right click for the Properties and you see why:*
- *Change the value of MY_LOGICAL_ATTRIB to true.*
- *Delete feature MAGENTA_1.*
- *Right-click and choose Find Features…*
- *Verify that the recognized feature has al faces of the cavity.*

Machining Direction of C&A Features

In NX 7.5, the direction of C&A features is chosen arbitrarily. In many cases this will result in a wrong direction. We can manually adjust the direction by supplying the correct values for the depth orientation vector, as done in screenshot below:



Machining Rules for a Color & Attribute feature

There is no difference between a parametrically recognized feature, and identified feature and a C&A feature when it comes to defining a Machining Rule. The source of the feature is irrelevant for the Operation Selection module. Therefore, creating a Machininig Rule for feature type MAGENTA follows the same path as for any aother feature.

Exercise: Create a Machining Rule for a Color and Attribute Feature

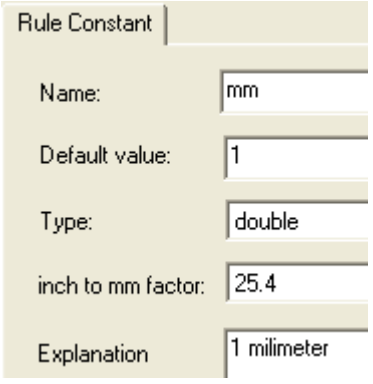
- *Launch Machining Knowledge Editor if necessary and open the machining_knowledge.xml source*
- *Create a new Rule under the RuleLibrary*
- *Further specify the Rule as per this suggestion:*

Name	Mill_Magenta	OutputFeatures (mwf.)	MAGENTA
OperationClass (oper.)	CAVITY_MILL	InputFeatures (lwf.)	BLANK
Priority	1	Resources (tool.)	END_MILL_NON_IND

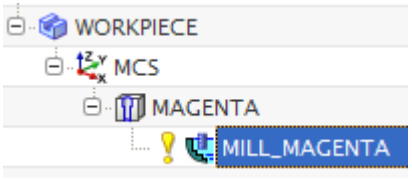
- *Choose a tool with a Diameter that is small enough to machine the corners:*

```
REM Tool Attributes

tool.Diameter < 8*constant.mm
```

Note we used a constant named mm here with value = 1. This makes the condition valid for inch parts too.



- ***File > Build...***
- *In NX, **Create Feature Process...** on feature magenta.*
- *Verify the result,as below:*



CHAPTER 7

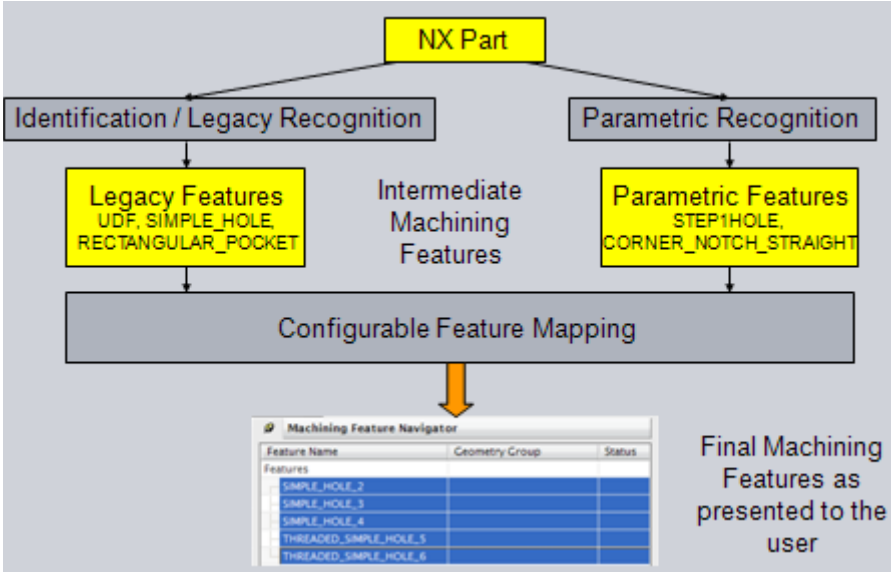
Wire EDM

content for wedm is still under development

CHAPTER 8

Feature Mapping

Feature Mapping is a powerful capability enabling us to transform features immediately after the NX command **Find Features** before they appear in the Machining Feature Navigator. Feature Mapping is treated in this tutorial since mapping rules are created with the MKE, very similar to machining rules.



Using Feature Mapping -case 1

The route sketched on the right-hand side in the above scheme. Allows customers who use their private best machining practice to benefit from the improved parametric machining feature recognition since NX 6.

- By mapping recognized features to customer specific UDF features.
- So existing automatic process selection can be applied to components that were not designed using customer specific UDF features.

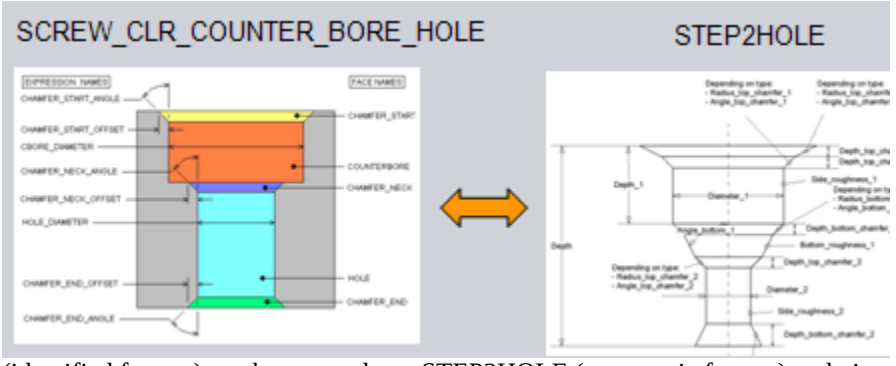
Using Feature Mapping -case 2

The route sketched on the left-hand side in the above scheme. Allows customers using UDF based design to work with NX 6 best machining practices even though these practices do not reference the UDF types.

- By mapping customer UDF features to standard NX parametric features.

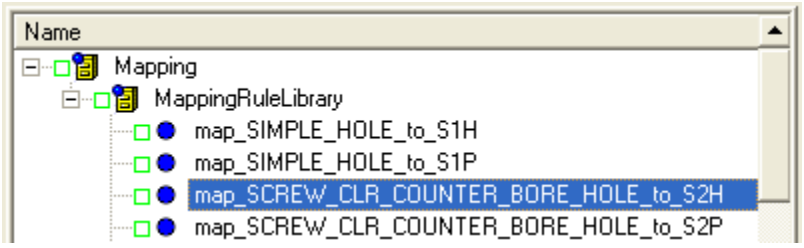
Example

The below picture illuminates how a SCREW_CLR_COUNTER_BORE_HOLE

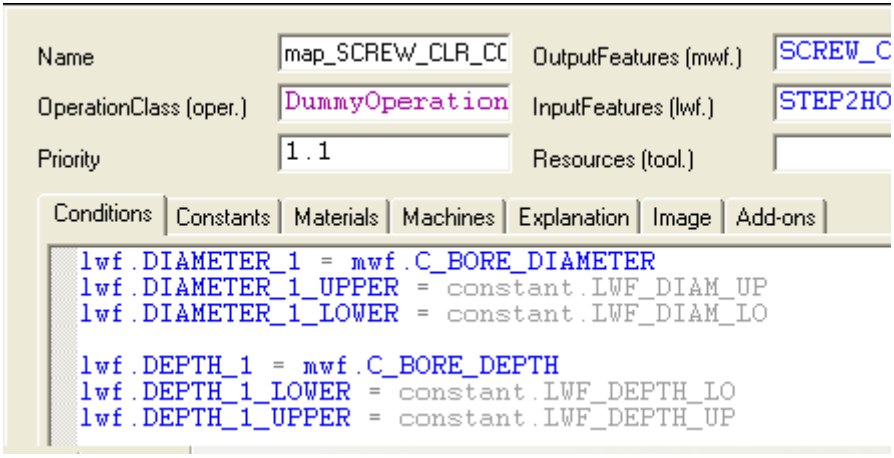


(identified feature) can be mapped to a STEP2HOLE (parametric feature) and vice versa. Not only the type of the feature is changed by the Mapping Rule, also most of the parameter values can be deduced. Sometimes parameter values are identical so they can simply be copied.

Mapping Rules are defined with the Machining Knowledge Editor (MKE) very similar to machining rules.



The Conditions of the mapping rule define how the parameters of the features map



on each other.

Note that a mapping rule always has operation class DummyOperation and has no Resource.

Mapping to Hole or Pocket feature types?

The feature type SIMPLE_HOLE can usually be through or blind. There is not always an easy decision criterium when we want to map this feature to either a STEP1HOLE or a STEP1POCKET. The same applies to most other identified features types.

In the ootb mapping rules we only map an identified feature to a through hole when we are sure the identified feature is through. The decision is made on the attribute DEPTH_LIMIT. The following conditions must be true:

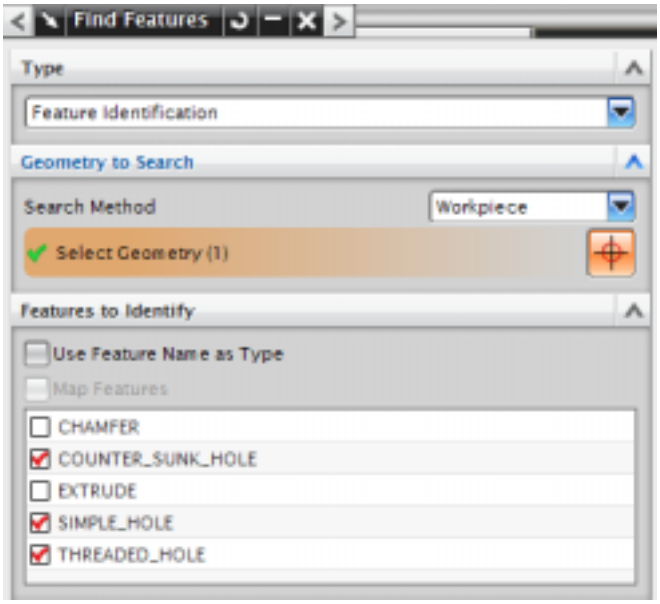
```
REM Application Criteria

is_defined(mwf.DEPTH_LIMIT)
mwf.DEPTH_LIMIT = "THROUGH BODY" OR
mwf.DEPTH_LIMIT = "UNTIL NEXT" OR
mwf.DEPTH_LIMIT = "UNTIL SELECTED"
$$ Mapping denied. This feature may map to a STEP2POCKE
```

When DEPTH_LIMIT does not exists or it has no value or its value is not one of the three mentioned above, there is a rule to map to a POCKET feature. The mapping rule to a pocket feature has a slightly lower priority.

Activity: Feature Identification not followed by feature mapping

- *Load the testmodel.prt and delete all operations and features.*

- *Execute **Find Features...** on the testmodel*



- *In the **Type** box, select Feature Identification.*
- *In **Features to Identify**, select the 3 check boxes as in screenshot above.*
- *Press the **Find Features** command icon.*
-

Activity: Feature Identification followed by feature mapping

- *Load the test_model.prt and delete all operations and features.*
- *to be complete in 7.5.028 or later*

Identification or Recognition

Feature Identification

- Re-uses the design feature's Type and Attributes
- The model faces are retrieved from the design feature.

Strength:

- Directly re-use information from the Design model
- No loss of information
- No need for “recognition”

Weakness:

- Requires that Designers model with machining features only
- A machining feature must REMOVE material (a rib can not be machined)
- Works only for models designed in NX
- Does not work together with Synchronous Modeling
- Design features often don’t really define correctly what needs to be machined.

Continue to use Feature Identification when:

- You work predominantly with in-house designed parts
- and the parts have been modeled using UDF’s
- and your company is not yet using PMI
- You work with MW and/or PDW

Feature Recognition

- Performs geometrical search against a library of “known” feature types to find features and their faces
- Performs attribute extraction per feature type.
- Recognition is not trivial and can not guarantee 100% perfect results, sometimes due to feature intersections.

The way to go when Feature Identification is not applicable.

What this example intends to explain

OperationClass class DummyOperation allows the definition of MachiningRules without a Tool. These can be used to transform a more worked feature into a less worked feature without a real machining operation.


Estimated 30 minutes.

Conditions

REM Application Criteria: there are none.
REM Tool Attributes: there are none.
REM Less Worked Feature attributes: just copy

lwf.DIAMETER_1 = mwf.DIAMETER

lwf.DIAMETER_1_LOWER = constant.LWF_DIAM_LO

lwf.DIAMETER_1_UPPER = constant.LWF_DIAM_UP

lwf.DEPTH_LOWER = constant.LWF_DEPTH_LO

lwf.DEPTH_UPPER = constant.LWF_DEPTH_UP

lwf.SIDE_ROUGHNESS_1 = constant.*achievable_Roughness_DRILL_Upper_str*

REM Operation Attributes: there are none.

Activity: Transformation of classes using a DummyOperation

- *Create a MachiningRule with MWF* SIMPLE_HOLE*, LWF* STEP1HOLE, OperationClass DummyOperation
- *Enter the Conditions as presented under “Conditions” on page 129.*
- *Work on all Tabs and when finished, move the mouse over ‘MachiningRuleLibrary’, click MB3 and choose* ***Build****.*
- *Execute* ***Create Feature Process*** *on feature SIMPLE_HOLE.*
- *Verify that the SIMPLE_HOLE is machined in 2 operations and the last is the “Dummy: Transform” operation.*

One of the places where DummyRules are heavily used is in Feature Mapping. There you can do a lot of powerful things like:

- Transform legacy feature types like SIMPLE_HOLE into STEP1HOLE.
- Transform STEP1HOLE into feature types like SIMPLE_HOLE.
- Interpret user-attributes for tolerance information and translate them into explicit numerical tolerance values
- See “Feature Mapping” on page 123.

CHAPTER 9

Machining Knowledge Customization Project

This chapter describes the OOTB content, the tasks commonly found in a customization project and the milestones to be reached.

In the context of this tutorial, the term ***Customization*** is used to describe a universe of

- machining features
- operation types
- tools
- workpiece materials
- machine tools

Since NX 7.5 this is extended with

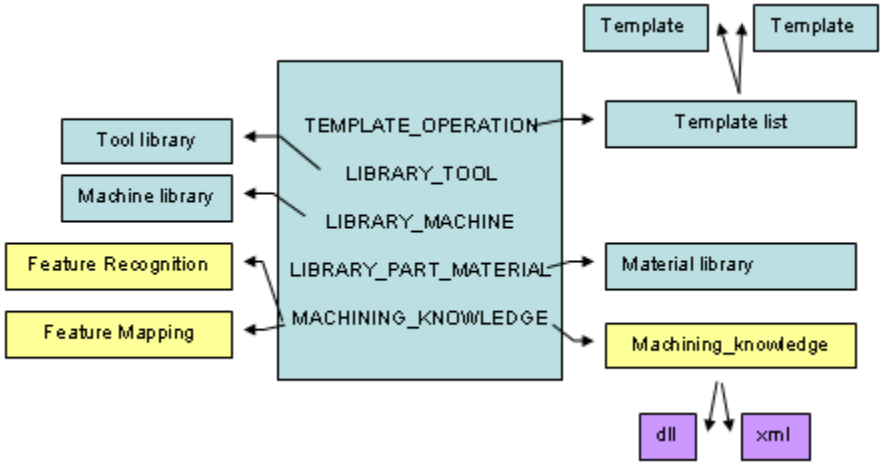
- NX Objects such as CutLevels, Cycles, UDE's and more.

Customization Files

CAM Configuration dependent customization

The root file is the CAM configuration file in the UGII_CAM_CONFIG_DIR. An example is *feature_machining.dat*. The .dat file is changed compared to NX6, since the key FEATURE_MACHINING is obsoleted in NX 7.5.

- MACHINING_KNOWLEDGE: now also points to the feature recognition rule library that is used by the ***Find Features...*** command.
- MACHINING_KNOWLEDGE: now (since NX 7.5) also points to the mapping knowledge library that is optionally used by the ***Find Features...*** command.
- MACHINING_KNOWLEDGE: points to the machining knowledge library used by the ***Create Feature Process....*** command
  .



Tools, Machines and Part Materials

More entries in the .dat file relevant for the customization are:

- LIBRARY_TOOL (the tool library)
- LIBRARY_MACHINE (the machine library)
- LIBRARY_PART_MATERIAL (materials library)

Operation Types

The TEMPLATE_OPERATION entry defines which template parts will be available and hence which operation types. If an operation type is used in more than one of the template parts, then the definition in the last template part will be used, in order of appearance in the .opt file.

Features

When you generate or update the customization with the Machining Knowledge Editor, the feature customization is built up from the following parts:

- The parametric feature types which can be recognized by command ***Find Features***.
- Any additional customer defined feature for which a recognition rule exists in the Recognition Rule Library.
- The UDF library (User Defined Features).
- Feature_definitions.def for legacy feature recognition.
- Feature_identification.def.

Default Customization

In directory MACH\machining_knowledge_editor\data there is one single system file containing the base customization of the knowledge library.

- customization_base.xml

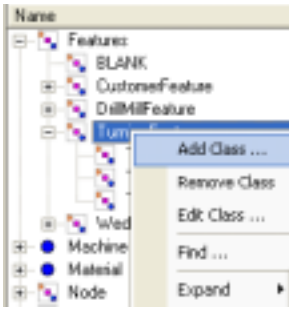
This file should not be modified as it will be replaced by software updates.

Additional Customization

Your company's specific additions can be added interactively.

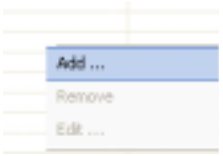
Typical additions to the customization are:

- New customer specific feature types and their attributes.



On each node in the Features branch in the Customization View you can can click MB3 and use commands to Add, Remove and Edit a feature class.

- New attributes to feature types.



For each node in the Feature branch you can can click MB3 in the attributes list and use commands to Add, Remove and Edit attributes of the selected feature class.

NX Version upgrades

With each NX version, improvements and extensions to customization_base.xml can be expected. The knowledge library customization can be updated by checking active the 'Update Upon Load' in the ***View*** **>** ***Options***. Then reload the machining_knowledge xml source.

Exercise: Understanding the customization files

- *Trace back the files mentioned in feature_machining.dat in the UGII_CAM_CONFIG_DIR. Open them in NotePad and read their contents.*
- *In the customization view of Machining Knowledge Editor, try to understand the customization of the tools, materials and machines in relation to the data listed in the files.*

OOTB content

The ootb rule content is shipped with each major release of NX. It is continuously improved based on feedback from the field. Please visit the FBM site and pull the latest Content Kit from there:

https://myc2005.ugs.com/marketing/partmfg/nxfbm/default.aspx

The ootb rule content is captured in the following source file:

- MACH\resource\machining_knowledge\machining_knowledge.xml

You can easily view the ootb rule content outside the Machining Knowledge Editor application with:

- machining_knowledge.chm

The ootb rule content is based on this template part (only the unit system is different):

- MACH\resource\template_part\metric\machining_knowledge.prt
- MACH\resource\template_part\english\machining_knowledge.prt

An explanation of the OOTB Rules

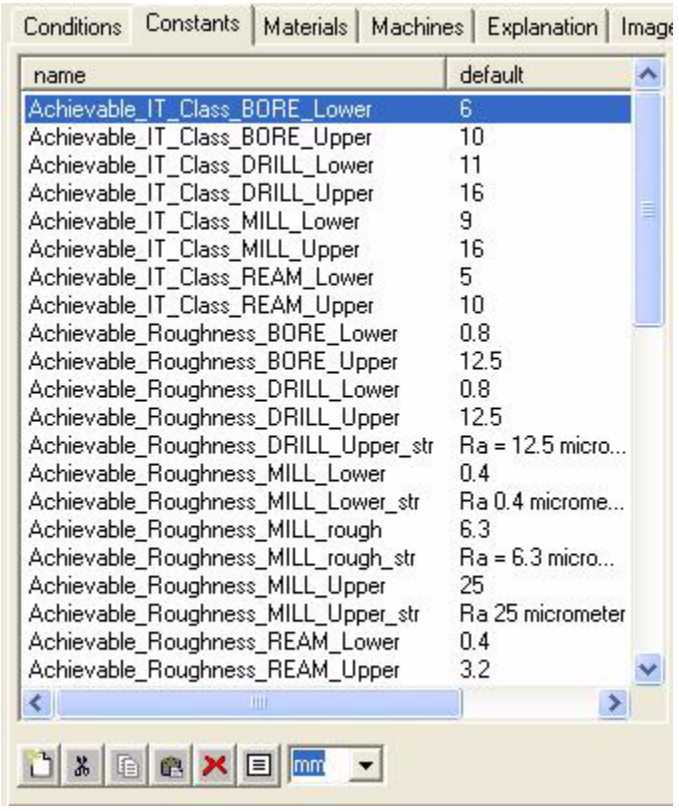
What do all these achievable... constants mean?

The majority of the constants we use are threshold values for the quality that can be obtained with a Drill, Mill, Reamer or Boring bar.

The two qualities checked in the rule's conditions are derived from the PMI and are the surface roughness and the dimensional tolerance.

For Drill, Mill, Reamer or Boring bar there are _upper and _lower threshold values of achievable surface roughness and achievable dimensional quality.
That gives 4 x (2 plus 2) = 16 constants. You see them in the screenshot below that

was taken from the Machining Knowledge Editor. The names of these constants are always something like '**Achievable...**'.



Roughing operations like Drill can be applied when these 2 conditions are both true:

```
IT_class_ISO(mwf.DIAMETER_1,mwf.DIAMETER_1_UPPER, mwf.DIAMETER_1_LOWER) >=
constant.Achievable_IT_Class_DRILL_Lower

roughness_value(mwf.SIDE_ROUGHNESS_1) >=
constant.Achievable_Roughness_DRILL_Lower
```

Finishing operations like Ream and Bore can be applied when either the roughness or the diameter tolerance is in the achievable interval for reaming and boring.

The expressions have a form like this and are found in the conditions of all finishing rules in the library:

```
IT_class_ISO(mwf.DIAMETER_1,mwf.DIAMETER_1_UPPER, mwf.DIAMETER_1_LOWER) >=
constant.Achievable_IT_Class_REAM_Lower
AND
(
roughness_value(mwf.SIDE_ROUGHNESS_1) >=
constant.Achievable_Roughness_REAM_Lower
AND
roughness_value(mwf.SIDE_ROUGHNESS_1) <=
constant.Achievable_Roughness_REAM_Upper
)

OR

roughness_value(mwf.SIDE_ROUGHNESS_1) >=
constant.Achievable_Roughness_REAM_Lower
AND
(
IT_class_ISO(mwf.DIAMETER_1,mwf.DIAMETER_1_UPPER, mwf.DIAMETER_1_LOWER) >=
constant.Achievable_IT_Class_REAM_Lower
AND
IT_class_ISO(mwf.DIAMETER_1,mwf.DIAMETER_1_UPPER, mwf.DIAMETER_1_LOWER) <=
constant.Achievable_IT_Class_REAM_Upper
)
```

Note Since NX 6.0.3 the roughness condition and IT class condition are logically OR. That implies that only one of them needs to be TRUE and the rule will be applied. It was a design choice not to restrict possible application of Rules too much. You may want to be more strict by changing the Roughness condition and the IT_class condition to be logically AND.

Exercise: Find conditions on the PMI in the rules.

- *Select a rule, for instance Drill_Up_S1H*
- *Press the Conditions Tab*
- *Find the section as depicted here*

```
roughness_value(mwf.SIDE_ROUGHNESS_1) >= constant.Achievable_Roughness_DRILL_
$$ Side Roughness too fine for DRILL
```

- *Find the expressions on the achievable IT_class in this rule.*

For Milling there is an extra threshold value for the achievable roughness. This defines the roughness value that requires finishing the surface.

Clearly, a feature's IT-class (usually defined on its Diameter) and its lowest surface roughness value determine the tools that are required.

All this information can be captured into a single table from which we can derive the required tool, given a combination of IT-class and Surface Roughness.

We made the assumption that Drilling is cheaper than Boring which is cheaper than Reaming. When in this table we read D,B,R the preferred solution is using a Drill, when that is not available, try a Boring bar and when that is not available look for a suitable Reamer.

TABLE 1. Operation (Tool) selection in order of priority for combinations of IT grade and Surface Roughness

Roughness Ra	IT 5	IT 6	IT 7	IT 8	IT 9	IT 10	IT 11	IT 12	IT 13	IT 14	IT 16
0.4 µm (16 µin)	R	R	R	R	R	R	R	R	R	R	R
0.8 (32) or 1.6 (63) or 3.2 (125)	R	B R	B R	B R	B R	B R	D B R	D B R	D B R	D B R	D B R
6.3 (250) or 12.5 (500)	R	B R	B R	B R	B R	B R	D B	D B	D B	D B	D B
25 (1000)	R	B R	B R	B R	B R	B R	D	D	D	D	D
50(2000)	R	B R	B R	B R	B R	B R	D	D	D	D	D

Exercise: Understanding the tolerance threshold values

- *Lookup the Achievable... constants in the Machining Knowledge Editor.*
- *What is the only tool that can machine a feature with a diameter tolerance = IT5?*
- *For which combinations of IT-class and surface roughness, all tools (Drill, Bore, Ream) could be used?*

Discussion of some other constants...

For a minimal explanation please see the Explanation field in the constant's Options dialog. (Select the RuleLibrary node, press the Constants tab and double click the constant.)

Modify the OOTB or Start from Scratch?

When is modification of the out of the box content the better option:

- You do **not** have User Defined Features **or** your UDF's can easily be mapped on the standard parametric feature types.
- You use the standard tool library that comes with NX or Teamcenter.
- Your processes are relatively simple and you are quite satisfied with the results that are generated ootb.

When is it better to start from scratch.

- Your company has defined its own best practices or is planning to do so.
- You think your processes are very different from what is supplied ootb.
- You do not want to invest in understanding the ootb content.

Workflow in a customization project

Each customization project consists of the following phases.

TABLE 2. Activities and Milestones in a FBM customization project.

Activity	Milestone
"Analyze your products for manufacturing features"	"Milestone: A list with all feature types"
	"Milestone: One or more reference models."
"Analyze your operation sequences"	"Milestone: List of Operation Sequences for all selected features."
"Analyze your tool libraries"	"Milestone: A list with all tool types"
"Implement Knowledge"	"Milestone: Implemented Rule Library"
"Maintain knowledge"	

Analyze your products for manufacturing features

Although it is theoretically possible to automate the complete operation generation of feature based products, it is often too costly to do so. We advise to start with the features most common within the typical product families.

The first step is to analyze which are typical products in your organization. This normally results in product families where you can describe how often products will be programmed, how many of the operations are standard within the products, etc. In general one of those product families should be chosen for a first project phase.

The next step will be to identify the common features within this product family. This should result in a list of common features with the percentage of occurrence within a product. This will normally result in a list of features where it also should be visible which feature types are the most common. In general the first project phase should include those feature types that cover 20% of the feature occurrences within the product family. In a typical second phase, this should be extended with those feature types which would extend this to 50% and in a third phase to 80%.

The last step in selecting the feature types is to check on some of the typical products, how the software will recognize or identify these features. It is no use trying to focus on feature types that are not recognized properly.

What comes OOTB

When you generate the customization automatically, the software takes into account:

- The parametric feature types which can be recognized by command ***Find Features***.
- The UDF library
- Feature_definition.def
- Feature_identification.def

The NX CAM help library includes a listing of the recognized parametric features. Browse to CAM > Feature-based machining > Manufacturing features > Parametric Feature Recognition.

For your convenience we reproduce (and updated) that list in the appendix. See “D: Definition of the standard non-STEPPED features” on page 154.

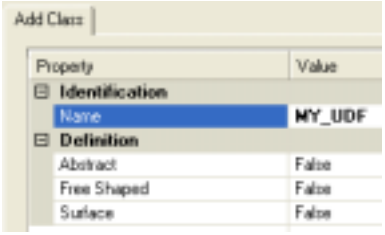
How to add features that are not standard.

Use an xml editor like xml spy (www.xmlspy.com) or any other which you find convenient.

In the following exercise we will demonstrate how to add MY_UDF to the customization, a new feature type which has 3 attributes of different types.

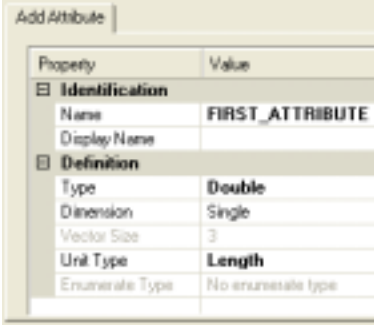
Exercise: Adding a User Defined Feature (UDF) to the customization

- *Open an existing xml source file with **File > Open...** or start a new file with **File > New...***
- *Choose **View > Customization***
- *Expand the Feature branch in the Tree View.*
- *MB3 on Customer Feature, choose **Add Class...***
- *Enter Name value MY_UDF, leave the other fields at their defaults and press*



OK.

- *With MY_UDF selected, choose **Add...** in the attribute window pane.*
- *Enter Name value FIRST_ATTRIBUTE, choose Type Double and Unit Type*



Length as shown.

- *Define a 2nd and 3rd attribute of type Integer resp String.*

Note The Display Name, when not specified, is equal to the Name.

- *Close the customization view.*

Exercise: Creating a Recognition Rule for the new UDF

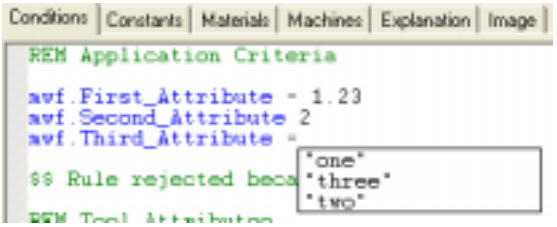
- *Press the Recognition tab to have access to the various Recognition rules.*

Note This manual is not (yet) a guide on how to define parametric feature recognition.

- *to be completed when -Prolog style- feature recognition becomes available.*

Exercise: Creating a Manufacturing Rule for the new UDF

- *Create a new rule with Output Feature = MY_UDF.*



Note Adding customer enumerated types like one,two,three as above is not anymore possible since NX 7.5

- *Playwise, enter conditions using the three UDF attributes as above.*

Milestone: A list with all feature types. It is the complete list of features you can expect in your production models. This should include a percentage of occurrences within the typical products.

Milestone: One or more reference models. The models have instances of the features types you can expect in your production models, including PMI. These models should be a mix of typical products and of specific test models designed to analyze problem areas.

Analyze your operation sequences

What comes OOTB

You need to invest time exploring the ootb rules. There are a few options:

- try them on your selected features.
- explore them in the Machining Knowledge Editor.
- explore them in the generated documentation which is obtained with MKE command ***Write Documentation***.

If you use operation sequences that are not standard ootb.

In fact that is more than likely. Analyze your operation sequences. For every feature type all possible operation sequences should be described. Although there is no single best way of doing this, it helps to use forms like “Sequence Form: Step2Hole” on page 132. This allows for a common sequence description.

Important is to identify why this sequence is chosen and not one of the other possible sequences, and if this sequence will change based on material or available resources. It is necessary to check if the decision criteria are based on feature attributes or on other factors. In the last case, it can be worth to add additional attributes to the features, which allow the user to define the choice, and to add tagging for these feature attributes to the system.

A next step in the analysis of your operation sequences is to create a table as in the example below. It lists all sequences identified in the previous step and enables you to identify common steps within the sequences.

produced feature >	Step1Hole(thread)				Step1Pocket (thread)							
In-Process Feature >	Sequence 1	Sequence 2	Sequence 3	Sequence 4	Sequence 1	Sequence 2	Sequence 3	Sequence 4	Sequence 5	Sequence 6	Sequence 7	Sequence 8
pocket_round_tapered												
SpotDrill	1	1	1		1	1	1		1	1	1	1
SpotDrill_in_S1P												
step1hole												
Drill_S1H												
Drill_in_center_S1H	2	2			2							
Drill_up_S1H	3											
Chamfer_SH1_Drill	4	3	4		3							
Chamfer_SH1_Mill	4	3	4		3							
Bore_S1H		4										
Ream_S1H		4										
Gun_Drill_S1H			3									
step1hole_thread												
Tap_S1H_thread					4							
step1pocket												
Drill_S1P												
Drill_in_center_S1P			2			2	2		2	2	2	2	2
Drill_up_S1P								3				
Spot_Face_S1P												
Mill_Rough_S1P_Contour						3	3					
Mill_Finish_S1P							5					
Chamfer_S1P_Mill						4	4		4	3	4	3	
Chamfer_S1P_Drill						4	4		4	3	4	3	
Chamfer_Fillet_Radius													3
Bore_S1P										4			4
Ream_S1P										4			4
Gun_Drill_S1P											3		

Horizontally you have the produced features. These are the ones obtained from the first milestone. And directly below you have a column for each anticipated sequence of operations that will finally produce that feature.

The first column holds the individual operations, sorted by the feature it produces, which can be an in-process feature.

The matrix is then filled by numbers which define the exact order in which the operations are performed.

Note, for instance, that sequence nr 3 which produces a STEP1HOLE is defined as:

```
Spot_Drill
  Drill_in_center_S1P
   Gun_Drill_S1H
     [Chamfer_S1H_Drill, Chamfer_S1H_Mill] (one of the two)
```

The following step is to describe the separate operations in more detail. An example could be as in “Step2hole operation” on page 133.

With all this information available, you must decide which operation sequences need to be implemented. It could be that some operation sequences are exotic and it is not worth to implement them.

Milestone: List of Operation Sequences for all selected features.

This should detail which operation sequences must be implemented.

Analyze your tool libraries

What comes OOTB.

The tool customization is based on the standard tool library that comes with NX, or with TeamCenter when you work in a managed environment.

How to add tools that are not standard.

- *When you need tool classes that are not standard you must add the tool class and then update the customization (only once). This is done automatically with MKE option “Update upon Load”.*

- *When adding a new tool class, tell NX which attribute is to be interpreted as the Cutter Diameter and Cutter Length. This is done with MB3 commands in the customization view.*
- *When you need tools of existing classes with dimensions that are not in the standard database you only need to add that tool to the library population.*
- *Please refer to existing NX training manuals and documentation about defining tool classes and populating them.*

Milestone: A list with all tool types. It has all the tool types you anticipate to use for your manufacturing features. These should match with the list of operations as chosen in the previous milestone.

Implement Knowledge

This section is just a collection of a few notes. But it's the bulk of the project.

In general, it is worth to consider, that the users always will need to define some operations manually. To speed up the process and assist the users, customized templates will be used for operations that are created frequently. It is wise to use these templates also in the knowledge definition. This results in uniformity between manually and automatically generated operations.

It is important to understand that other changes to the default software configuration can help in the success of the implementation. Examples are:

- Adding tagging data for features, which help the user to force the generation of certain sequences.
- Adding default program order groups, which help in generating an initial program sequence that will work without the user having to move operations.
- Adding KF post_create functionality to assist the user in putting the operations in the correct program order groups.

How to go about implementing the rules is treated in chapter "Machining Knowledge Editor Concepts" and chapter "Machining Knowledge Editor Exercises Part 1: Hole Making" .

Milestone: Implemented Rule Library.

Maintain Knowledge

Of course the manufacturing knowledge should be kept up-to-date. As best practices change also the knowledge should change accordingly. Regularly the knowledge should be checked against:

- How many manual changes need to be made before the operations generate correctly?
- How many operations are generated manually?

There should be a feedback loop from end-user(s) to the knowledge engineer.

CHAPTER 10

Appendices

A: Glossary of Terms

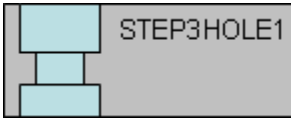
- Machining Rule: Captures 'best practices', describes the relation between manufacturing features, operations and tools. Often abbreviated to ‘Rule’.
- Machining Knowledge Source: Set of rules including the corresponding customization.
- Machining Knowledge Editor: Application to create and modify a Machining Knowledge Source.
- Machining Knowledge Library: Linked and compiled Machining Knowledge Source (the 'dll').
- Knowledge Engineer: Superuser who is able to modify the machining knowledge in the Machining Knowledge Source.

B: Naming conventions for Manufacturing Features

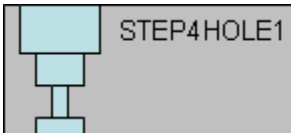
A few naming conventions apply to all standard features:

- HOLE features have no bottom, and can generally be reached from 2 sides.
- POCKET features have a bottom and can be machined from one side only.
- STEP1,2,3,4,5,6 features describe a single feature with a number of diameters.
- STEP**3**POCKET for instance is a blind hole with 3 diameters where the 1st diameter is larger than the 2nd and the 2nd diameter is larger than the 3rd.

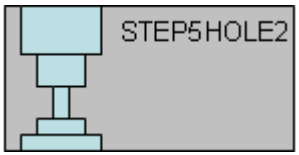
- STEP**3**HOLE**1** is a hole with 3 diameters where the 3rd is larger than the 2nd.



- STEP**4**HOLE**1** is a hole with 4 diameters where the 4th is larger than the 3rd.



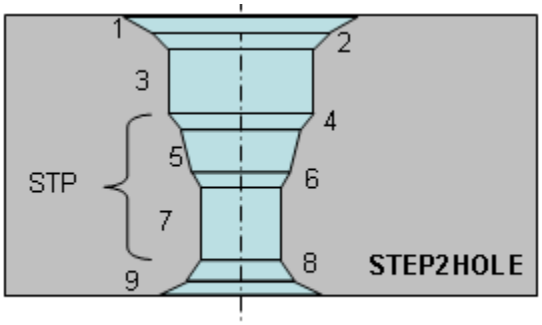
- STEP**5**HOLE**2** is a hole with 5 diameters where the 4th is larger than the 3rd and the 5th is larger than the 4th.



C: Definition of the standard STEPPED features

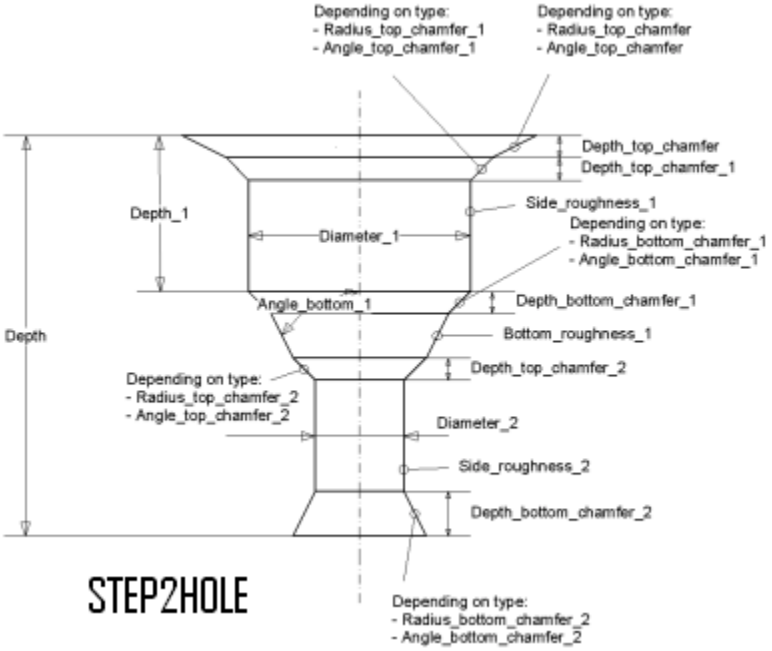
- Chamfer surfaces can be either torodial or conical
- Bottom surfaces are either planar or conical
- Chamfers are optional
- Bottom_ is optional if bottom_chamfer_1 or top_chamfer_2 is torodial

1. Top_chamfer
2. Top_chamfer_1
3. Cylinder_1
4. Bottom_chamfer_1
5. Bottom_1
6. Top_chamfer_2
7. Cylinder_2
8. Bottom_chamfer_2
9. Bottom_chamfer

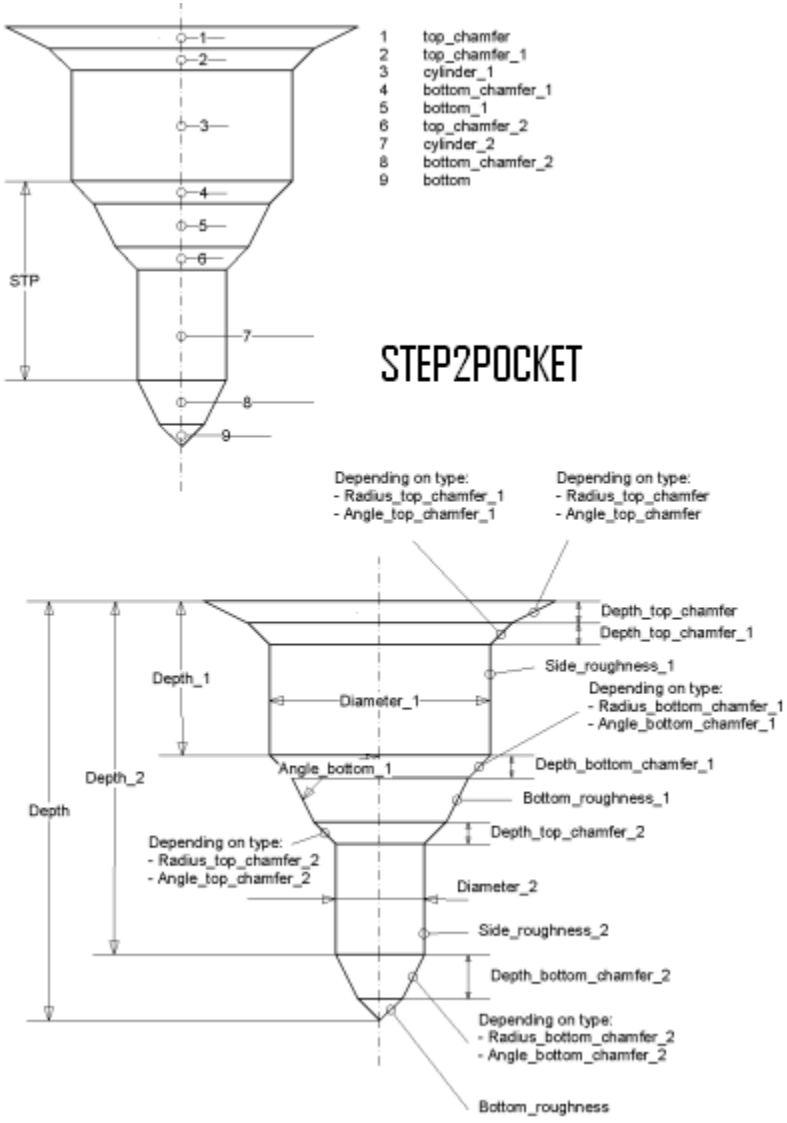


Picture above shows the most general constitution of a STEP2HOLE. For features with more STEPPED levels, section STP with elements 4,5,6,7 is repeated. Each added cylinder will get a higher index number.

Picture below show the **parameters** of the most general constitution of a STEP2HOLE.



Picture below shows constituent faces and parameters of the most general form of a STEP2POCKET.



•

D: Definition of the standard non-STEPPED features

This is a list of standard recognized parametric manufacturing features.

TABLE 1. Feature Types occurrence

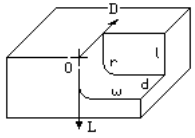
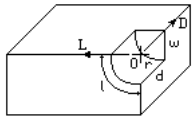
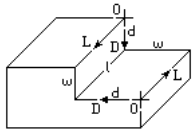
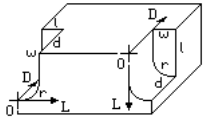
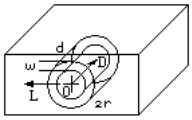
Feature	Type name	Notes
	CORNER_ NOTCH_ RECTANGULAR	
	CORNER_ NOTCH_ ROUND_ CONCAVE	
	CORNER_ NOTCH_ STRAIGHT	
	CORNER_ NOTCH_ U_ SHAPED	
	GROOVE_ AX_ CIRCULAR_ RECT	

TABLE 1. Feature Types occurrence

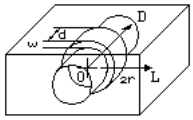
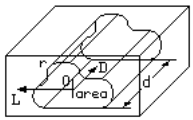
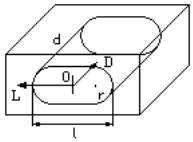
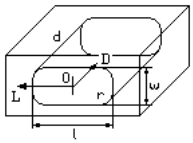
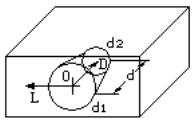
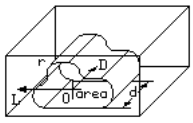
Feature	Type name	Notes
	GROOVE_ INS_ RAD_ RECT	
	HOLE_ FREE_SHAPED_ STRAIGHT	
	HOLE_ OBROUND_ STRAIGHT	
	HOLE_ RECTANGULAR_ STRAIGHT	
	HOLE_ ROUND_ TAPERED	
	POCKET_ FREE_SHAPED_ STRAIGHT	

TABLE 1. Feature Types occurrence

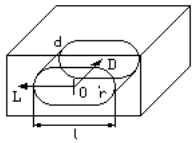
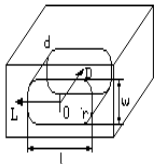
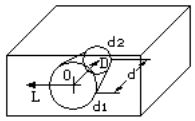
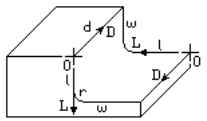
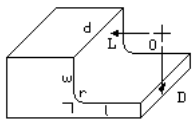
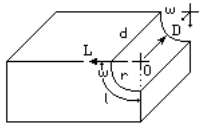
Feature	Type name	Notes
	POCKET_ OBROUND_ STRAIGHT	
	POCKET_ RECTANGULAR_ STRAIGHT	
	POCKET_ ROUND_ TAPERED	
	SIDE_ NOTCH_ RECTANGULAR_1	
	SIDE_ NOTCH_ RECTANGULAR_2	
	SIDE_ NOTCH_ ROUND_ CONCAVE_1	

TABLE 1. Feature Types occurrence

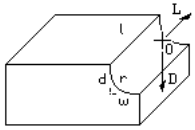
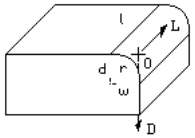
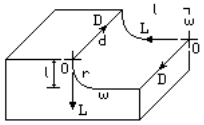
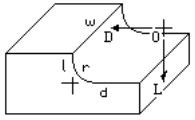
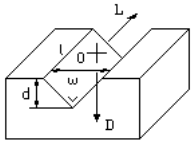
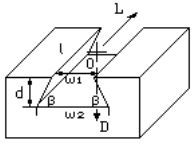
Feature	Type name	Notes
	SIDE_ NOTCH_ ROUND_ CONCAVE_2	
	SIDE_ NOTCH_ ROUND_ CONVEX_2	
	SIDE_ NOTCH_ U_SHAPED_1	
	SIDE_ NOTCH_ U_SHAPED_2	
	SLOT_ 90_DEGREE	
	SLOT_ DOVE_TAIL	

TABLE 1. Feature Types occurrence

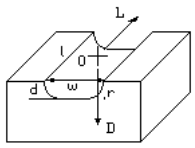
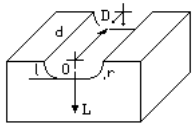
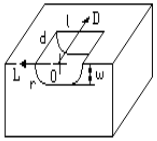
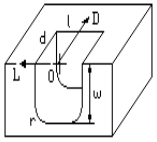
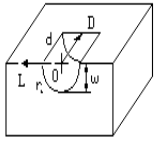
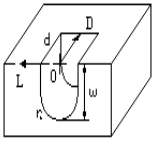
Feature	Type name	Notes
	SLOT_ OBROUND_1	
	SLOT_ OBROUND_2	
	SLOT_ PARTIAL_ OBROUND	
	SLOT_ PARTIAL_ RECTANGULAR	
	SLOT_ PARTIAL_ ROUND	
	SLOT_ PARTIAL_ U_SHAPED	

TABLE 1. Feature Types occurrence

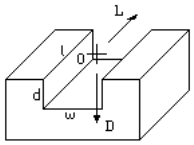
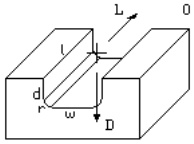
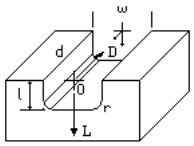
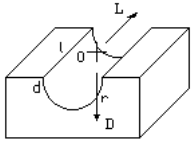
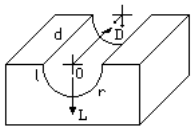
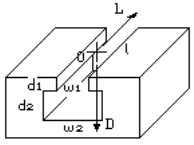
Feature	Type name	Notes
	SLOT_ RECTANGULAR_1	
	SLOT_ RECTANGULAR_2	
	SLOT_ RECTANGULAR_3	
	SLOT_ ROUND_1	
	SLOT_ ROUND_2	
	SLOT_ T_SHAPED	

TABLE 1. Feature Types occurrence

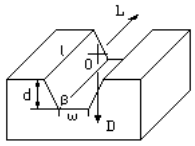
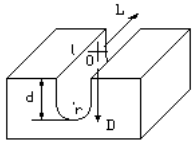
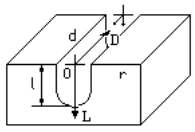
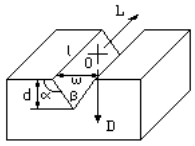
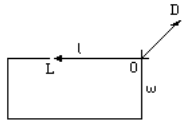
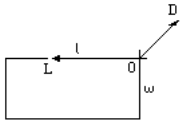
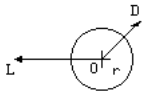
Feature	Type name	Notes
	SLOT_ UPSIDE_DOWN_ DOVE_TAIL	
	SLOT_ U_SHAPED_1	
	SLOT_ U_SHAPED_2	
	SLOT_ V_SHAPED	
	SURFACE_ PLANAR_ RECTANGULAR	

TABLE 1. Feature Types occurrence

Feature	Type name	Notes
	SURFACE_ PLANAR	Any flat surface, not necessarily rectangular.
	SURFACE_ PLANAR_ ROUND	

E: Tips and Tricks

Choosing priority for the MACHINING_RULE

The MachiningRule.priority is an attribute which determines the sequence in which MachiningRules are tried, where a higher value means that the rule is tried earlier. The value has a meaning within a set of rules that have the same type of More Worked Feature. When, for instance, rule 1 machines a step1hole and rule 2 machines a step1pocket, the relative priorities are not important, since these rules are not competing. If rule 1 and rule 2 both machine a step1hole, the rule with higher priority will be tried first and only if it is rejected rule 2 will be tried.

Generally it is good practice to give the cheaper rule a higher priority. Conditions on the cheaper rule must then be designed such that it is applied only when that is allowed. This is typically customer specific knowledge, allthough a general framework comes with the ootb knowledge.

How to use the MACHINING_RULE conditions

The rules contain conditions like:

mwf.MACHINING_RULE = "TWIST_DRILL"

or

lwf.MACHINING_RULE = "TWIST_DRILL"

These conditions are used to create a certain sequence of operations. An example is the sequence where tapping is always preceded by a drilling operation. In this case the tapping rule (for example ‘Tap_S1H_thread’) can contain the condition:

lwf.MACHINING_RULE = "TWIST_DRILL"

and the drilling rule (for example ‘Drill_S1H’) contains:

mwf.MACHINING_RULE = "TWIST_DRILL"

This prevents other rules are used. For example the boring rules contain the condition:

mwf.MACHINING_RULE = "BORE"

For this feature, the boring rules will be rejected when the Tapping rule was applied before. (Of course the operation sequence in time is the reverse.)

This condition is not causing conflicts when the drilling operation should be the first operation. This is because the attribute MACHINING_RULE is normally not defined for recognized features so it will not exists on the feature and the condition on MACHINING_RULE is not applicable.

An exception would be if a face attribute MACHINING_RULE were defined on the model. In that case the feature recognition will define the attribute on the recognized feature and as a consequence the conditions containing the mwf.MACHINING_RULE will not be skipped. You can use face attributes to influence the sequence of operations when not enough data is available to choose between certain sequences or if a specific sequence is required.