Virtual tool

Software documentation

OPEN MIND
THE CAM FORCE

OPEN MIND Technologies AG

Argelsrieder Feld 5

82234 Wessling

Germany

Tel.: (+49-8153) 933-500

Fax: (+49-8153) 933-501

E-mail: `<sales.europe@openmind-tech.com>`

Web: www.openmind-tech.com

Table of Contents

1. Variables

Description of the available variables.

Connector parameters: These variables always reflect the current value in the job.

Example:

In the case of a sink hole, the diameter of the hole or the sink defined in the job is output. These values can also be changed during the transfer.

Tool parameters: Here there are variables for NC tool, tool, holder, extension, material. The values correspond to the values of the tool UI.

Job parameters: The parameters in the job can be read and written during the transfer.

Job list: The material to be used, the selected postprocessor and the machine coupling can be read from the current job list.

Feature variables: Values that are not provided in the connector can be read directly from the feature.

User variables: The user variables defined in the job list can be read and generated. A distinction is made between local and global user variables. Local user variables are only available only during the transfer of the macro. They are used to transfer data between individual working steps in the macro. These variables are deleted after the transfer. Global user variables are retained even after the transfer.

2. VT format

The instructions are described in the VT editor. This generates an XML file from the instructions. Five procedures are available in the editor to describe the tool to be searched for.

3. Procedures

The user interface of the Virtual Tool Editor is divided into tabs. Each of these tabs indicates a step in the tool search process. This is also known as a procedure.

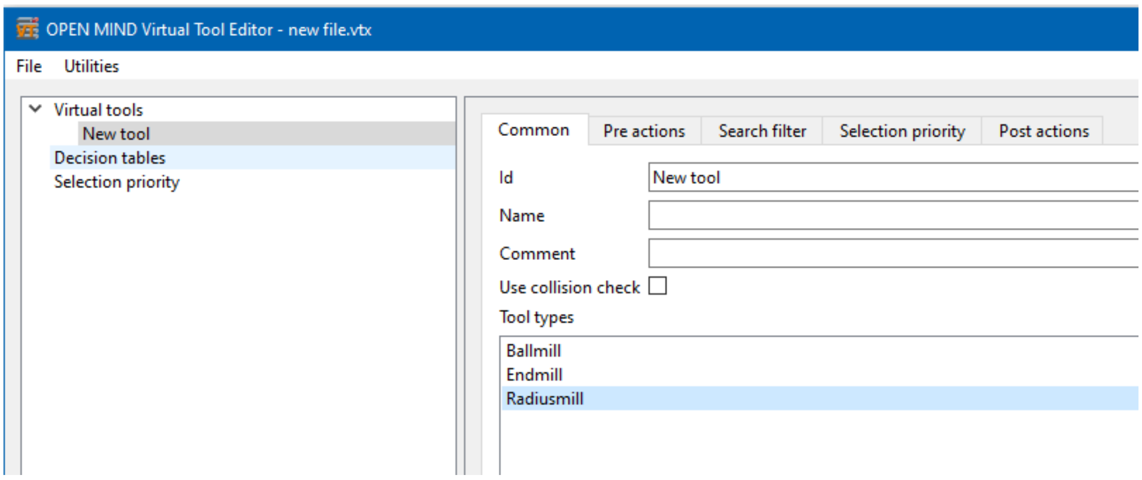
Common

The tool type is defined in this procedure. This is the first selection in the database.

Example: If a drill tool is defined, all other tools are excluded from the search. It is possible to define several tool types.

The tool types are then with a **OR**- Link connected. The tool type must then either be a ballmill **OR** an endmill **OR** a radiusmill.

The **ID** generated in the VT editor is displayed in the macro database in the macro jobs area as a selection option (**Tool ID**). This **ID** must be unique and can no longer be used for other tool descriptions. Name and comment can be freely defined.



Id: New tool
Name:
Comment:
Use collision check: ☐
Tool types:
Ballmill
Endmill
Radiusmill

With the option **Use collision check** the NC tool is checked for possible collisions with the part during the tool search. The collision check is only possible when machining holes, since the points and the machining depth to be checked are known. For milling operations, each collision check would require first calculating the NC paths and then checking each NC point. This is not possible for performance reasons.

The structure of the search and definition rules is always identical. The type is defined in the first column. There are different types which are used in different procedures.

Type	Available procedures
Set user variable	Pre action - Post action
Set job cfg	Pre action - Post action
Decision table	Pre action - Search filter - Post action
Set connector	Pre action - Post action
Set global user variable	Pre action - Post action
Sequence	Selection priority

Type	Available procedures
Condition	Selection priority
Min	Selection priority
Max	Selection priority

The second column defines the condition. This defines when the action is to be executed. The left expression is always checked against the right expression.

Condition	Comment
Connector.Diameter = 10	The query is executed only if the hole diameter is equal to 10mm.
Connector.Diameter > 10	The query is executed only if the drill diameter is greater than 10mm.
Connector.Diameter < 10	The query is executed only if the drill diameter is smaller than 10mm.
Joblist.Material like ‚Aluminium‘	The query is executed only if the material Aluminum is selected in the job list.
Joblist.Material != UNKNOWN	The query is only executed if the material is unequal UN-KNOWN, i.e. known. If no material is selected in the job list, the variable is returned with UNKOWN.

Pre action

This procedure is used for calculations, parameter assignments or calling decision tables.

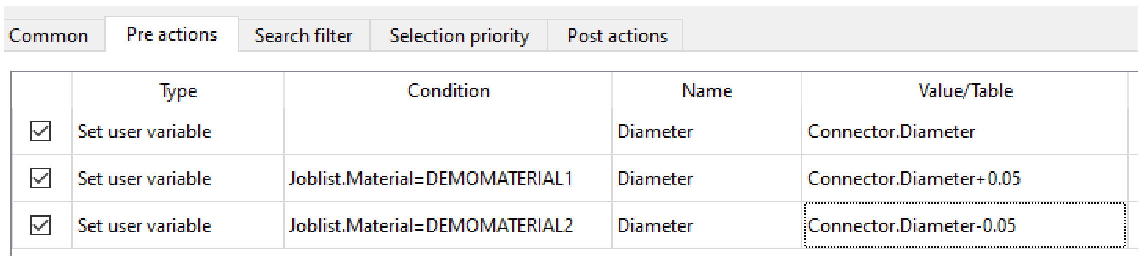
There are manufacturing situations where required tool parameters must must have values other than those provided by the connector or feature.

To calculate these values, it is recommended to create a pre action. Alternatively, the calculation of the tool can also be done in the procedure **Search filter**, but this can make it confusing within this procedure.

Example: Depending on the selected machining material, the drill tool should be 0.05mm smaller or larger. To search for a suitable tool, the **Diameter** user variable is created. Depending on the material, the diameter from the connector +- 0.05mm is assigned to this variable in order to search for a suitable tool.

More complex calculations must be performed step by step. The mathematical rule point calculation takes precedence over line calculation is not taken into account in the VT Editor.

In the following example, two materials are defined for the decision. If several such queries have to be defined, it is recommended to use a **Decision table**.



Common | Pre actions | Search filter | Selection priority | Post actions

	Type	Condition	Name	Value/Table
☑	Set user variable		Diameter	Connector.Diameter
☑	Set user variable	Joblist.Material=DEMOMATERIAL1	Diameter	Connector.Diameter+0.05
☑	Set user variable	Joblist.Material=DEMOMATERIAL2	Diameter	Connector.Diameter-0.05

This topic is explained in the section Decision table (page 11).

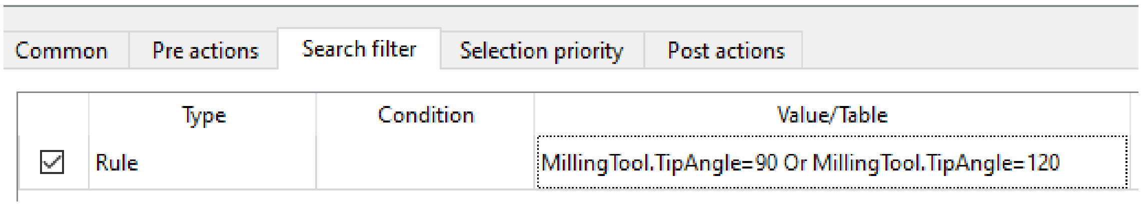
Search filter

The actual tool search is performed in this procedure. The search rules defined here are sent to the tool database as an SQL query. Basically, the entire tool database is always searched.

The query rule must first contain the tool parameter, followed by the comparison operator (=, <>, like) and finally the value that the tool should have.

All rules are sent to the database with an **AND** link. If a query is to be linked with an **OR**, this must be written in one line.

Example:



Common | Pre actions | Search filter | Selection priority | Post actions

	Type	Condition	Value/Table
☑	Rule		MillingTool.TipAngle=90 Or MillingTool.TipAngle=120

Here you are looking for a tool, that has a tip angle of 90 or 120 degrees.

As described in the **Pre action** section, more complex calculations or multiple assignments should be performed in the pre action.

Numerical values can be queried with =, <>, >!, <!. Alphanumeric values such as tool name, tool folder, couplings, etc. can be queried with = or LIKE. If the equal sign is used, the expression searched for must be exactly correct.

Example: If a tool has the NC name **Reibahle H7 Din 8093** it must be queried with

NCTool.Name = ‚Reibahle H7 Din 8093‘

or with

NCTool.Name like ‚%H7%‘

In the first query only the tool, that has exactly this name will be found. With the second query, every tool is found that contains an 'H7' in its name. The **%** character is used in SQL as a wildcard. In this case, it does not matter which characters are specified before and after the H7. Texts should always be specified in quotation marks ' '.

Selection priority

This procedure defines which of the found tools should be used. With the search rules from the procedure **Search filter** often several tools are found. The more precisely the search rules are defined, the fewer tools will be found.

If only the tool type **Ball Mill** with diameter 10 is defined, all ball mill cutters with D10 will be found in the database.

It is not recommended to define the search filters so precisely, that only one tool is found. In most cases, this is also not possible. In order to select the right tool from the list, different priorities can be set.

For the search filter, it does not matter, whether the tool type is defined first and then the diameter, tool length, etc.. All specified search criteria are used together.

The situation is different with the selection priority. Here the criteria are processed in the order, in which they were defined. The following parameters are available as decision criteria.

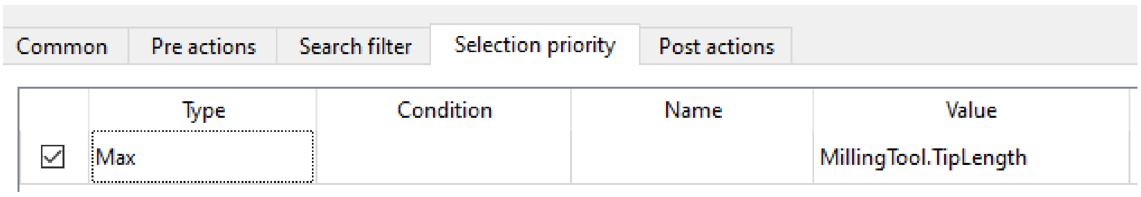
MIN: This criterion is used to select from the tool list the tool for which the smallest value has been defined for this parameter.

Example:

If you have searched for tools with a diameter of +- 0.5 mm, the tool with the smallest diameter will be used.

MAX: Is the opposite of MIN. In this case, the tool with the largest value defined for this parameter would be used.

Example:

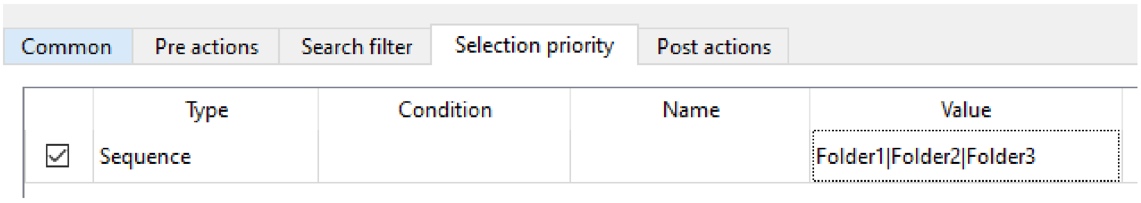


	Type	Condition	Name	Value
☑	Max			MillingTool.TipLength

The tool with the longest cutting length is used.

SEQUENCE: Several values can be defined here, which are used in the specified order. The parameters are separated with a |.

Example:

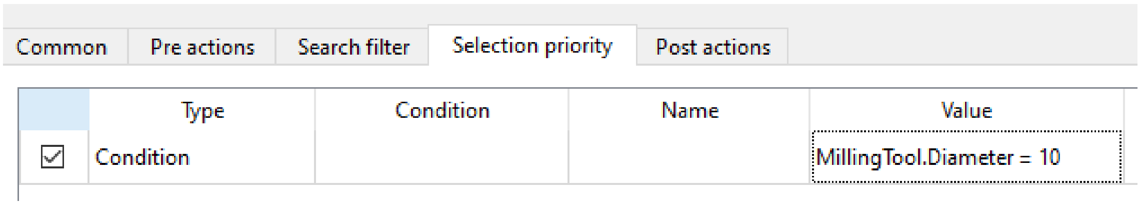


	Type	Condition	Name	Value
☑	Sequence			Folder1|Folder2|Folder3

The tool located in folder1 is used first. If there are no tools in folder1, then folder2 and then folder3 are searched. If there is no tool in one of these three folders, all tools are kept.

CONDIDITION: Here a fixed value can be defined for a parameter.

Example



	Type	Condition	Name	Value
☑	Condition			MillingTool.Diameter = 10

If a tool with a diameter of 10mm +-0.5mm was searched for, it can be defined here that always the tool with exactly the diameter 10mm is used, even if tools with smaller or larger diameter were found.

When applying the selection priority, it is important to note, that the order of prioritization plays a crucial role. After each of the specified lines, the tools, that do not meet these conditions are deleted from the list of found tools. If none of the conditions are met, all the tools are kept.

Example:

The following tools were found

1 Drill tool D9.90 Length 100mm Folder3

2 Drill tool D10.0 Length 110mm Folder2

3 Drill tool D10.1 Length 90mm Folder2

If you prioritize first according to the minimum diameter, tool 1 is retained. If prioritized by the shortest tool first, tool 3 is retained. If the order of the folders is prioritized first, tools 2 and 3 are retained.

Only so many tools are deleted from the list that at least one tool remains at the end of the search. If several tools are still available at the end, the first tool from the list is used.

Post action

After the tool search and prioritization, the tool found for the work step is output. As the last procedure, various actions can then be defined in the postprocessing action. Even if no tool is available in the database for the defined search parameters, a final action can still be performed here.

Found actions

Found actions apply in the case, that tools are available and are used for the working step.

Not found actions

Not found actions apply, if no suitable tool has been found.

Finally, user variables, job cfg parameters or connector parameters can be set. In addition, a **decision table** can also be called here, which can set one of the parameters.

User variable / Global user variable: These variables can be used for calculations or for the exchange of information between the individual working steps. The tool search is carried out within a macro from bottom to top, so for the last working step within a macro the search is carried out first. Depending on which tool was found, this can also change the tool in a previous job.

Example: Macro for centering - drilling - tapping

First, the tool for tapping is searched for. During the tool search, a tap and a forming are found. By prioritizing, the tap is preferred. Via the user variable, the information that the thread is being formed is transferred from the tapping cycle to the drilling cycle. In this way, the appropriate drill tool can be searched in the drilling step.

User variables are deleted again after the macro transfer. Global user variables are retained after the macro transfer. In this way, information can also be exchanged between different macros.

Job cfg parameter: When transferring the working steps from a macro, each parameter can be changed in one work step.

Example:

When drilling, depending on whether the hole is a blind hole or a through hole, the break through length can be rewritten or the tip angle compensation can be switched on.

If no tool is found, the search parameters for the searched tool can be stored in the tool comment. This allows you to quickly and easily determine which tool is required.

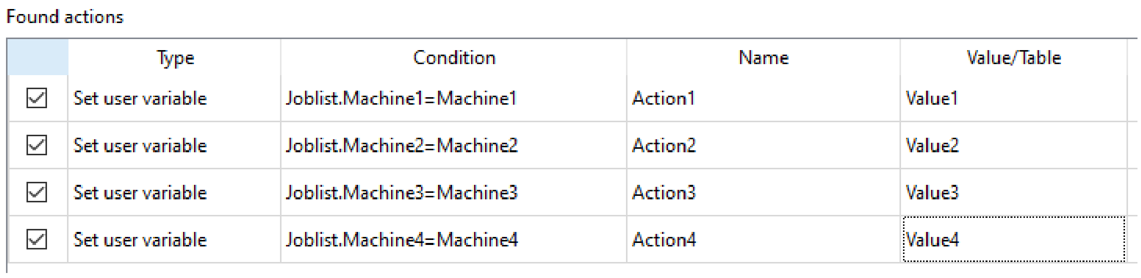
Set connector: This function can be used to overwrite values specified in the working step in the Feature Connector or to activate or deactivate individual drilling steps.

Example:

A hole should be predrilled with a shorter tool for predrilling. The machining depth should not be carried out to the entire depth of the hole. The shorter tool should first be used to predrill to the maximum depth possible with this tool.

4. Decision table

All parameters can also be set with the decision table. The decision table is mainly used if different machines, materials or other production-relevant situations lead to the fact that several possibilities can result for a parameter. In most cases it is then easier and clearer to set these different possibilities not by the **condition** within the definition, but in a **decision table**.

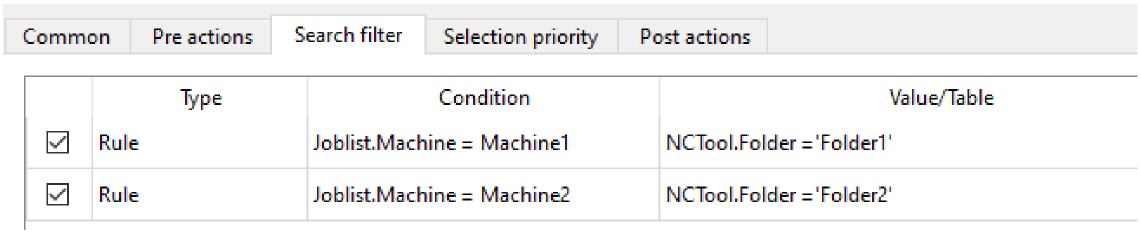

Found actions

	Type	Condition	Name	Value/Table
✓	Set user variable	Joblist.Machine1=Machine1	Action1	Value1
✓	Set user variable	Joblist.Machine2=Machine2	Action2	Value2
✓	Set user variable	Joblist.Machine3=Machine3	Action3	Value3
✓	Set user variable	Joblist.Machine4=Machine4	Action4	Value4

The decision table allows to define simple tables in which different criteria can be defined to find the suitable tools.

Example:

Depending on the NC machine used, the tools are to be searched in different folders. One possibility is to define a separate query for each machine and to query the individual machines in the conditions.



	Type	Condition	Value/Table
✓	Rule	Joblist.Machine = Machine1	NCTool.Folder ='Folder1'
✓	Rule	Joblist.Machine = Machine2	NCTool.Folder ='Folder2'

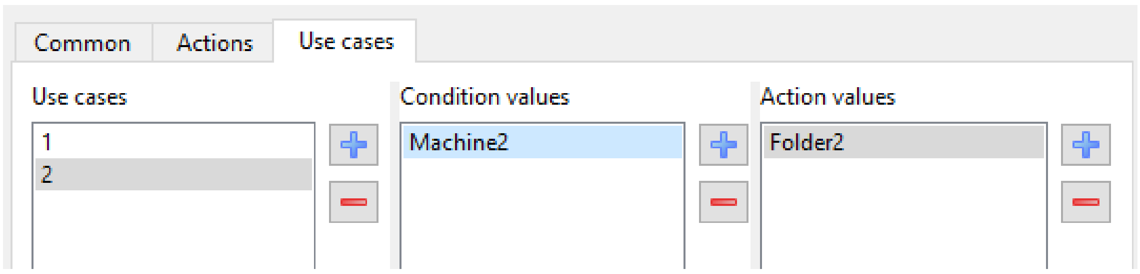
This solution is acceptable if only two or three decisions are to be made. With several machines in this example, the search filter quickly becomes confusing due to the number of different rules. Here, the clarity as well as the data maintenance can be significantly improved by the decision table.

The first thing to do is to create a decision table. This is done in the VT Editor under **Decision table** . This is where the recommended working method differs, because the decision table is usually working "backwards". The use cases are therefore defined first.

Use cases

Use case: 1	Use case: 2
Condition value: Machine 1	Condition value: Machine 2
Action value: Folder 1	Action value: Folder 2

The conditions are addressed for later programming as **C1** - **CN**, the action values as **A1** - **AN**, derived from the English terms **Condition** and **Action**. Several conditions and action values can also be defined for each application.


Common | Actions | Use cases

Use cases: 1, 2

Condition values: Machine2

Action values: Folder2

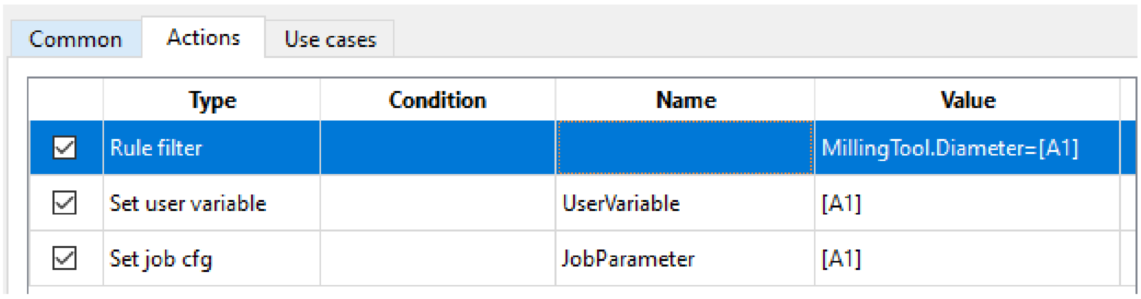
Actions

The **Actions** tab now defines how the parameters from the use cases are used.

Example:

If a certain machine is defined in the job list, of which it is known that the spindle holder is created as **HSK 63** and if NC tools have been created for this machine in the tool database in corresponding folders that contain the designation **HSK 63** , then only in these folders the corresponding NC tool can be searched for using the decision table.

Also **Rule filter**, **User variables** or **Job cfg parameters** can be generated.


Common | Actions | Use cases

	Type	Condition	Name	Value
✓	Rule filter			MillingTool.Diameter=[A1]
✓	Set user variable		UserVariable	[A1]
✓	Set job cfg		JobParameter	[A1]

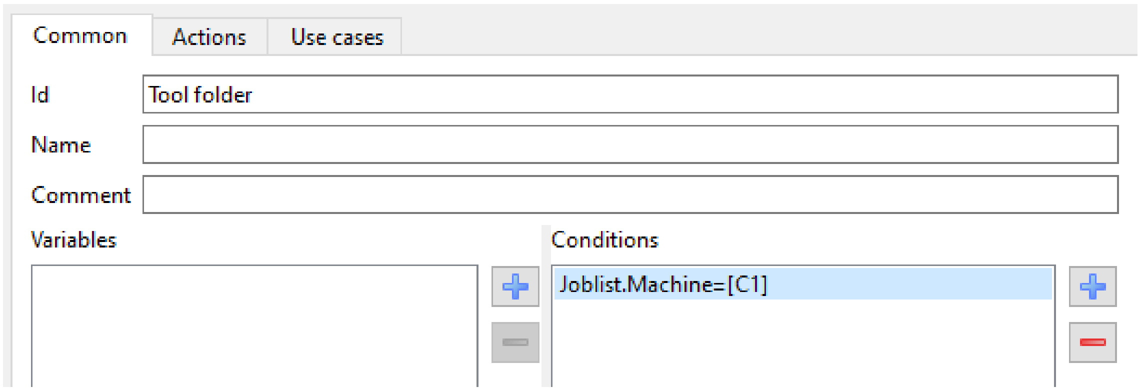
The definition is created in the same way as in the section for the tool definition. Here, however, the parameters from use cases **A1** - **AN** are used, instead of CONNECTOR parameters or user variables.

In the **Common** tab, it is defined, which use case is applied.

Common

First, the **ID** of the decision table is specified. With this **ID** the decision table is called up in the search filter. **Name** and **comment** can be freely assigned and are used as additional information.

A parameter from the **Connector**, **Job list**, **Feature** or **User variable** can be used here.


Common | Actions | Use cases

Id	Tool folder
Name	
Comment	

Variables:

Conditions: Joblist.Machine=[C1]

In the use cases it is searched in which use case the machine name from the job list (postprocessor name) is defined.

If the machine name = **Machine 1**, use case 1 is used.

If the machine name = **Machine 2**, use case 2 is used.

Depending on the use case, the internal parameter **A1** is filled with the value defined in this use case and inserted in the lines of this use case.

All rows defined here are then inserted at the position from which the decision table was called.

Decision tables can be called from the procedures **Pre action**, **Search filter** and **Post action**.

If the Decision table is called from the **Pre action** or **Post action** function, only user variables or job cfg parameters can be defined in the decision table. If the decision table is called from the **Search filter**, only a search filter can be defined.

Any number of **Search filter**, **User variables** or **Job cfg parameters** can be defined.

Several condition values and action values can be defined in the **Use cases**, which are then respectively identified with **C1**, **C2** ... .. **CN** and queried with **A1**, **A2**... .. **AN**.

A **Decision table** can be called from all VIRTUAL Tool definitions.

The use case table can be modified and extended as needed.

If, for example, a new machine is required, a third use case can simply be defined for machine 3. This is then automatically available in all tools that are using this decision table. In this way, the decision tables can also be used subsequently to extend already existing virtual tool definitions.