Manufacturing Management Software

Method Editor

Making IT Work

Save Time

Save Money

Improve Performance

Comprehenisve - Proven - Affordable

www.match-it.com

Smart
Funding Innovation

Table of Contents

Using the Method Editor

This section describes what methods are, their significance and how to create them.

1 What is a method?

The term *method* refers to the information Match-IT holds about how you manufacture your products or perform your services.

This includes *bill-of-material* information (what materials are required) as well as *routing* information (what processes are required) and much more.

Methods can be completely self-contained and just describe how to make a simple ‘widget’, or they can be combined in complex ways to describe huge structures with 100’s or 1000’s of parts and operations.

Overall, a method consists of a series of *STEP*s. A STEP represents a single operation. Each STEP in the method must be performed in sequence to create the end result.

Each STEP can consist of any number of *PART*s and any number of *RESOURCE*s. A STEP can have other things associated with it too; we’ll introduce those later.

A PART is a reference to a quantity of a material that is required for that STEP. When there are multiple PARTs, it can either mean all are needed or it may mean there are choices.

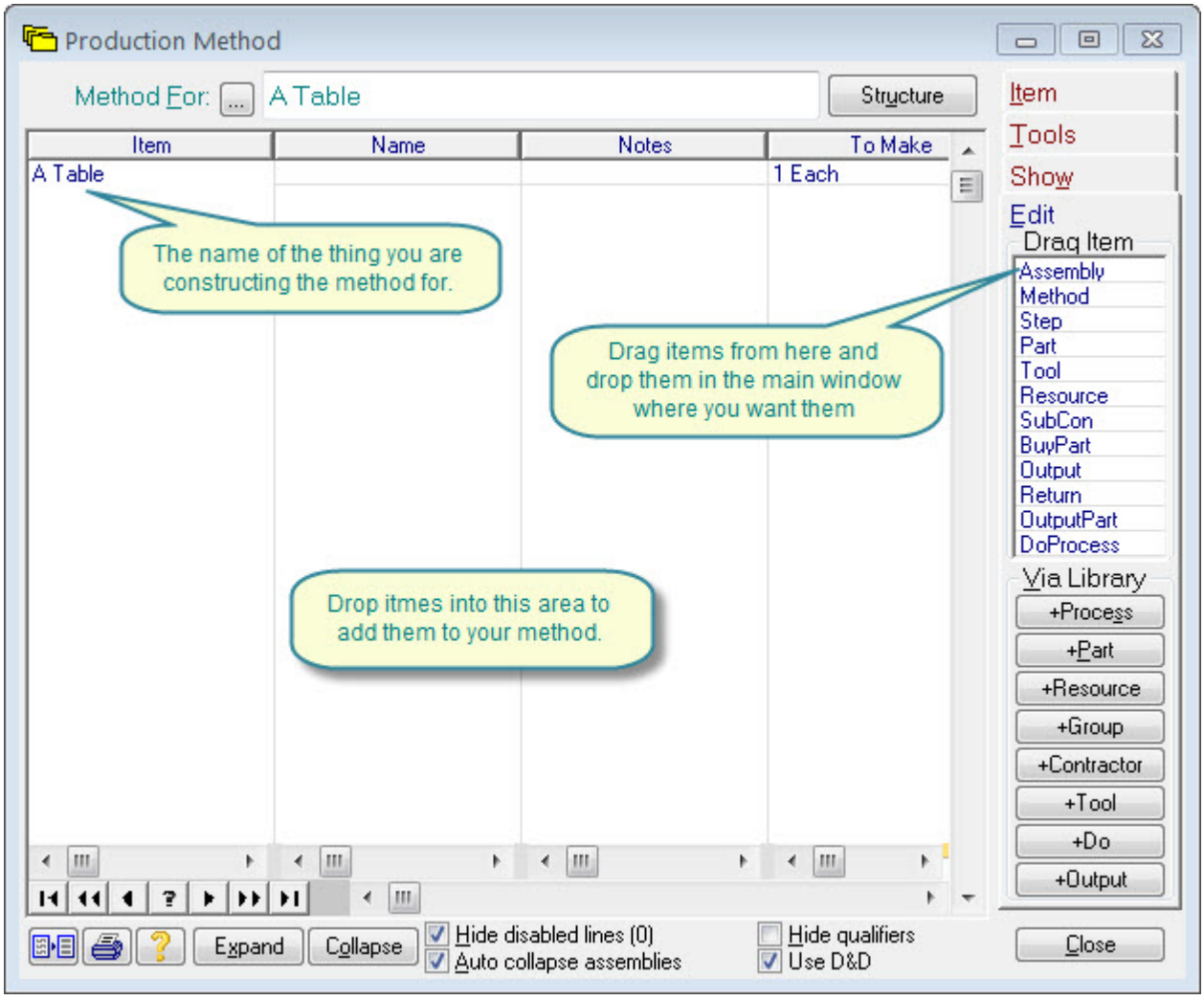
A RESOURCE is a reference to a machine or a person that is required for a period of time to perform the process required for that STEP. When there are multiple RESOURCEs, it can either mean all are needed at the same time, or it can indicate there are choices.

This basic structure can be depicted like this:

```
Method (make)
      Step 1
            Part A
            Part B
            …
            Part N
            Resource A
            Resource B
            …
            Resource N
      Step 2
            Part …
            Resource …
      …
      Step N
            Part …
            Resource …
      Method (end)
```

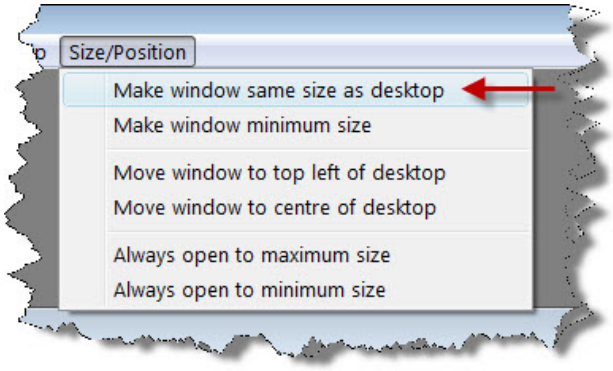

2 How do I create a method?

You create methods, or modify existing ones, using the *Method Editor*. You can access the editor from any material record by pressing Method. If you do this on a new material record you'll see something similar to this:



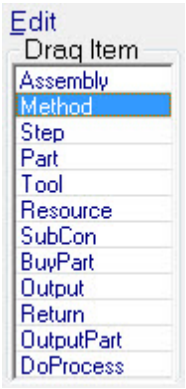
The main area will initially be empty except for a line showing the name of the product, this area is where the method is shown. The buttons and tabs on the right provide a variety of tools to help you construct methods. A description of each of these tools is given later. For now, we'll use a few of them to construct a very simple method. We will construct a method that describes how to make a simple table, consisting of 4 legs, 4 feet and a top.

You will benefit from making the method editor window as big as you can, do this by pressing `Size/Position` then `Make window same size as desktop` as shown here.

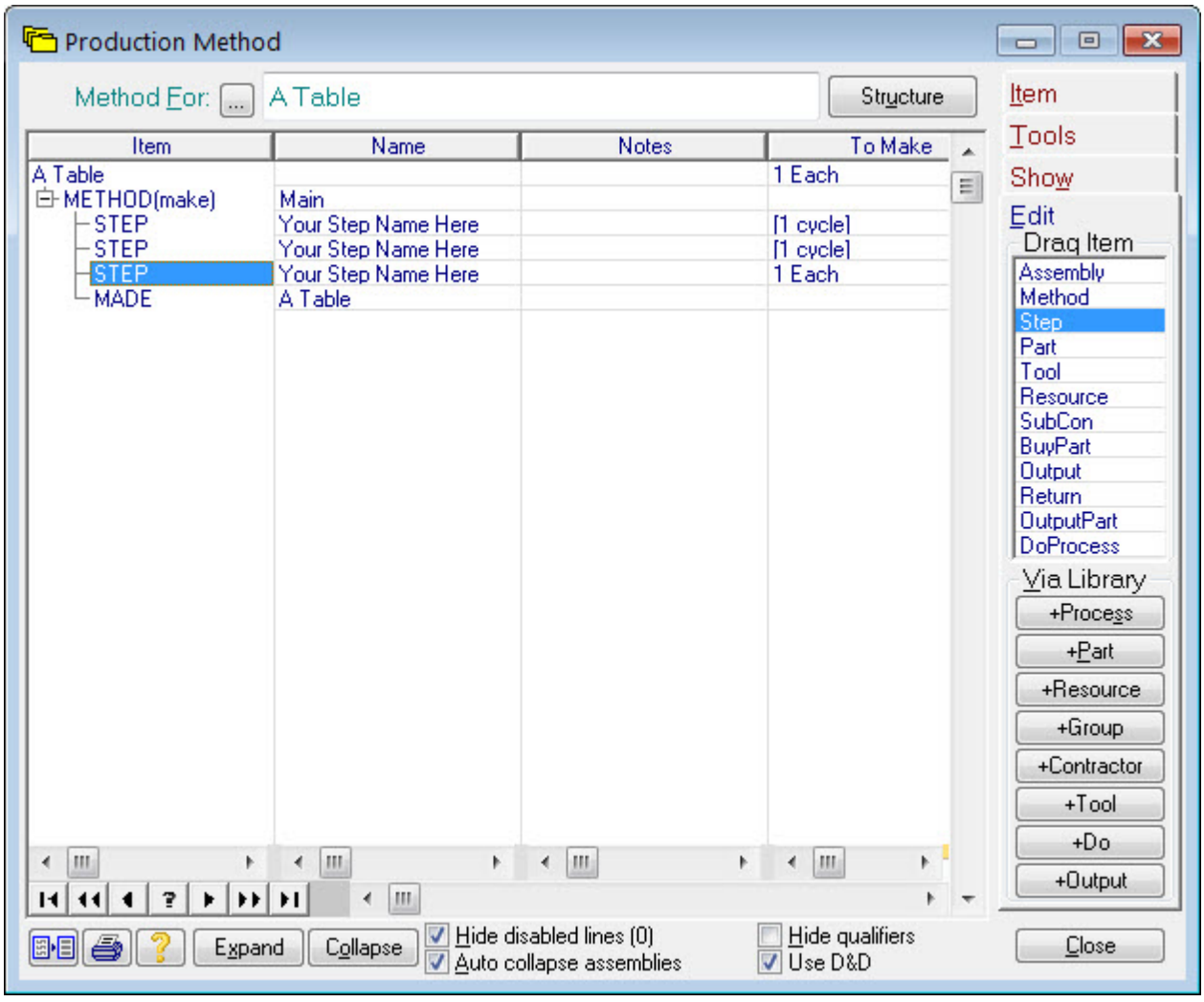


The process we will describe has three operations; first we cut the legs, then we cut the top, then we assemble it. Create a new material record called `a table`, mark it as a manufactured item, then press its Method to open the method editor.

Start by dragging a `Method` item into the method information area, then drag three `Step` items below it. Use the `Drag Item` tool on the `Edit` tab to do this (as right).

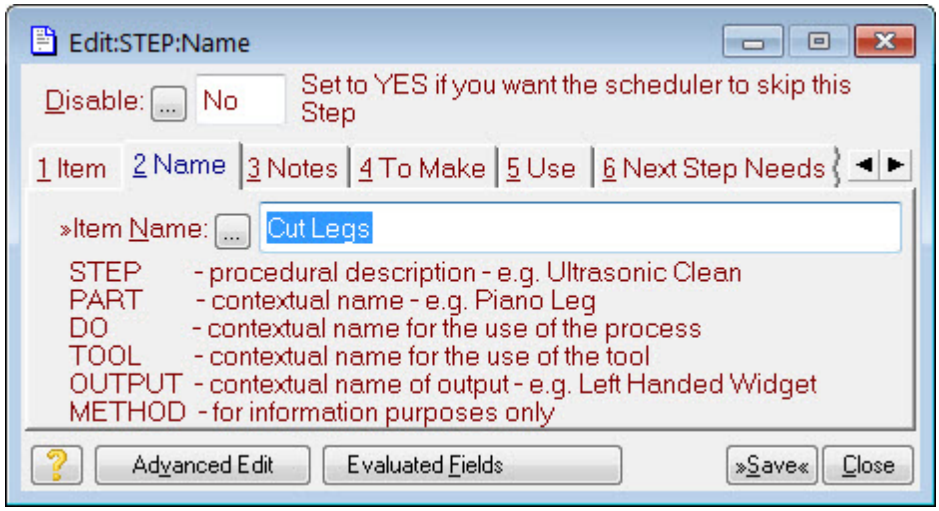


Your method should now look similar to this:

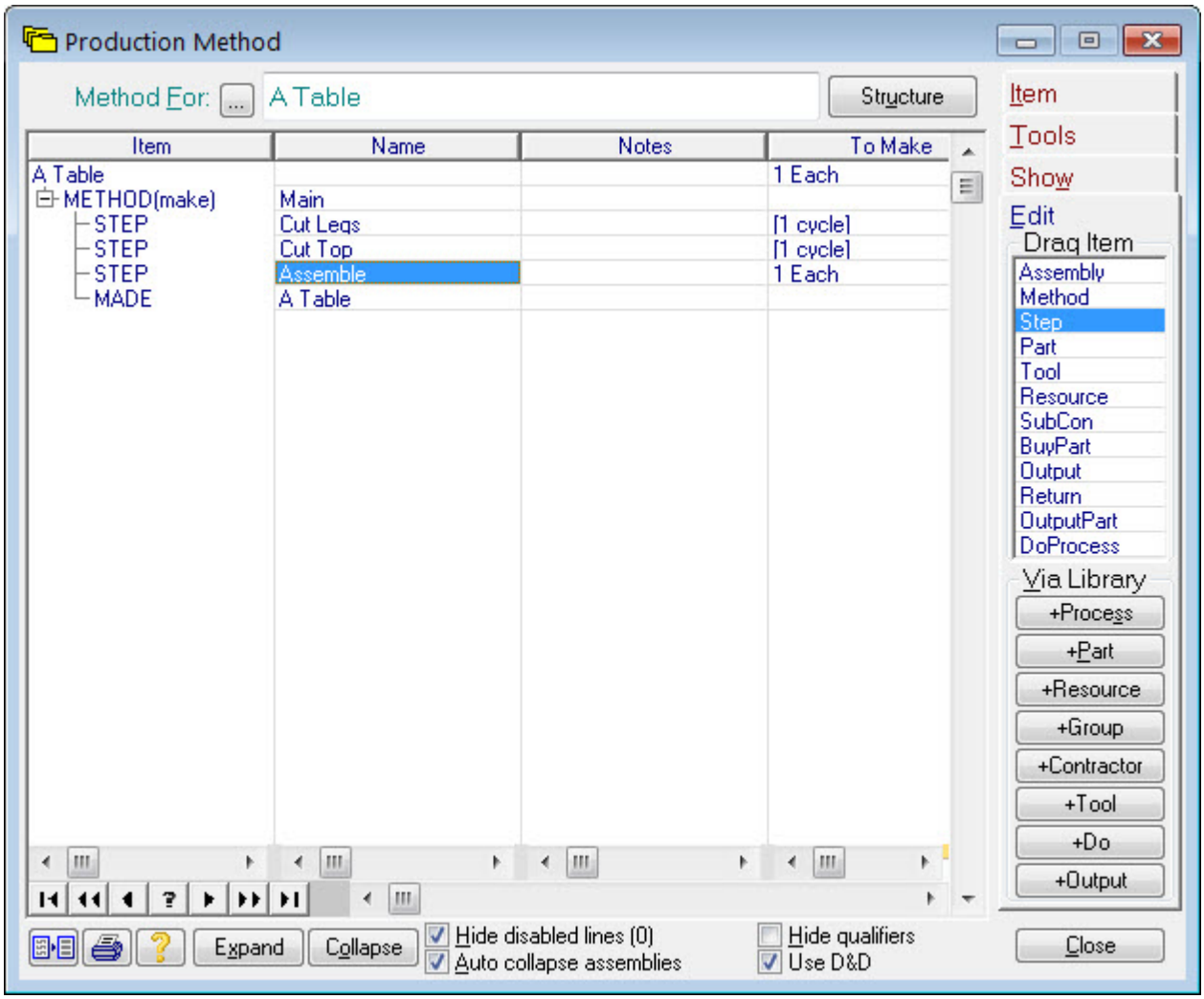


Double click on the first `Your Step Name Here` entry and type `Cut Legs` into the form that appears (see below), then press »Save«.

This is giving the first operation a name that means something to you. This name will be used on the works order paperwork.



Double click on the second and third `Your Step Name Here` entries and type `Cut Top` for the 2nd and `Assemble` for the 3rd. Your method should now look similar to this:

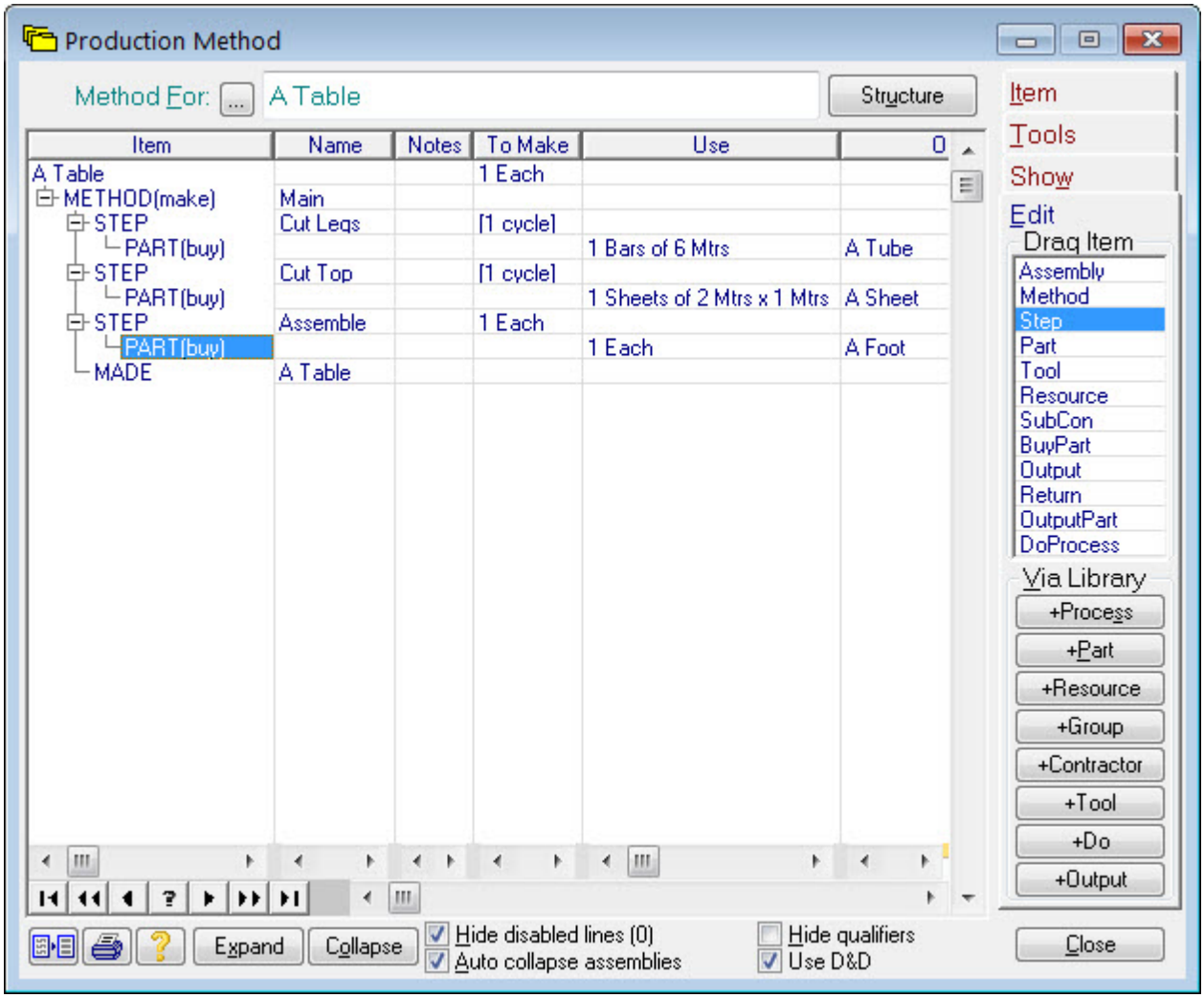


The next thing to do is to define the materials, or *parts*, required for each step. We will specify that we need some tube for the first step, a piece of a sheet for the second and some feet for the third. Refer to tutorial on how to create parts and create the following materials and parts:

1. A Tube as a 1D:Tube that is a bought item in 6 metre lengths.
2. A Sheet as a 2D:Sheet that is a bought item in 2 metre x 1 metre sheets.

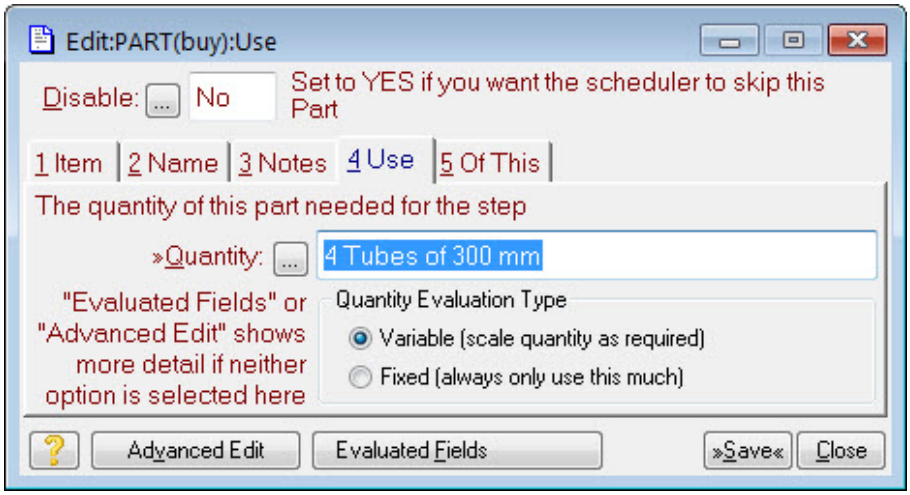
3. A Foot as a 1D:Item that is a bought item in units of 1 each.

Once you've done that, come back to your method editor and press the +Part tool. This will open your materials catalogue in a mode that allows you to drag items from it and drop them in your method. Find the *A Tube* material in your catalogue, then drag it into the *Cut Legs* step of your method (you may need to move the windows around a bit to do this). In a similar manner, drag the *A Sheet* material into the *Cut Top* step and the *A Foot* item into the *Assemble* step. Your method should now look similar to this:



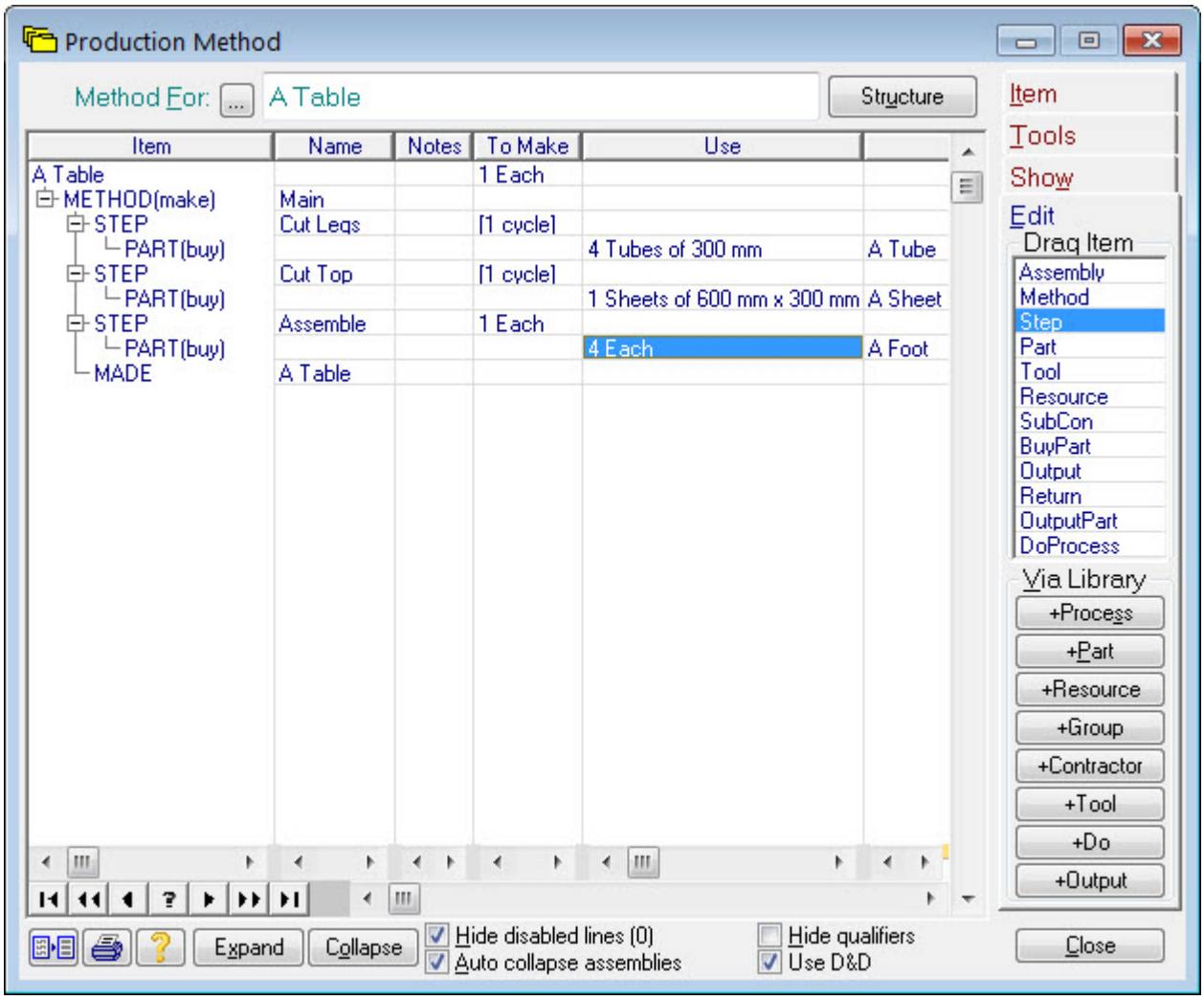
Notice the Use column shows a quantity. When you drop a material into a method, the quantity is initially set at 'one' of whatever you dropped. We'll edit that to be what we really want. Lets say our table has 300mm legs, a 600mm x 300mm top and requires a foot on the end of each of the 4 legs.

Double click on the `1 Tubes of 6 Mtrs` quantity associated with the `A Tube` material. An editor will open that allows you to change the quantity, type in `4 Tubes of 300mm` and »Save«.



This specifies that you need 4 lengths of 300mm of *A Tube* to make the legs. Notice you specify both the quantity and the size. The stock system is fully aware of sizes and will always allocate material that is big enough to do the job. The *A Tube* material record was setup as a 1D material; this tells the system that it is a thing that can be chopped up to any length required. The stock system will create a stock record for each different length you hold or buy, so you are not restricted to holding stock in any particular size, nor do you need to refer to it as a fraction of a full length.

In a similar manner, set the required quantity for the table top as `600mm x 300mm` and the required quantity for the feet as `4`. Your method should now look similar to this:

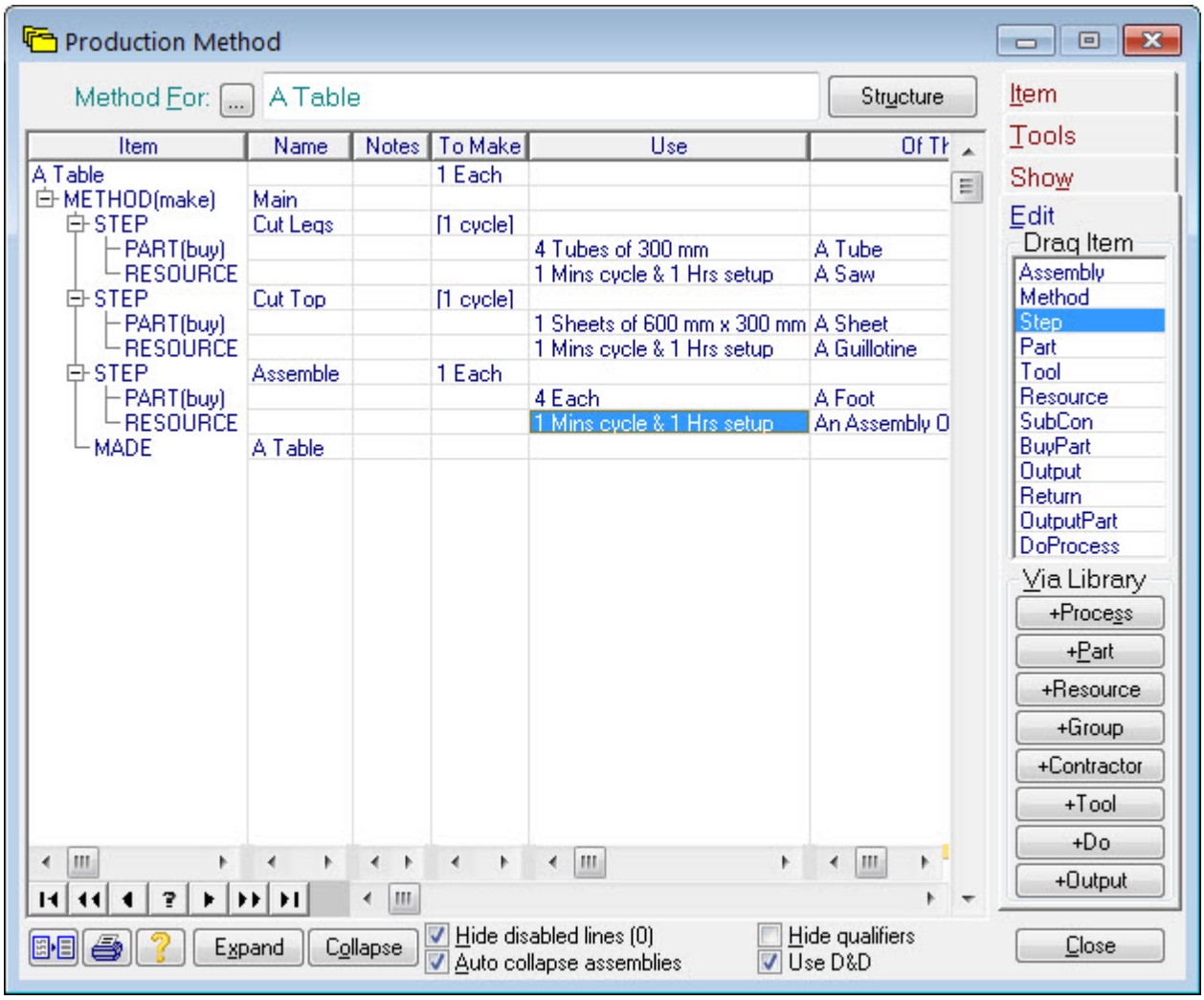


Now we've specified the materials required, the next thing to do is define the resources required. The resources specify what machines and people are required for each STEP and how much time is required. Two types of time can be specified - *SETUP* time and *CYCLE* time. SETUP time specifies the amount of preparation time required before any real work can begin. CYCLE time specifies how much time is required per *unit* processed. The unit is usually 'one' of something and is shown in the `To Make` column. The SETUP time is fixed and remains the same no matter how many units you are making, whereas the CYCLE time is scaled as necessary to reflect the number of units.

Referring to the tutorial again, create resources as follows:

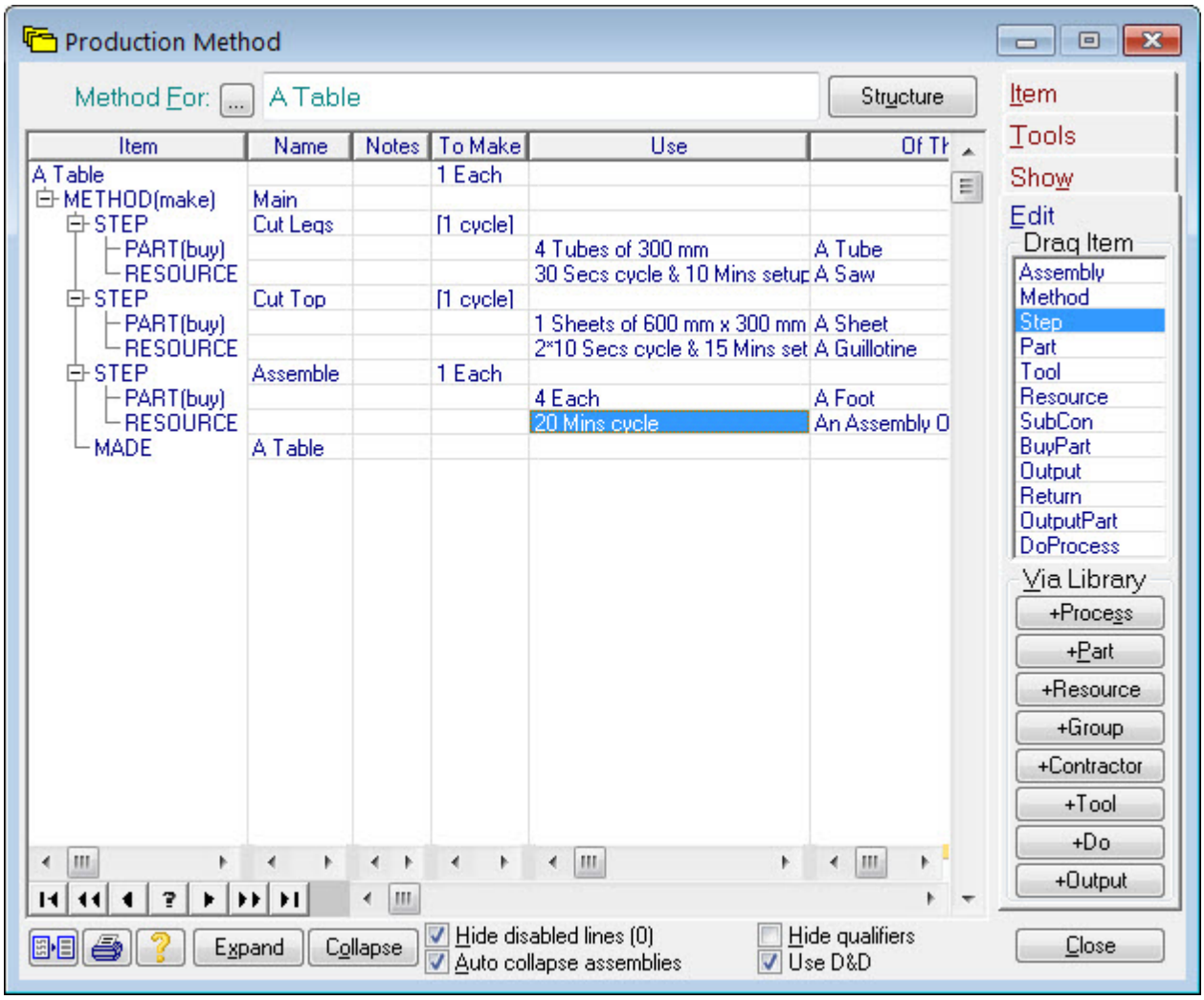
1. A Saw (a machine).
2. A Guillotine (a machine).
3. An Assembly Operator (a person).

Once you've done that, come back to your method editor and press the +Resource tool. This will open your resources catalogue in a mode that allows you to drag items from it and drop them in your method. Find the *A Saw* resource in your catalogue and drag it into the *Cut Legs* step of your method. In a similar manner, drag the *A Guillotine* resource into the *Cut Top* step and the *An Assembly Operator* resource into the *Assemble* step. Your method should now look similar to this:



Like dropping materials, initially the times set are 'one'. Just edit them to the real amount. Let's say it takes 10 minutes to setup the saw and 30 seconds for each cut; 15 minutes to setup the guillotine and 10 seconds per cut, with 2 cuts per table top; and finally no setup time for the assembly and 20 minutes to do the assembly of each table. Double-click on the `Use` column for each resource and edit the time as appropriate.

Your method is now complete and should look similar to this:



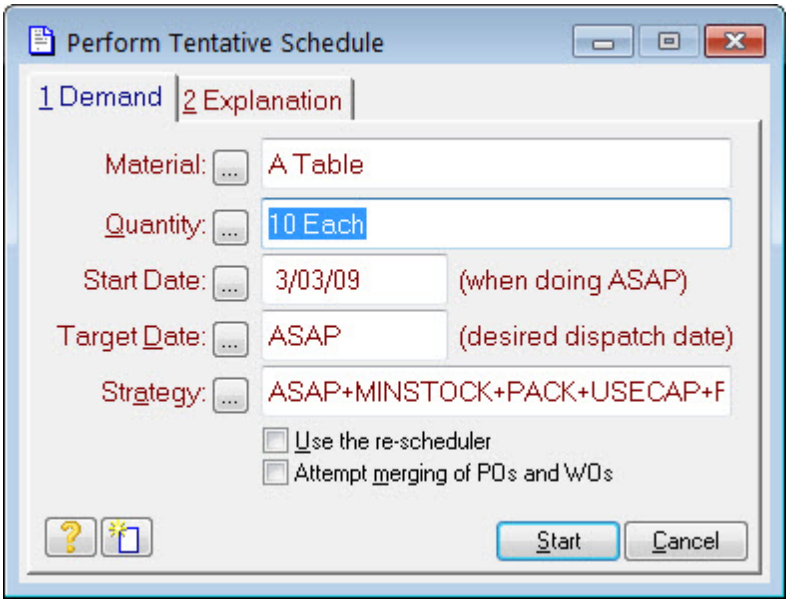
3 How do I test a method?

Once the method is complete it's a good idea to test it. This verifies that it makes sense to the planning system and you haven't specified something that is impossible (like trying to get a 300mm length from stock whose maximum length is 250mm). It'll also highlight omissions, like missing costs and suppliers.

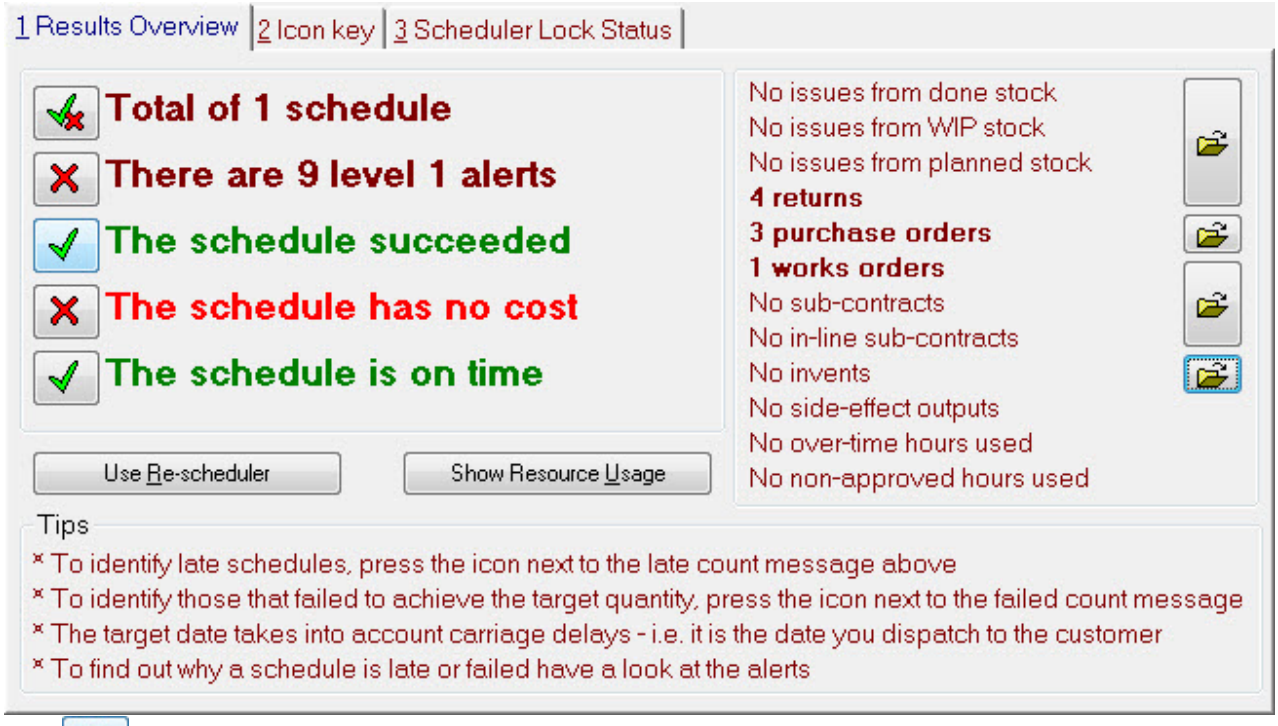
To test a method, select the Test tool on the Tools tab. This will bring up a form similar to the one on the right:

Enter a quantity to test against then press Start.

This runs the scheduler in a *What If?* mode as if you had no stock and nothing else going on.

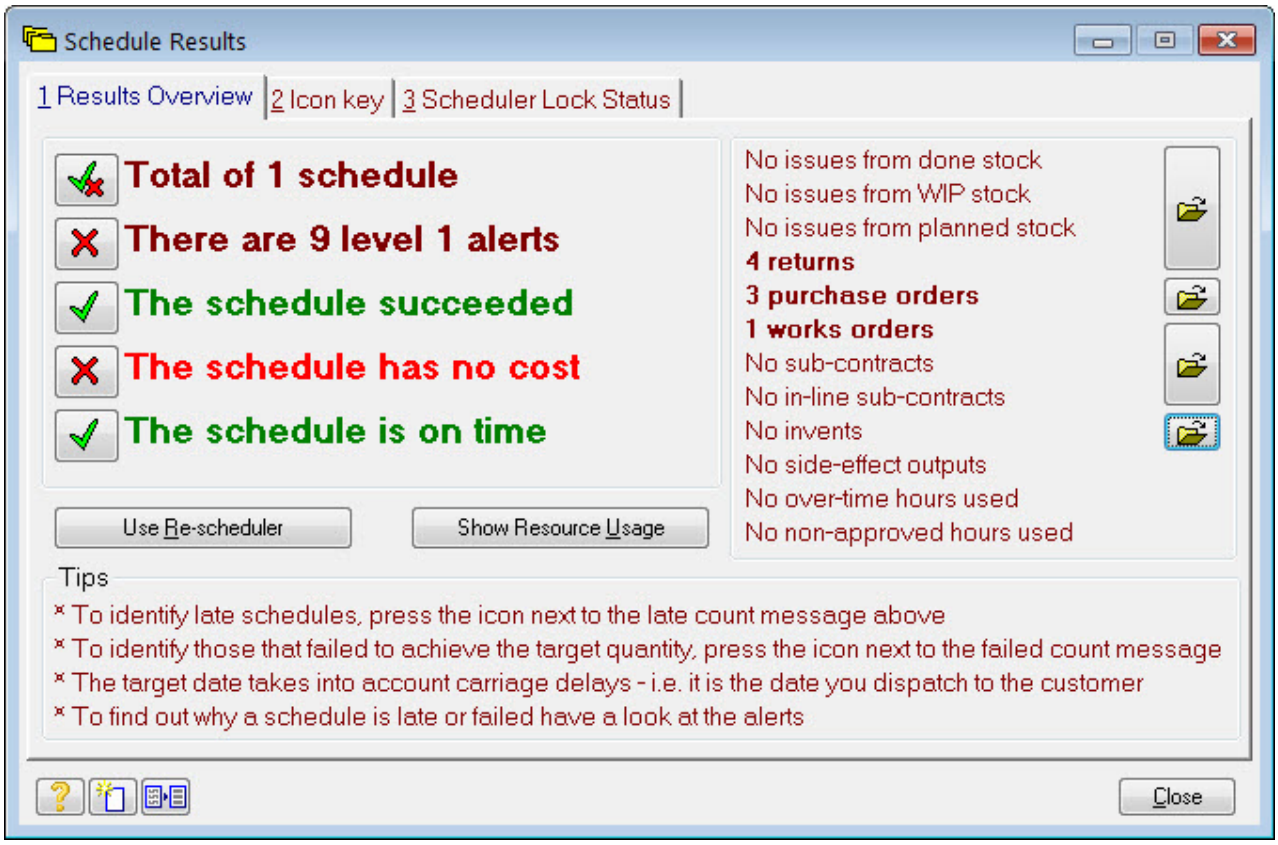


After a few moments a form will appear that shows a summary of the results. On the right side is a summary of the work involved in terms of the stock used (none in this test case), the number of purchases required, the number of works orders involved, etc. On the left side is a column of



and icons against various conditions. Ideally you want a tick on them all, but the critical one is The schedule succeeded. If you see this it means the method makes sense to the planning system and it will be able to plan work using it.

In the case of our *A Table* schedule the results will look similar to this:



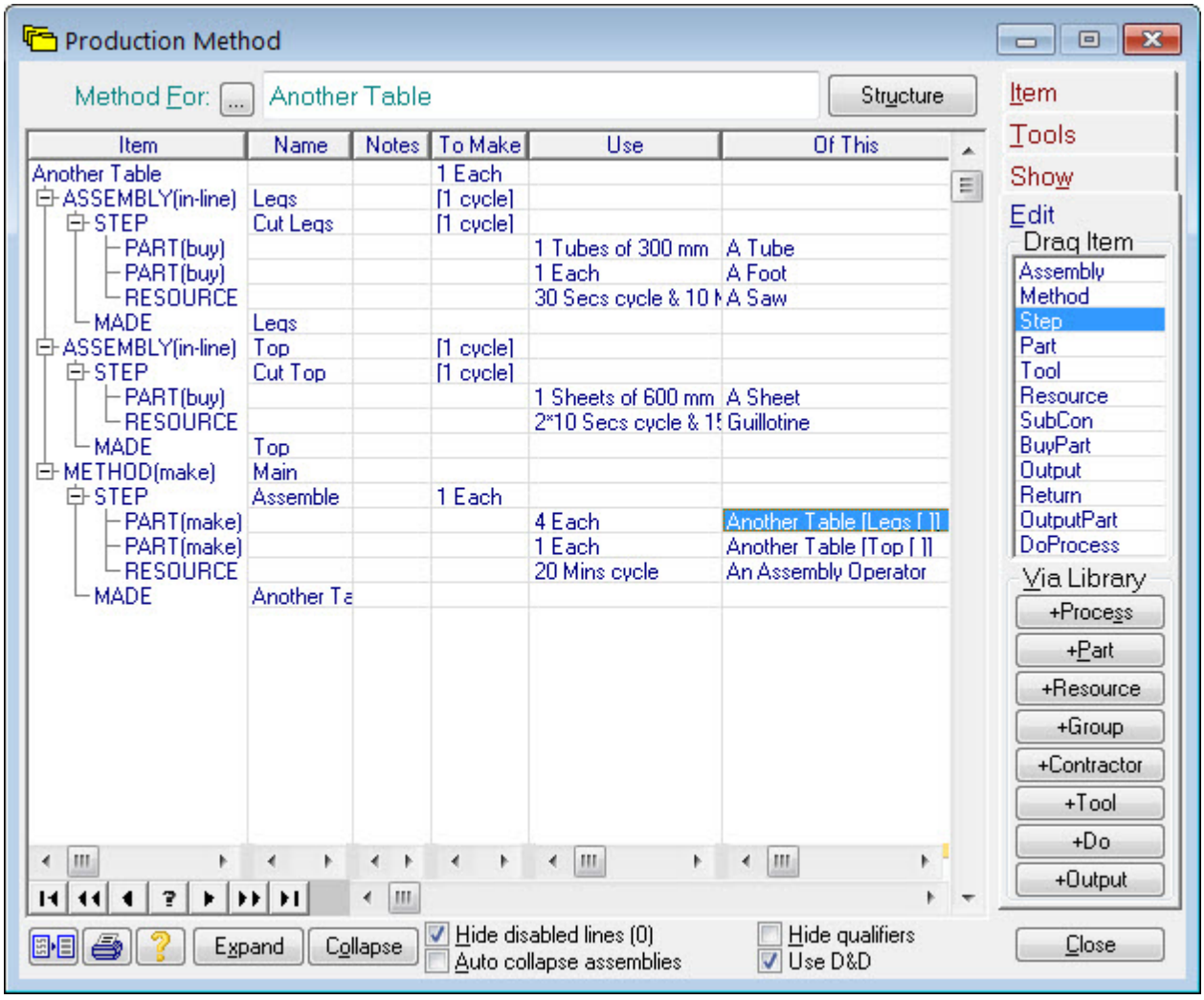
The  advice is telling us that either one of the resources used or one of the parts used has no associated cost. Adding suppliers and costs to the parts, or costs to the resources will resolve that. You can find out specifically what the issues are by pressing the  icon on the  advice. This will show you a list of all the alerts raised. An alert is a condition the scheduler noticed that is likely to be of interest. There are a very large number of possible alerts; they advise you of conditions that need resolving or of conditions that prevent you meeting a required delivery date.

4 Method Elements

This section contains reference information for every possible element that can be added to a method. Unless you've changed the short-cuts each is available in the `Drag Item` list.

4.1 Assembly

An *Assembly* record marks the beginning of a sub-method for an assembly that is going to be incorporated into the final product. An assembly can contain steps with parts and resources just like the main method. They are referred to in the main method by their name in a PART record. They behave in exactly the same way as a method for a manufactured part; the only difference is that its definition is embedded in another method. Here is another way to define our *A Table* example above using assemblies:



Here we've defined an assembly to make one leg and another to make the top, then in the main method they are just called up as PARTs. Notice the names used in the PART record are of the form `Product Name [Assembly Name]`, the part within the [and] is the name you gave the assembly.

Defining the method in this way allows the planning system to manufacture the legs and the tops at the same time if resource capacity allows.

Hint: You can make a PART record for an assembly by *control-dragging* the ASSEMBLY line into the method STEP where you want to use it. Hold the control key down, then drag the ASSEMBLY line and drop it in the STEP. A PART record will be created that refers to the assembly.

There are several *types* of assembly:

Works Order

A works order assembly specifies that you want the assembly made under its own separate works order and not be part of the main product works order.

In-Line

An in-line assembly specifies that you want the work for the assembly to be done as part of the works order for the main product. The assembly steps appear on the paperwork along with the rest.

Bought

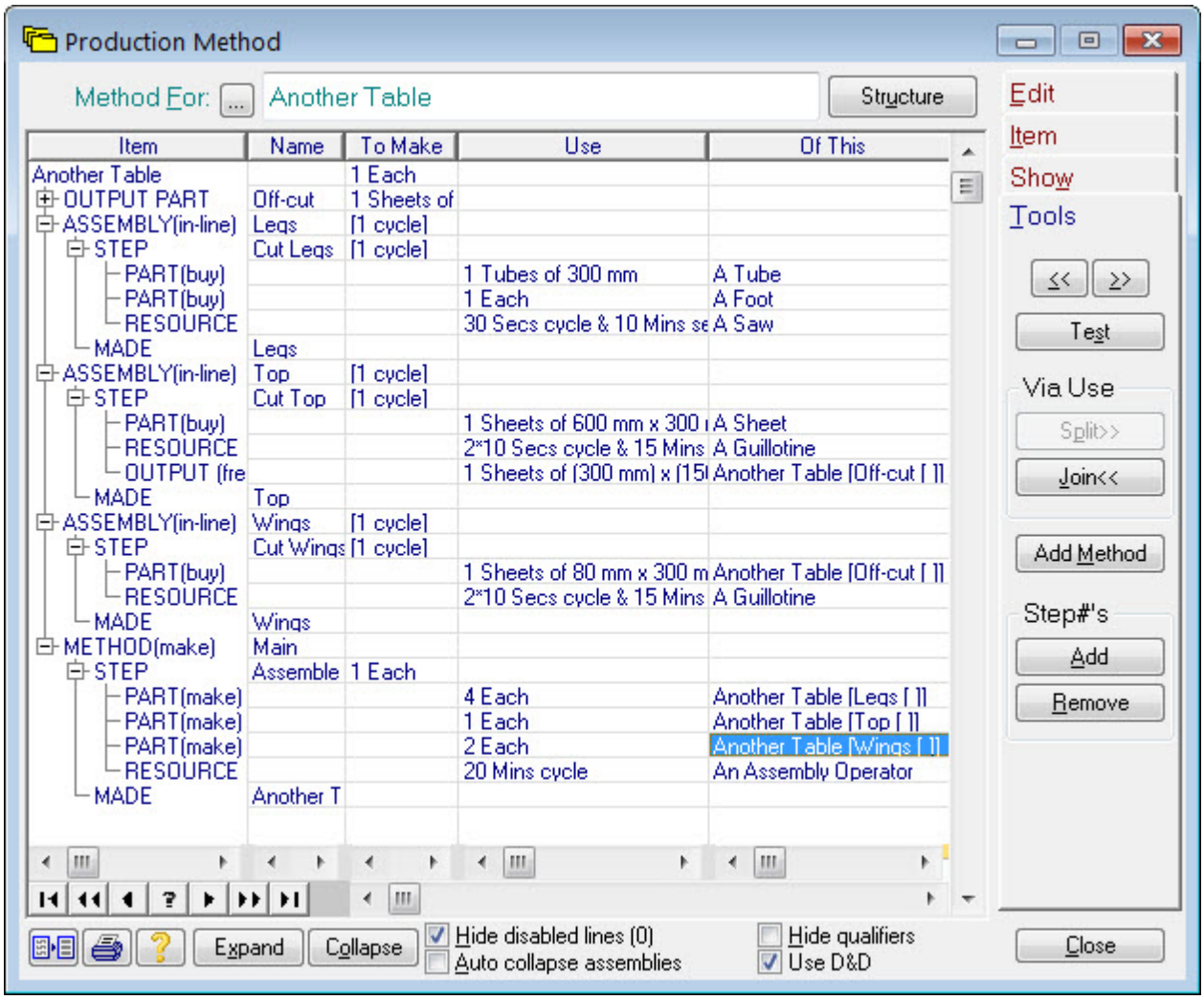
A bought assembly specifies that this is something you buy rather than make. These behave exactly the same as raw materials in your main materials catalogue. They are useful for special materials that have to be bought for this product and are not useful for anything else.

Free-Issue

A free-issue assembly specifies that the customer gives you the part for incorporation into the product; you do not have to buy it or make it.

Output

An output assembly is a place-holder for something the main method (or one of its assemblies) will create as a side-effect. For example, imagine our table has a rectangular hole cut out of the middle of the top and the piece cut out is to be used to make a 'wing' on each table edge. The method for this could look like this:

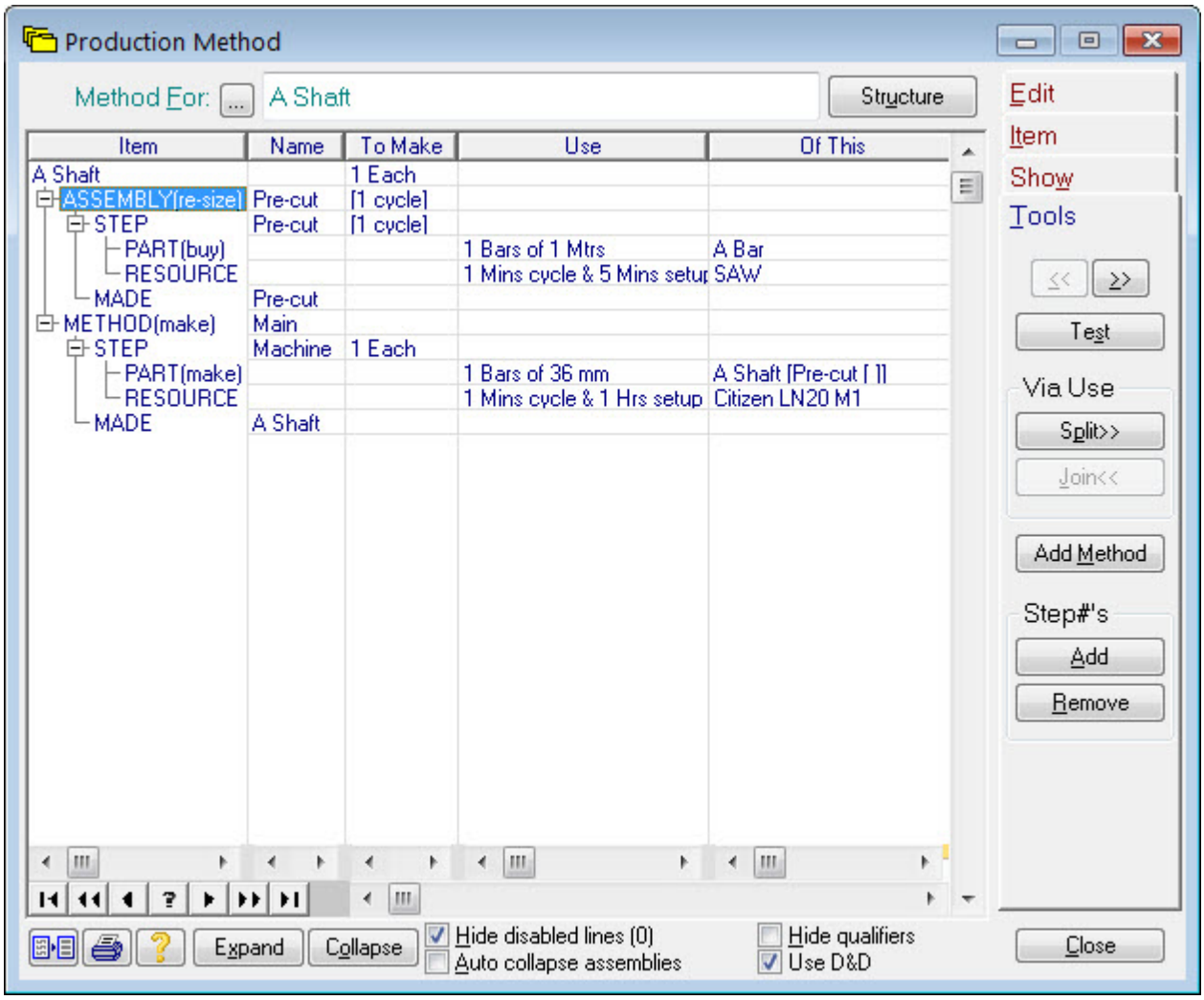


Notice the *Top* assembly is producing the OUTPUT and the *Wings* assembly is using it.

Re-size

A *re-size* assembly specifies that some raw material is going to be made into a different size. These are typically used to prepare bars for bar-fed lathes that cannot take the full length as supplied. A re-size assembly can only consist of a single STEP with a single PART.

Here is an example that reduces a 3 metre bar to 1 metre then machines those 1 metre lengths:



This will create two works orders, one to pre-cut the bars to 1 metre lengths, then a second to machine the 36mm shafts from those lengths. The size the bars are being pre-cut to is specified by the PART use column in the assembly, in this case “1 Bars of 1 Mtrs” The use column in the main method specifies the length required for each shaft, “1 Bars of 36 mm”. **Note:** the planning system will only cut-down full size bars if it cannot find any stock of bars at or below the pre-cut length (1 metre in this example). Also, any off-cuts are fully tracked through both cutting operations, as are yield calculations. The off-cuts from either operation (Pre-cut or Machine) are the same material and considered to be the raw material ("A Bar" in this case). So other methods requiring "A Bar" will pick-up off-cuts from wherever they occur.

The above is a ‘1D’ example, but re-size assemblies can be used equally well on ‘2D’ materials (i.e. sheets).

In-Line re-size

An *in-line re-size* assembly is identical to a *re-size* assembly except the re-size operation is done within the same works order as the main method (as a separate STEP).

4.2 Method

A *Method* record marks the beginning and end of a production method. There may be more than one method record, in which case you are defining alternative ways to make something. In this case the planning system uses the first (enabled) method it comes across (starting from the top) and the rest are ignored.

There are several *types* of method:

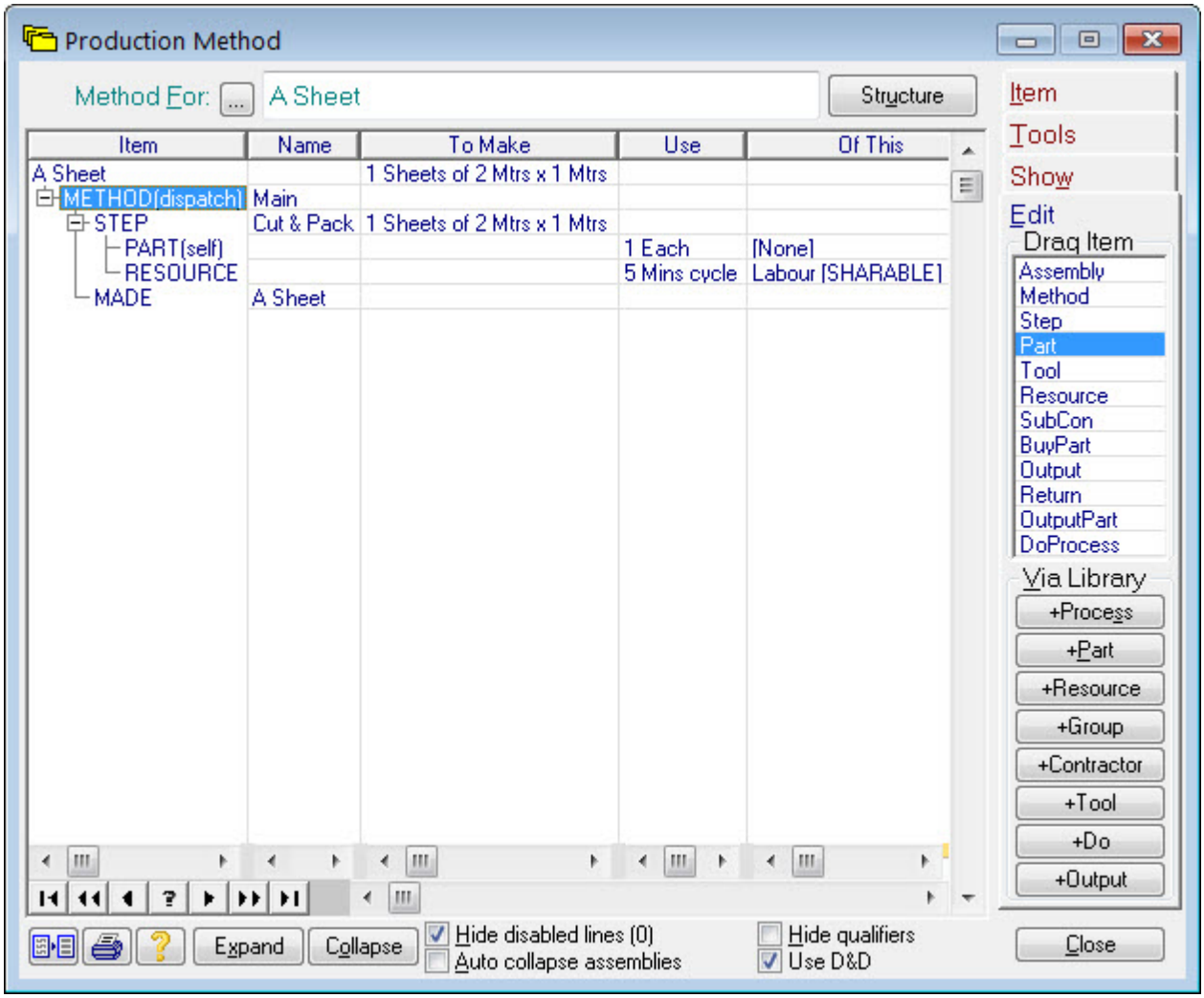
Make

A *make* method defines a production method to turn raw materials into products. This is the usual case, see example above.

Dispatch

A *dispatch* method defines a method that is used to prepare stock for dispatch, such as re-packaging. These are optional and if present cause a works order to be produced for every dispatch. They can only consist of a single STEP and that step must have a *SELF* part in it. They are useful to capture time and materials used for dispatching where the product is usually made for stock and kept on your shelves until needed. If you make and dispatch to order, it's usually more convenient to add a STEP in the make method.

Here is an example of a dispatch method that cuts and packs a part of *A Sheet* we created earlier:



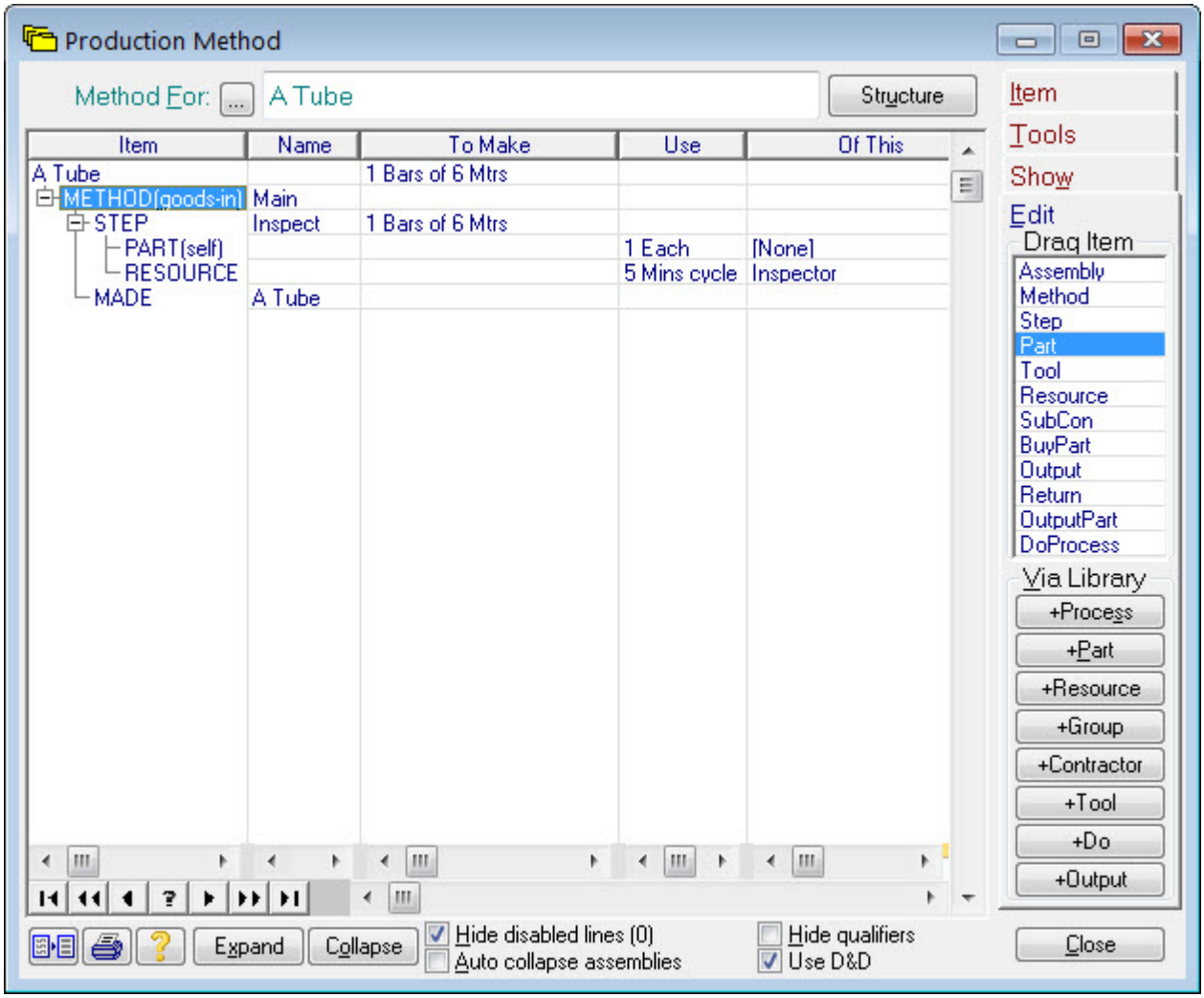
Notice the PART(self). This means, "I want whatever size the customer asked for in the order". A self part is a part record with no material (i.e. `Of This` is blank). If you do a test for various sizes you'll see that the planning system calculates the yield from your stock sheets and allocates an appropriate number to be cut as required.

Also notice, this dispatch method is attached to a bought material. That's fine, it just means you are prepared to sell your raw materials.

Goods-In

A *goods-in* method defines a method that is used prepare stock for the shelf that has been received from suppliers. This is useful to model significant processes, such as un-packing and inspecting. These are optional and if present cause a works order to be produced for every delivery expected from your suppliers. They can only consist of a single STEP and that step must have a *SELF* part in it.

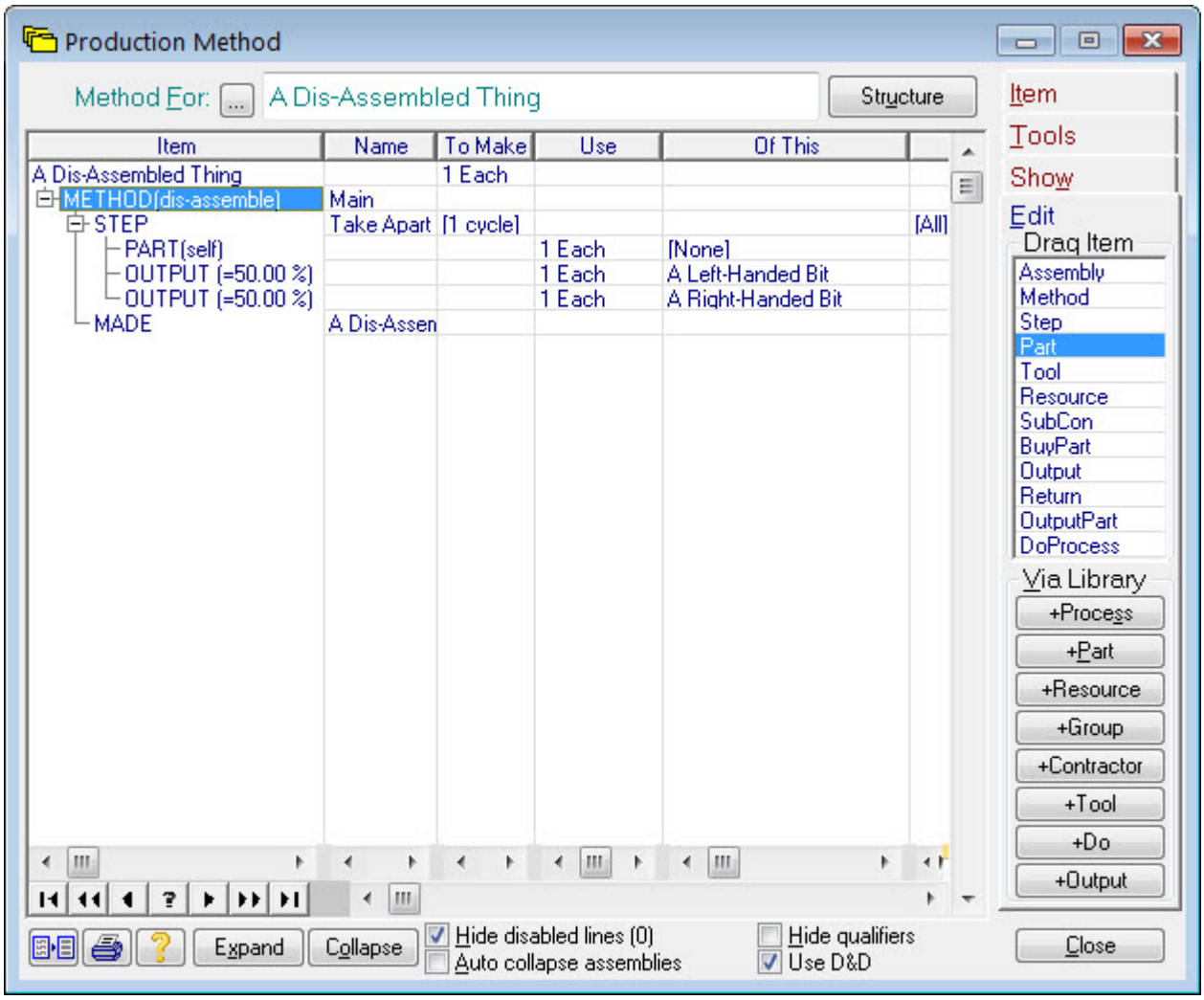
Here is an example of a goods-in method that inspects *A Tube*:



In this case the *SELF* part means, "use whatever was delivered".

Dis-assembly

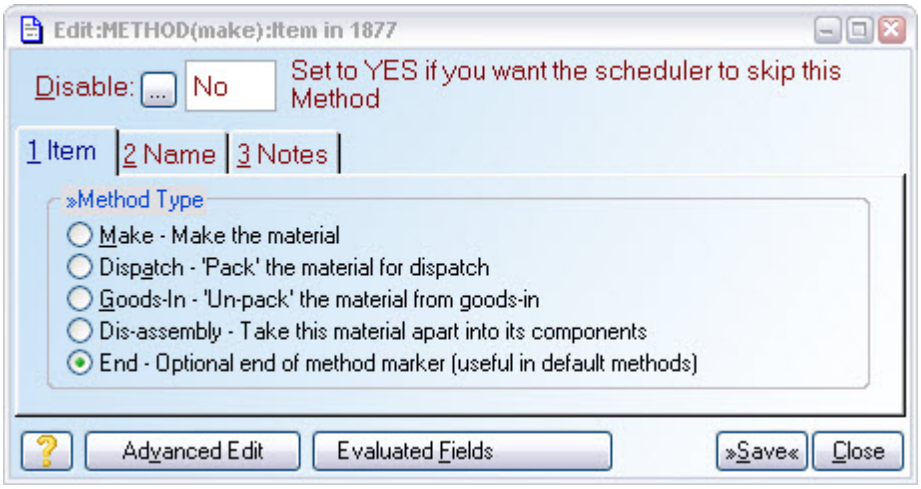
A *dis-assembly* method defines a method that is used to take a bought assembly to pieces. This is useful if you want to use components of the bought thing in your products and this is the only way you can get the components (think of the bought thing as a 'kit' of useful parts). Here is an example that extracts two parts from a bought assembly:



Here the *SELF* part means “take whatever we bought”. The two *output* records define what is being extracted and what they’re worth (as a percentage of the cost of buying the whole thing). Any method that requires *A Left-Handed Bit* will cause the assembly to be bought (via a Purchase Order), then taken apart (via a Works Order), then the part extracted is allocated.

End

An *end* record marks the end of a production method. They can be left out if the context is un-ambiguous, buts its safer to always use them, especially when you are using *LIKE* processes. To add an *end* record, drag a *Method* record onto the end of the method, double-click on it to bring up its properties, and select the End radio option. It should look like this:



It is particularly important to use a an explicit *end* method record when defining default methods. To not do so will invalidate the default method and it will not be used.

4.3 Step record

A *STEP* record marks the beginning of an operation. There are two basic types of STEP – in-house and sub-contracted. An *in-house* STEP is one you perform using your facilities; a *sub-contracted* STEP is one that is performed on your behalf by a contractor using their facilities. STEPs can have any number of PARTs, TOOLs, RESOURCEs (when in-house) or SUPPLIERs (when sub-contracted).

STEPs are performed in sequence, starting at the top of the method (or assembly) and working down. The result of each step is implicitly handed to the next step for further work. If you say nothing, the entire result of a step is handed to the next, but this can be controlled if necessary (via the "*Next Step Needs*" quantity).

A STEP can also be in-line or separate. An *in-line* STEP is performed within the same works order as the STEP after it. A *separate* STEP causes the planning system to create a separate works order for subsequent steps. When a sub-contracted step is separate, the planning system will create a works order to create the free-issue 'kit' to pass to the contractor, a purchase order to specify what you want the contractor to do, and another works order to cover any work required once the contractor has done their bit.

Another specification you can give to STEPs is whether progress logging is optional or compulsory. When *compulsory*, attempting to log progress on subsequent steps will not be allowed unless progress has also been logged on this one.

4.4 Part record

A *part record* specifies some material that is required to perform the operation. That material, may be bought (a 'raw' material) or made (a sub-assembly) or both. The material required and how much of it can be specified. Usually, the quantity specifies how much material is required to make 'one' of something. The planning system then scales that up as necessary to meet any particular demand. The stock allocation algorithm is aware of dimensions and min/max size constraints, and various other factors and takes these into account when calculating yields.

The PART record specifies 'what' you want and the planning system calculates 'how'.

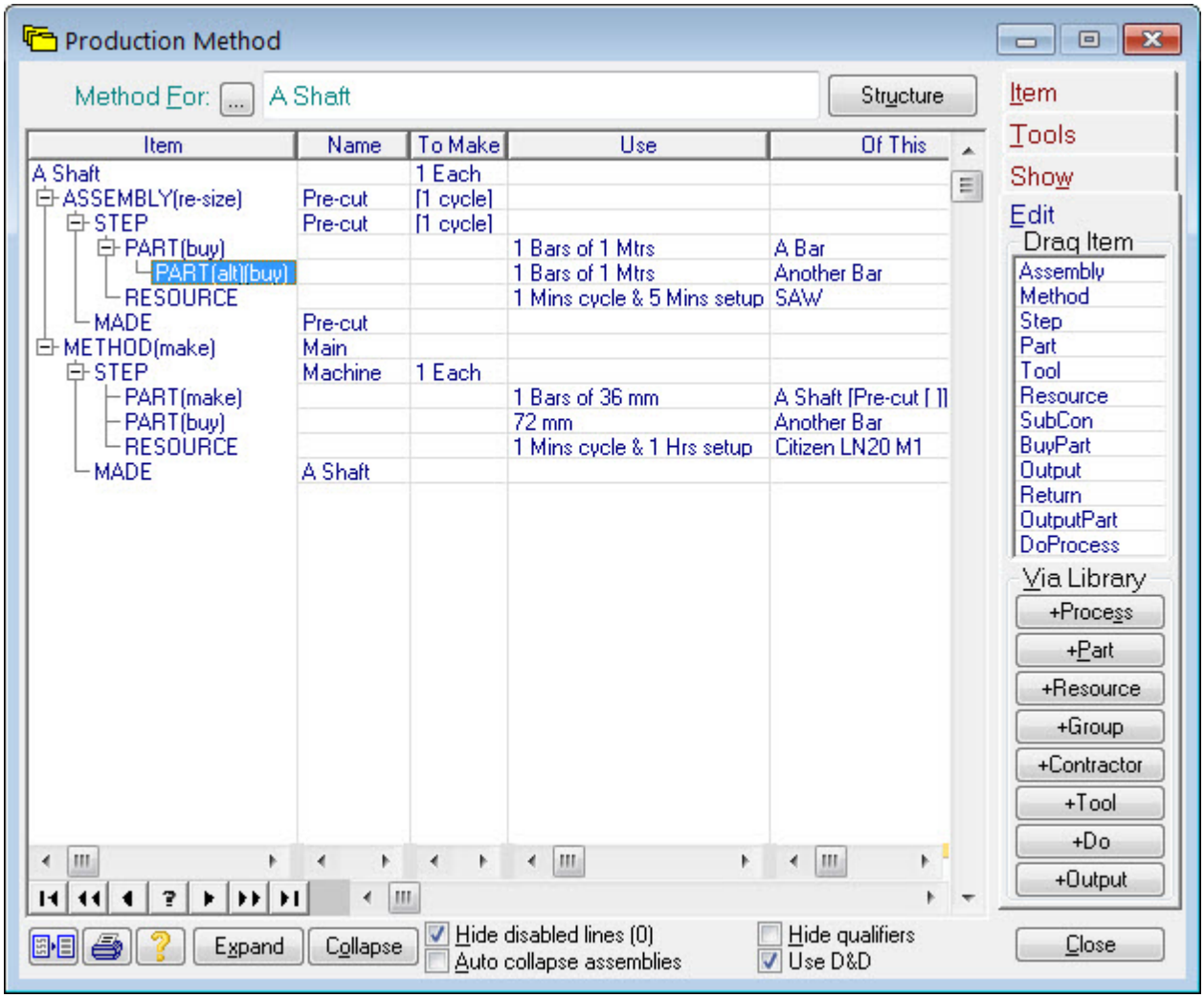
4.4.1 Part Types

There are several part *types*.

First Alternative

This marks the beginning of a PART specification and also defines the first alternative (choice) for the material. Choices for the part are identified by indentation; multiple parts are identified by being on the same level, like

this:



Here the re-size assembly can use *A Bar* **OR** *Another Bar* and the main method needs *A Shaft [Pre-cut [ ]]* **AND** *Another Part*.

Next Alternative

This marks another choice for a PART (see above). There can be any number of alternatives.

Reference Only

This is just a place marker for documentation purposes. It'll appear in the works order, but no stock will be allocated for it.

Use same parts as another step

This is useful if you have a step with a large number of parts and the same parts are required for more than one step. In this situation, just refer this PART to the STEP and the same parts as that step will be used here. For example, if you make a left-handed widget and a right-handed widget and they are identical from their component parts point of view. This means you only have one parts list to maintain when things change. Other parts can be included as well, so you could use this facility to specify a common 'kit' then specialise it as necessary.

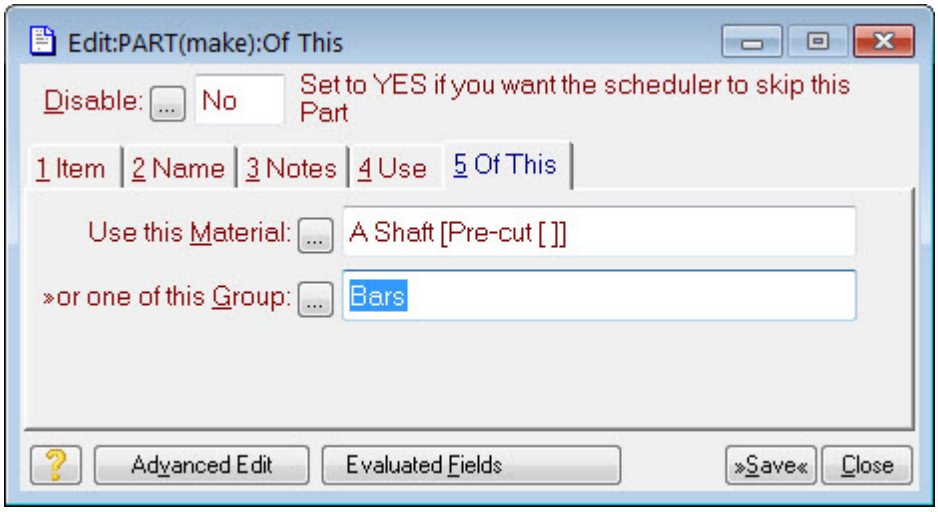
Use the outputs of another step

This is useful in conjunction with dis-assembly methods. Say you buy something, then take it apart and re-build it into something else. Just specify all the parts extracted in the dis-assembly method, then refer to that

step with one of these part types. The benefit is that, again, you only have a single parts list to maintain.

4.4.2 Materials (Of This column)

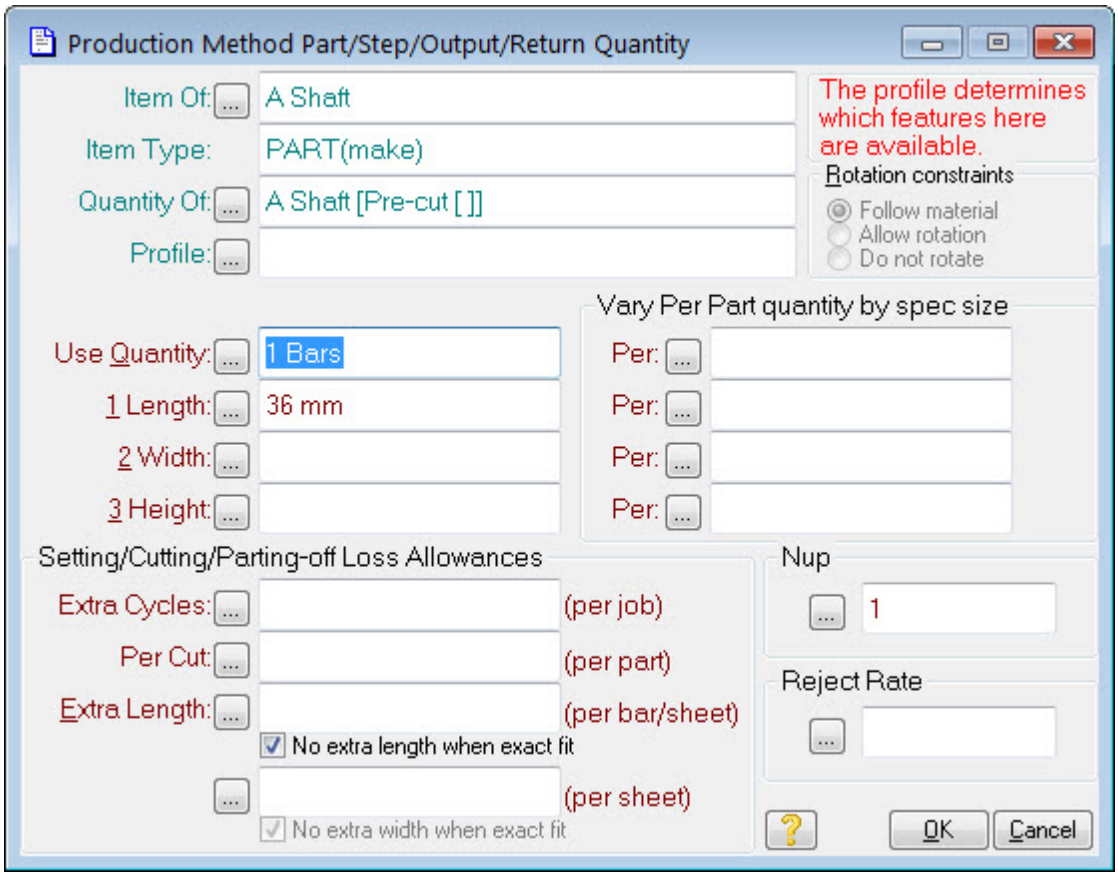
The material of a PART can be specified explicitly or it can be specified as any member of some group of similar parts, or both. If a group is specified you are defining a choice of any member of that group. If both a group and a specific material is given, you are again specifying a choice of any member of the group **OR** the specific material.



If neither the material nor the group is specified (i.e. both fields are blank), the part is referred to as a 'self' part. A self part is implicitly referring to itself (i.e. the product or assembly it is within). This is only valid for goods-in, dispatch and dis-assembly methods.

4.4.3 Quantities (Use column)

The quantity of a PART specifies how much of the material is required to make the unit quantity of the product. The unit quantity is shown in the `To Make` column. When dimensioned material is involved (e.g. bars, rods, extrusions, sheets, plates, etc.) there are a number of 'loss' allowances that can be specified. The form below shows the various aspects of a quantity. This form is reached by selecting **`advanced edit`** then **`more...`**



Right-click in a field to get an explanation of that field.

The planning system is aware of dimensions and takes yields into consideration when allocating part of a 'thing'. Depending on how you specify the quantity you want, the yield calculations will be 'strict' or 'lose'. A *strict* calculation takes all dimensions into account and will ensure only pieces that are big enough are allocated. A *lose* calculation doesn't care about physical sizes so long as the quantities 'add-up', e.g. 20mm + 30mm is good enough to satisfy a requirement for 50mm when being lose but not when being strict.

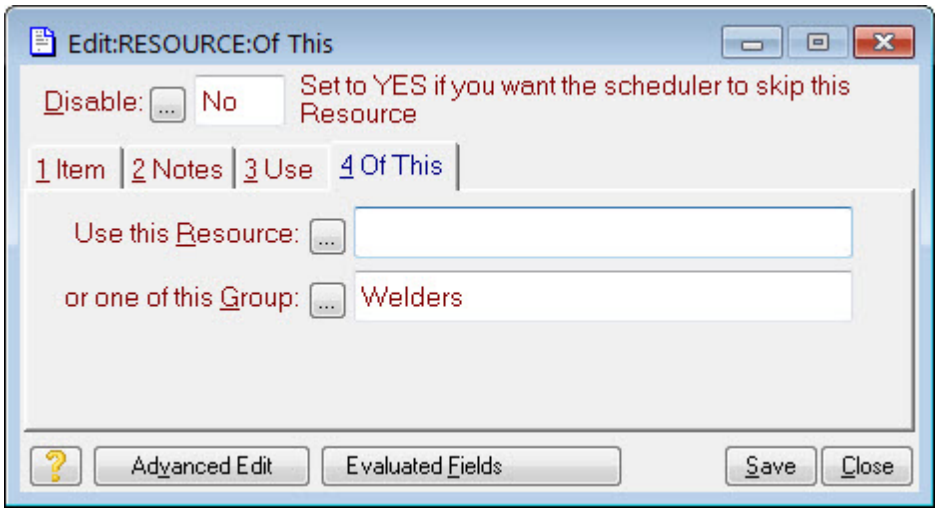
If the quantity you require is purely specified as a length, e.g. 36mm, then you are being 'lose' and saying "I don't care how the big the pieces are so long as they add up to 36mm".

If you want the quantity as a single piece of a required length then you must specify it as "1 bars of 36mm".

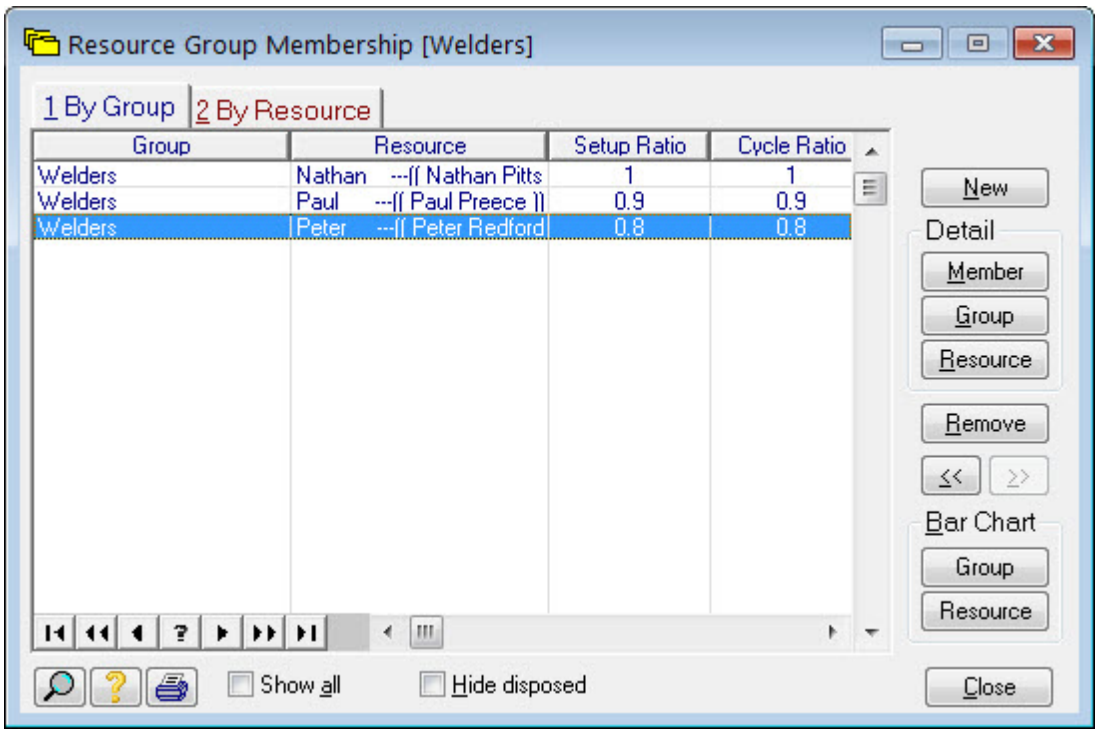
4.5 Resource record

A *resource record* specifies a person or a machine that is required to perform some operation and how much time is required. The resource can be specified explicitly or it can be chosen from a group of similar resources. Like PART records, you can specify choices. Choices are particularly useful for resources as it helps to prevent bottlenecks. If a particular resource is busy when there are choices, the planning system will just look for one that is not busy. If there are no choices available, the job will be delayed until a suitable resource becomes available.

Using resource groups is also useful to model skills. E.g. in a method requiring a welding skill, rather than specify particular people with that skill, instead define a group called 'welders' and put the people with that skill in the group. Then in the method just say you need a 'welder':

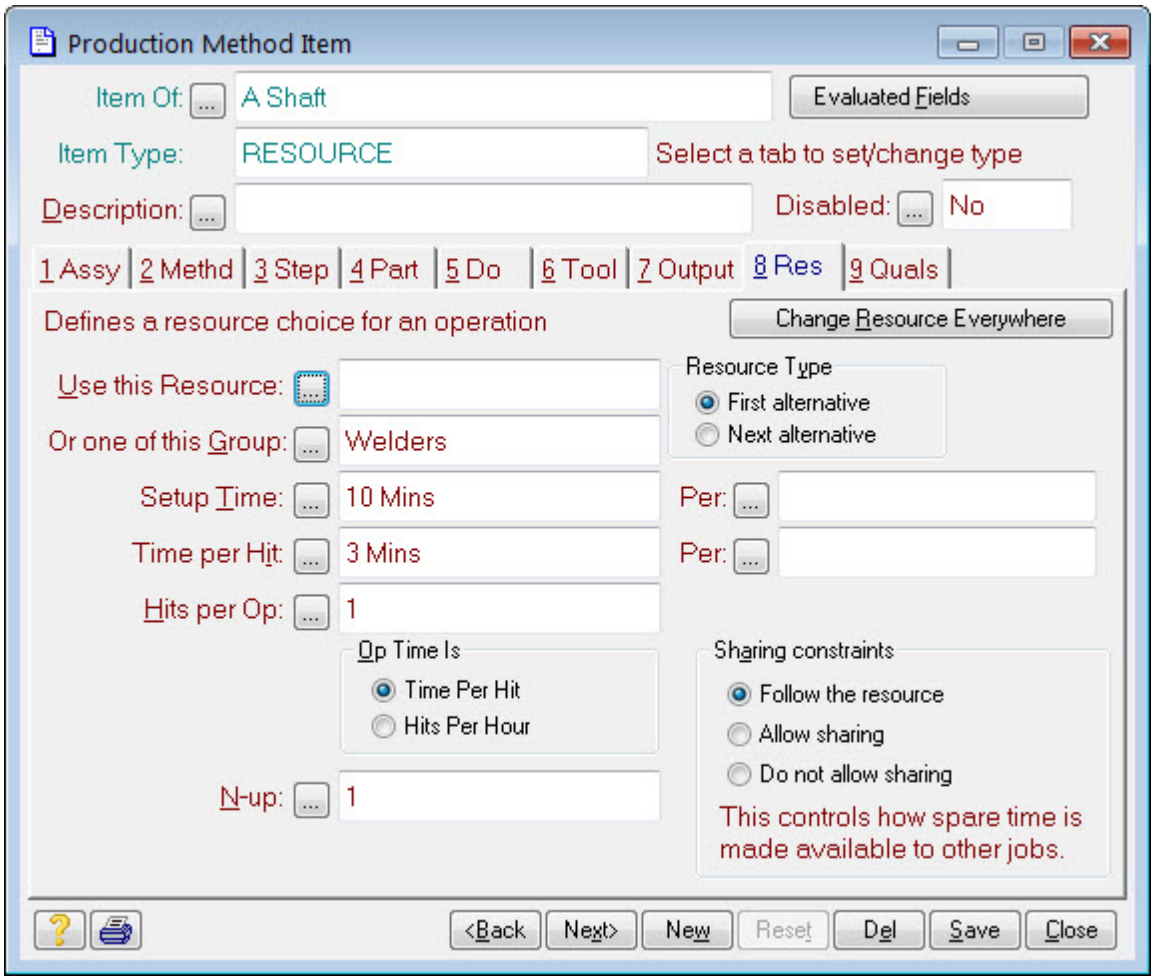


There are two main benefits to this: 1) you only have to change one thing as you hire and fire welders – the group membership; 2) you can specify relative capabilities so that more capable people are used in preference to less capable ones:



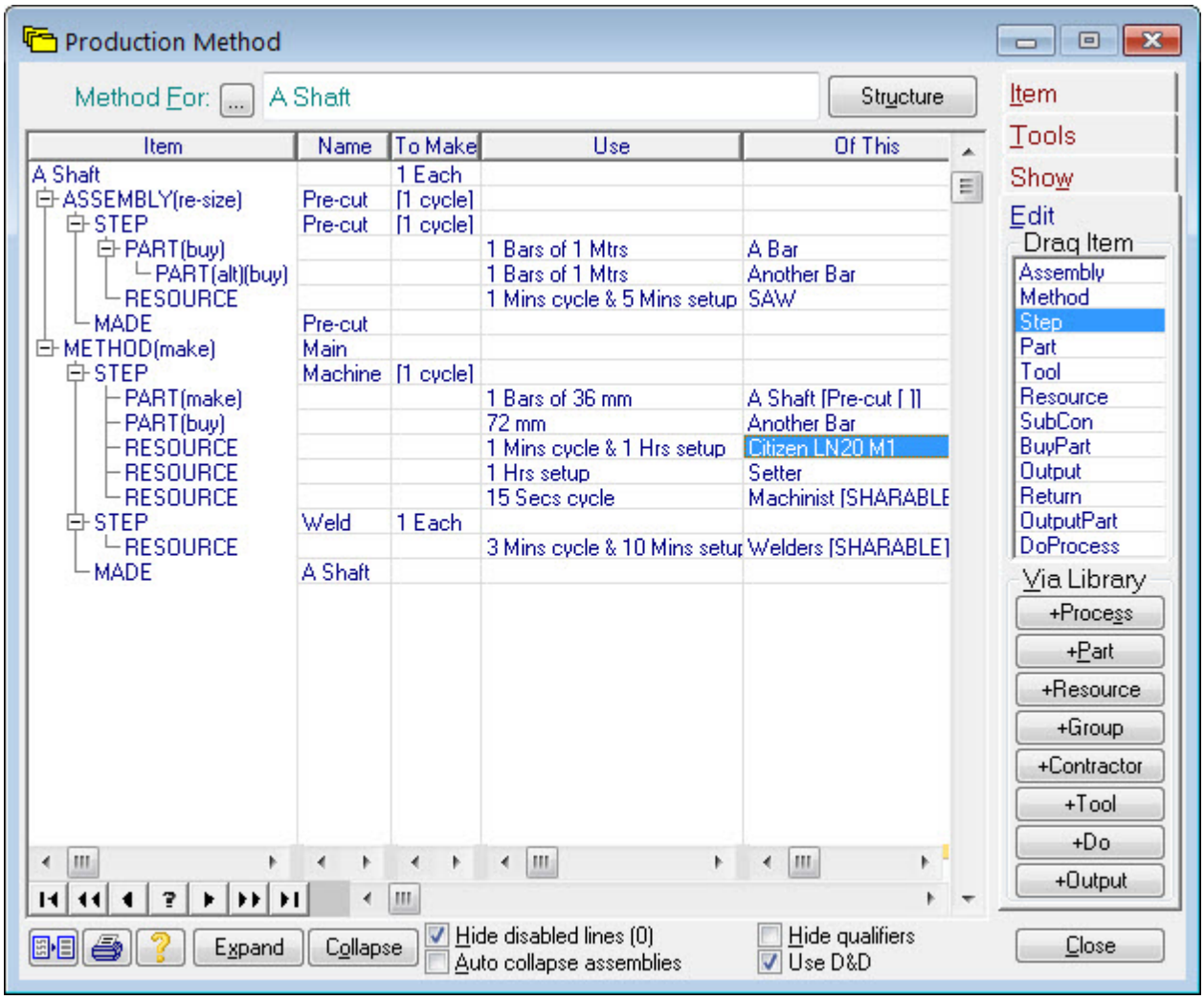
The setup and cycle 'ratio' in the above example shows the relative capabilities of the members of our welders group. A value of 1 is the benchmark, values less than 1 indicate less capability and values more than 1 indicate more capability.

There are two types of time that can be specified for a resource: *setup* time and *cycle* time. The setup time is fixed, it represents preparation time, the same amount is required no matter how many things you are going to make. The cycle time usually represents how much time is required to make 'one' of something. This time is scaled up by the planning system to reflect how many are actually going to be made. Like PART quantities there are a number of aspects of resource time:



The form above shows the various factors and is reached by selecting **`advanced edit`** on a resource record. Right-click on a field to learn more about it.

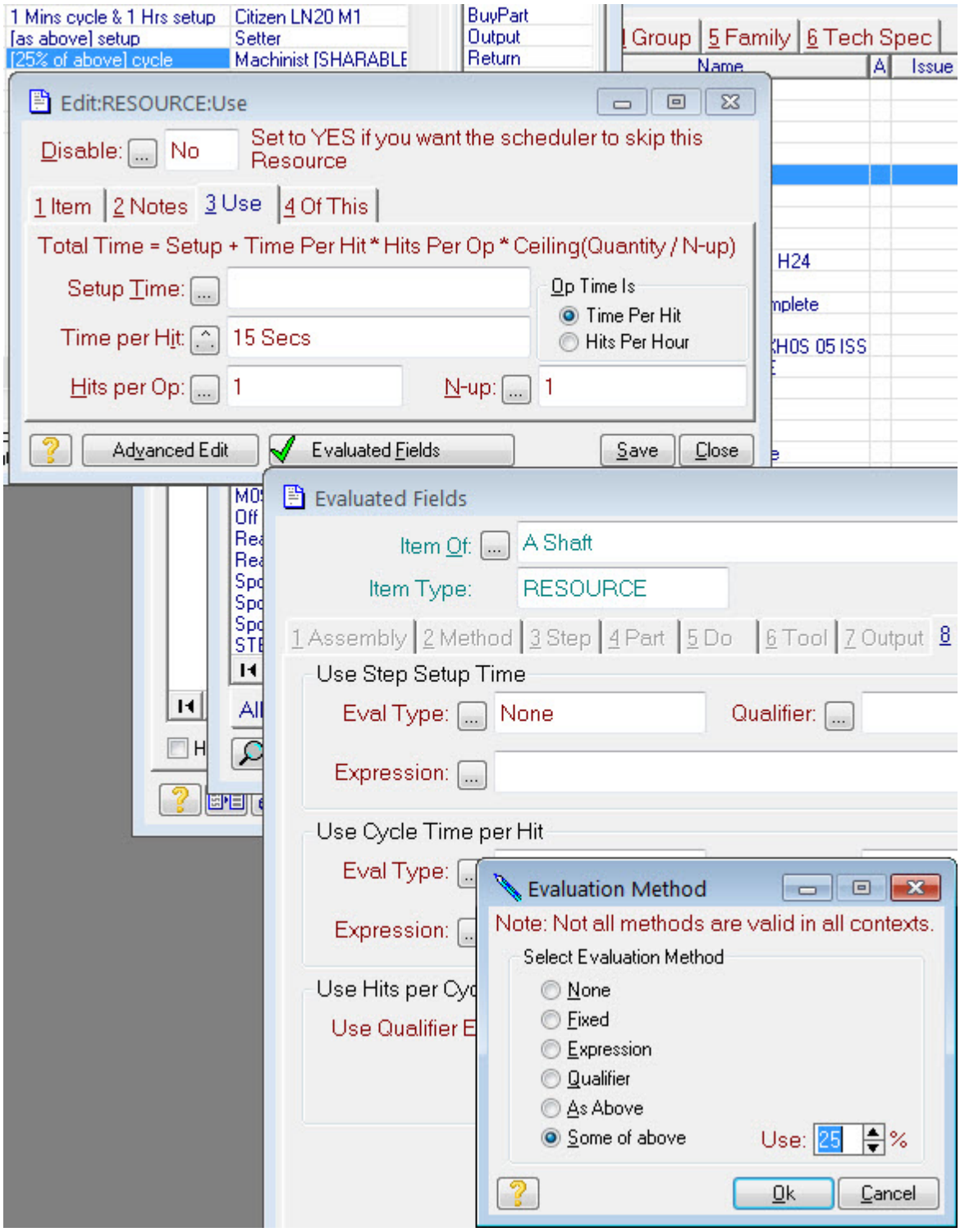
As well as resources choices, an operation may also require multiple resources to perform its operation:



The machine step in the above example has a requirement for three resources: the Citizen LN20 M1 **AND** a Setter **AND** a Machinist. Notice the Setter has no cycle time and the Machinist has no setup time. This means you need a Setter only for setup and he's released once the setup is complete and the Machinist is not required until the machine has been set. Also, notice the cycle time for the Machinist (15 seconds) is less than that of the Citizen LN20 (1 minute). This means the Machinist is not fully occupied running the Citizen and because the Machinist is 'sharable', he is allowed to be allocated to other jobs to use up any spare time.

Generally, machines are not sharable and people are. Sharable means the resource is allowed to be allocated to more than one job at a time.

When using multiple resources in a step, as in the *Machine* example above, you can alternatively specify the times relative to the first resource on the step. This means you only have to edit one record should you decide to update the times for the step. This is particularly useful when the step is defined as a 'boiler-plate' in the process library. To specify times as relative, use the **`Evaluated Fields`** tool and select the `As Above` or `Some of Above` options as the `Eval Type`, as illustrated below:



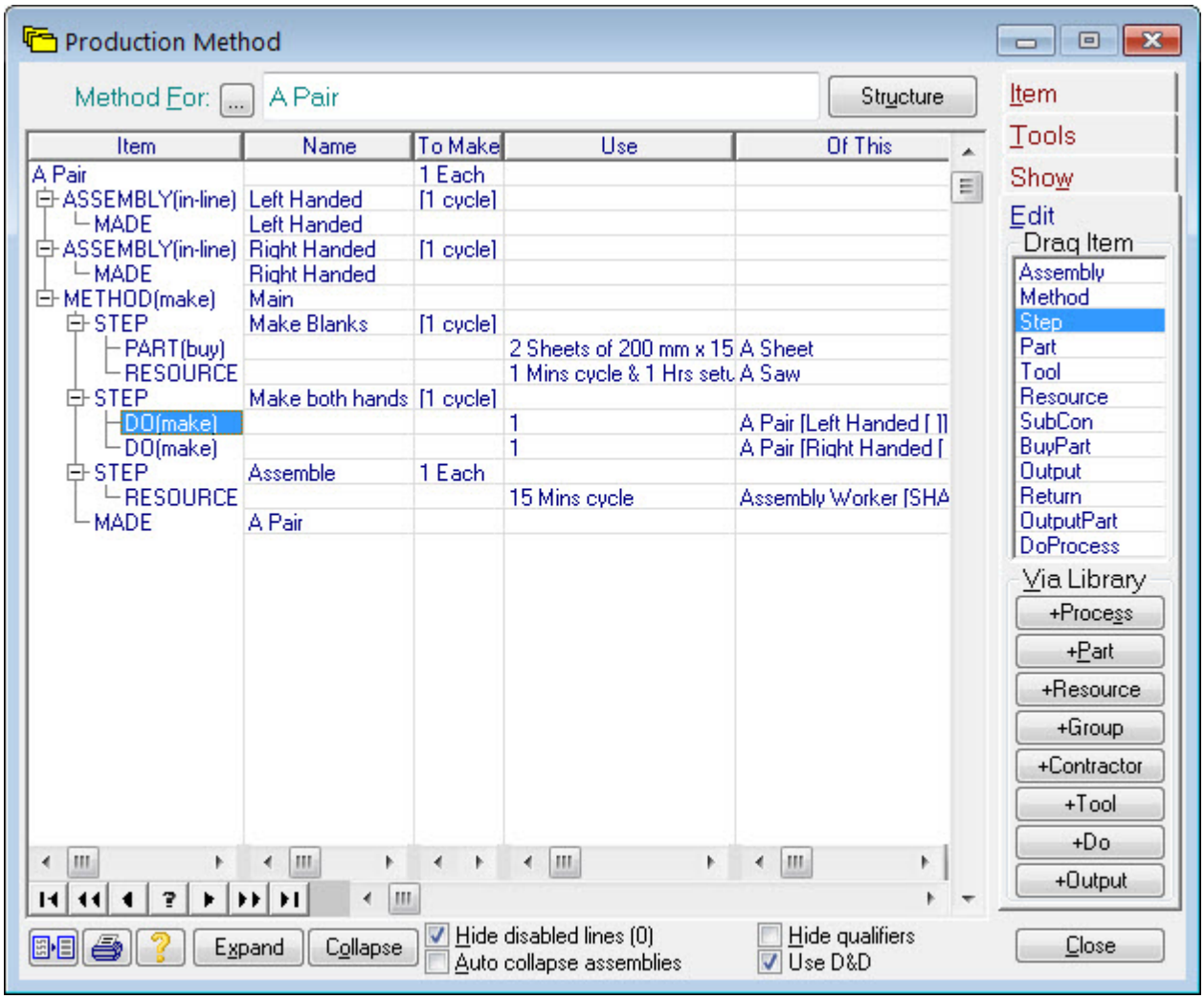
4.6 Tool record

A *tool record* identifies a special tool or jig that is required to perform the operation. A tool is very similar to a PART. The only difference is that a tool is not used-up by the operation (unless you break it) and is given back afterwards for re-use the next time. The benefit of using TOOL records is that it prevents you trying to run more jobs in parallel than you have tools for.

4.7 Do record

A *do record* is very similar to a part record. The difference is that the acquisition of a *do* item does not start until the whole step it's within is ready to start. By contrast, the acquisition of parts is organised so that it ends when the step is ready to start.

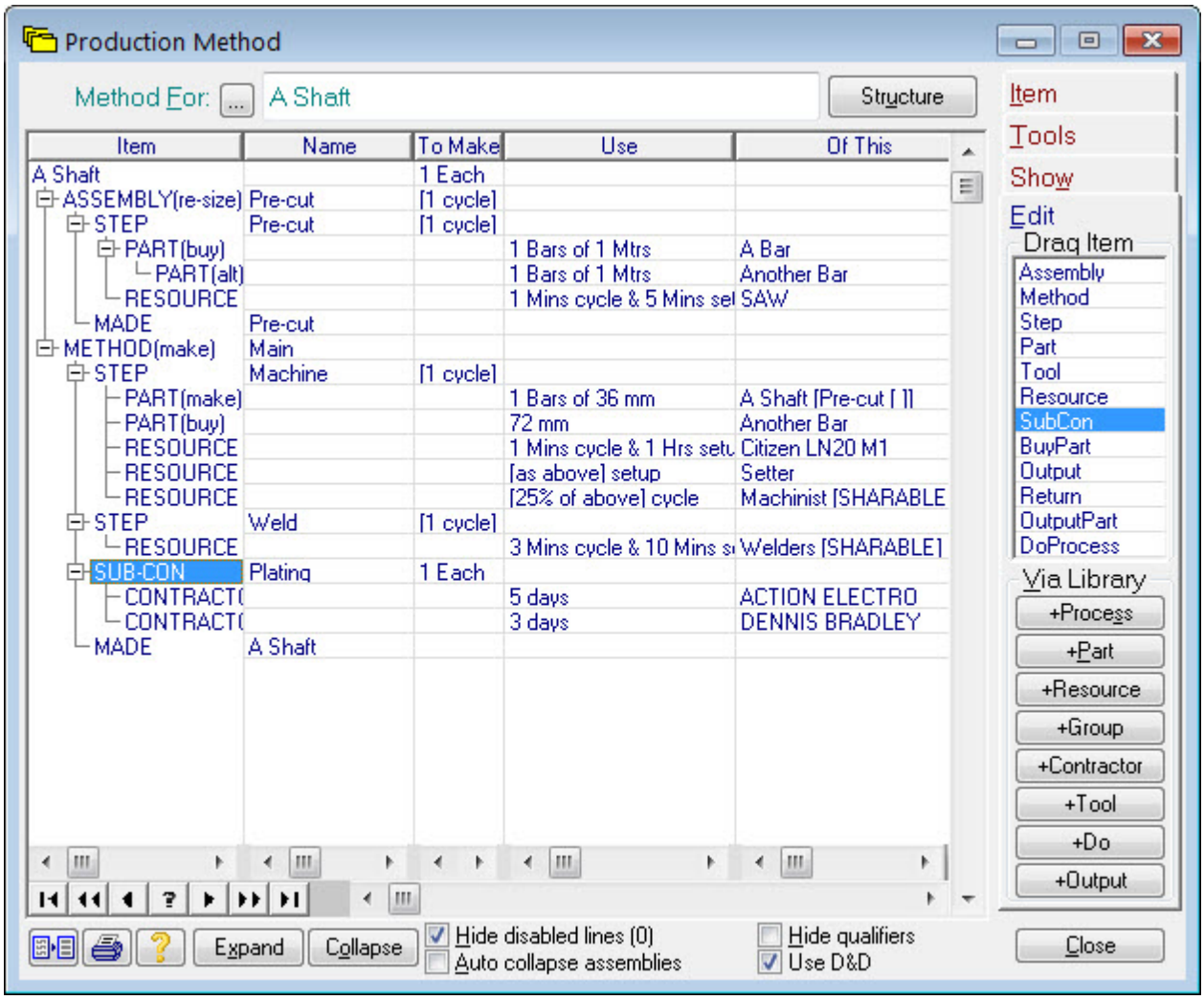
This is useful to force serialisation for the case where the previous step produces something that is required by one or more 'parts' in this step. For example, say you make something as a left and right pair and each half of the pair can be made in parallel, but both need a common part to be made first. The following example achieves this:



DO records can have *First Alternative* and *Next Alternative* variants in exactly the same was as *part* records.

4.8 Sub-con

A *sub-con* record identifies an operation that you are sub-contracting to an outside party to perform on your behalf. Any PARTs defined under the sub-con as well as all work up to the sub-con operation are free-issued to the chosen contractor along with a purchase order describing the work to be done. There can be any number of contractor choices to perform the operation. Unless you've specified otherwise, the planning system will choose the cheapest contractor when there are choices.



In the example above there are two choices on the *Plating* operation, one taking 5 days and the other 3 days. Unless the 5 day choice is cheaper, the planning system will choose the 3 day option because it's quicker.

4.9 Buy part

Dragging a *BuyPart* from the drag item menu will create a BOUGHT PART assembly. These are the same as bought raw materials as they appear in your materials catalogue. The benefit of defining them within the method like this is that you do no clutter your main catalogue with (possibly) very specialist items that are only used in this method.

4.10 Output

An *output* record identifies a side-effect that is being created as part of the operation. The material referenced and its quantity specifies what is created and how much of it is created. See the output assembly description given earlier for an example usage.

4.11 Return

A *return* record also identifies a side-effect that is being created. The difference is that a return is an ‘accidental’ side-effect whereas an output is a ‘deliberate’ side-effect. The significance of this is that the planning system will never make something just to get at its accidental side-effects, but it will try to make a thing if it's the only way to make an ‘output’ it requires elsewhere.

4.12 Output part

Dragging an *Output Part* from the drag item menu will create an OUTPUT PART assembly. See the OUTPUT assembly description given earlier.

4.13 Like record

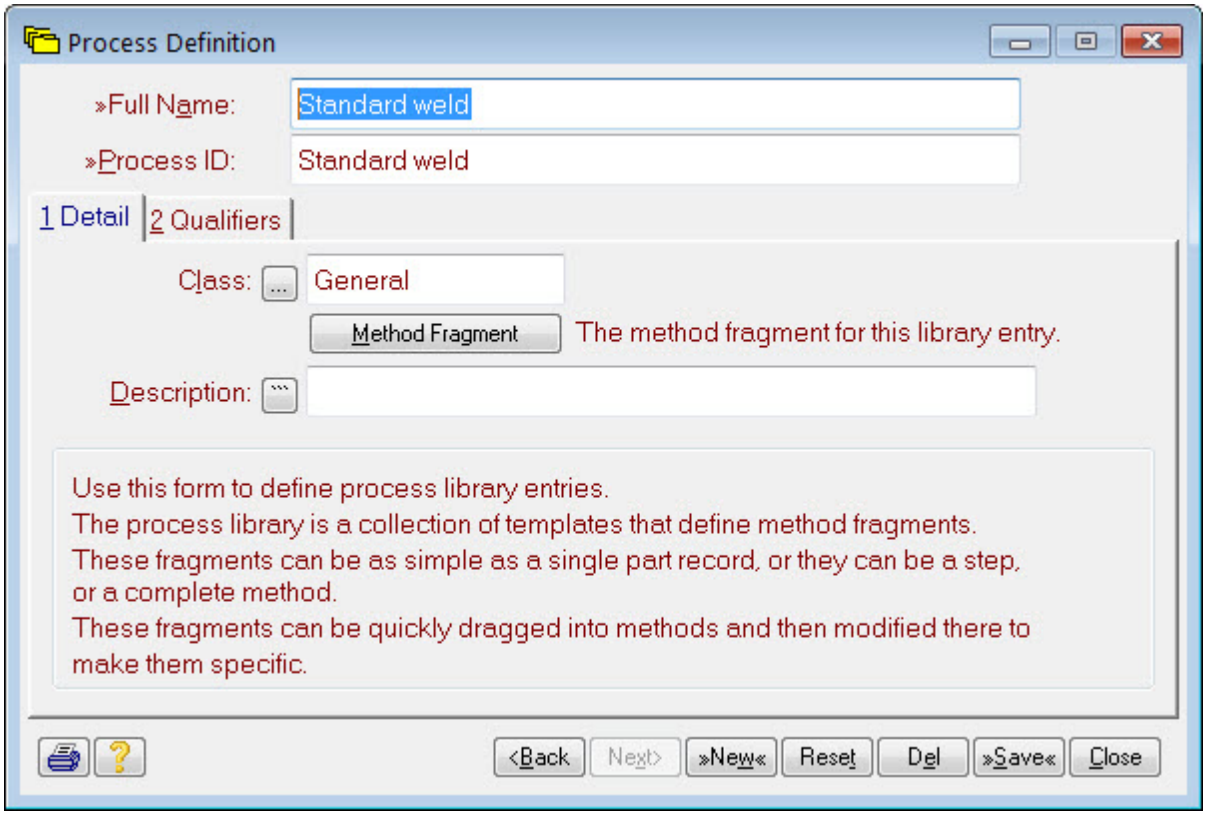
A *Like record* is a reference to a process library item. When the method is evaluated (by the scheduler) it will behave as if the library item referenced was inserted directly in the method. The library item referenced can be any method fragment that is meaningful in the context in which it appears.

A *Like record* is most useful for processes that are constant, e.g. a sub-contracted process such as painting or plating. If you use a *Like record*, then there is only one thing to update if the sub-contractor changes their prices.

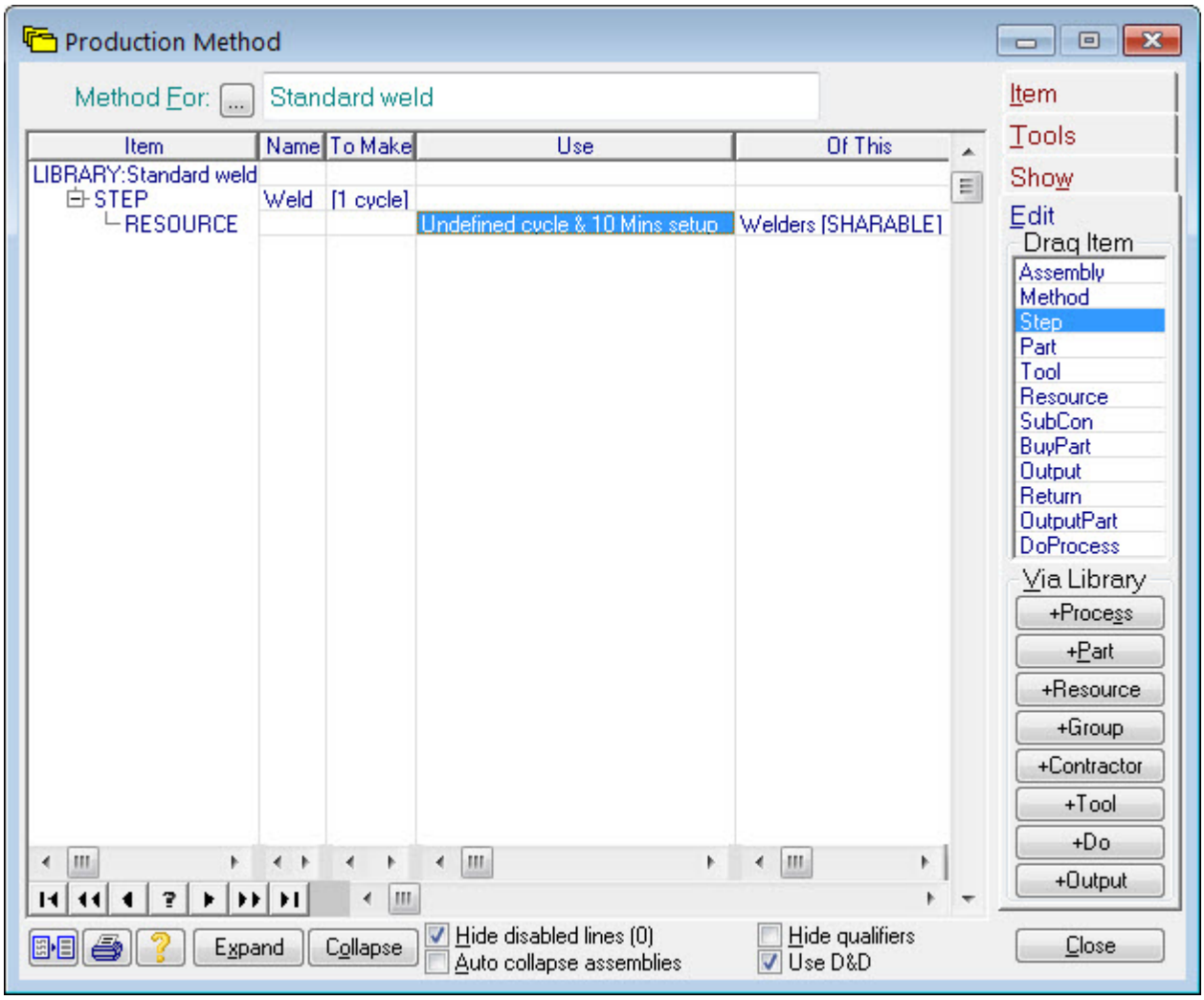
5 Using the Process Library

The process library can be used to hold useful method fragments that you use frequently. Elements from the library can be dragged and dropped into your methods as needed (as a copy that can be edited to suit), or they can be referenced via a Like record (if they are constant).

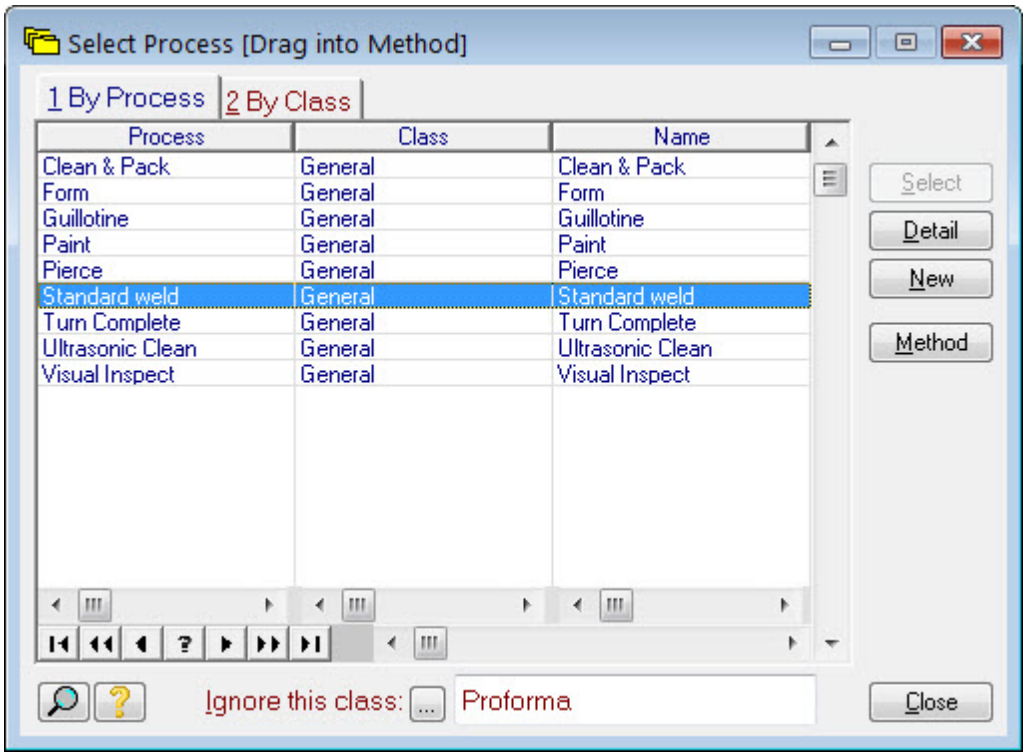
Library entries are constructed using the same method editor. For illustration we'll create a library entry that represents a welding operation. The simplest way to start is to press +Process when in any method. This will open the library selector. Then press New to create a new library entry. A form similar to the one below will open that allows you to give your library entry a name and a description:



Give your entry a name that means something to you and then press Method Fragment. An empty method editor window will open. Create your library entry just like any other method. The entries do not have to be complete, they are purely a mechanism to minimise the work required to construct your real methods. Below is how our welding entry could look:



Notice we've set the cycle time in this fragment as *undefined*. This is useful as a visual cue when the fragment is used to remind you that the actual time should be filled in. Notice also that there is no *METHOD* line in this fragment. That's because this fragment represents just a single STEP that will be dragged into some other method. If you close this editor and the process definition form, you'll find your new library entry has appeared in your process library:

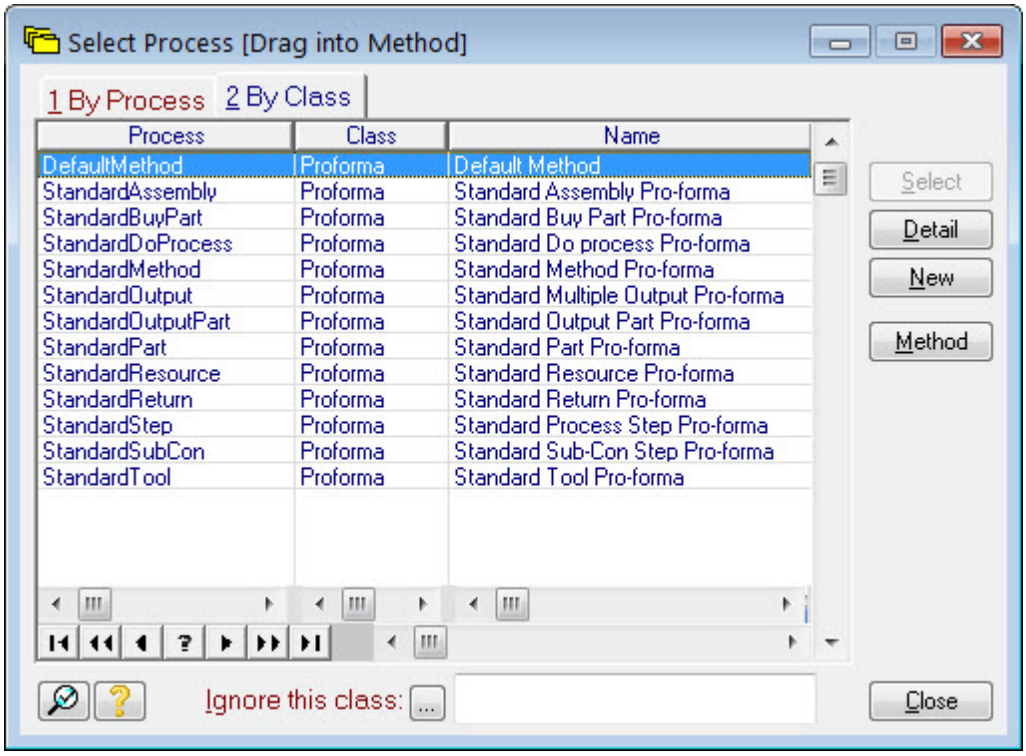


To use it in any method, just drag it from here and drop it in the appropriate place in your method. Then just adjust it to suit.

6 Setting Short-Cuts

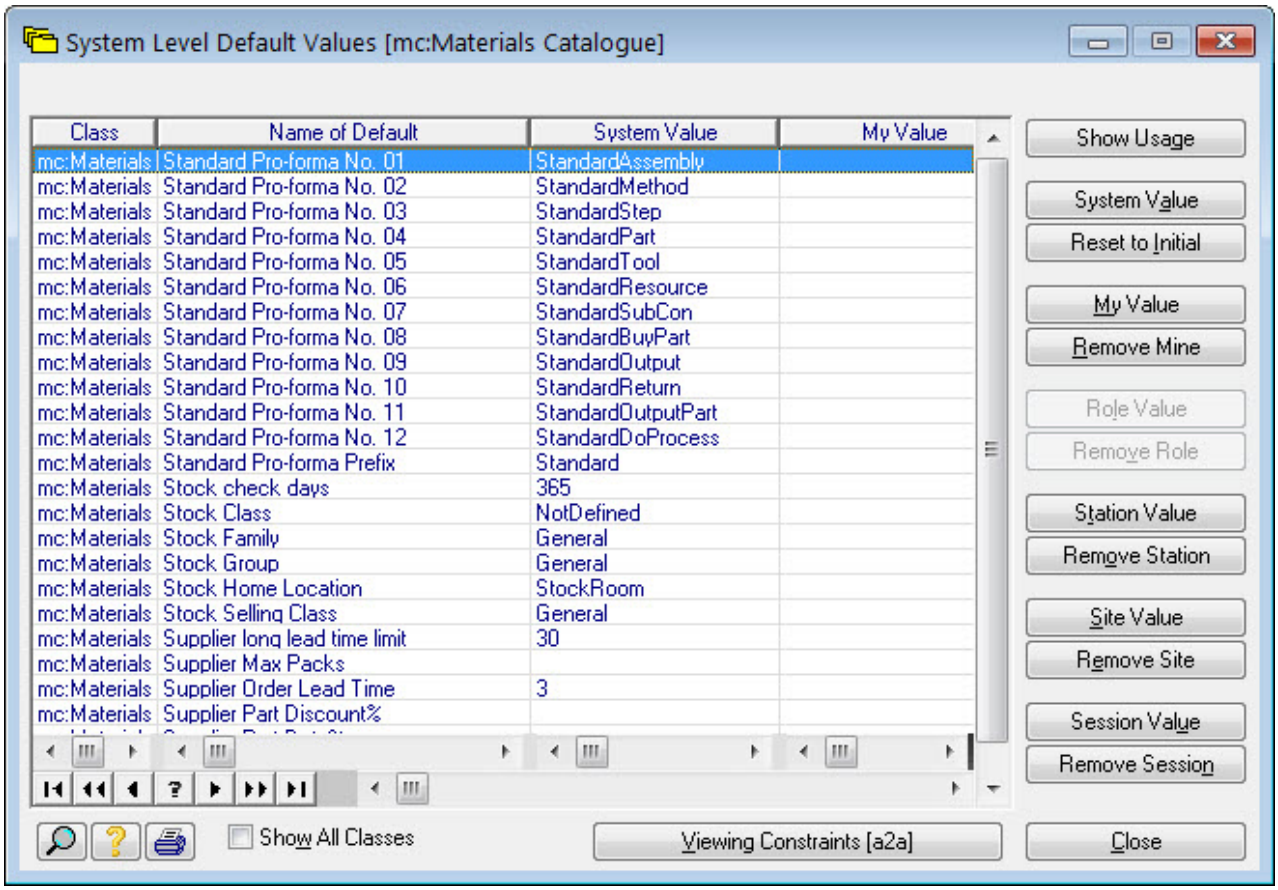
The items 'attached' to the *drag item* entries are just process library entries. You can change both the definition of the standard entries and also what entries are attached to the drag items.

To change the standard entries, press +Process to bring up the process library, then clear the `Ignore this class` field. You will see entries of class *Proforma* will appear. These are the standard method fragments:



They can be edited just like any other process library entry.

To attach a completely different fragment to a drag item list entry, press the **F4** key to bring up the list of defaults, then scroll the list to find the set of defaults with names of the form `Standard Proforma No. ##`:



The value of these defaults is a process library entry. To attach something else, just edit the default. For example, setting `Standard Pro-forma No. 12` to our *Standard Weld* process will result in the drag item list becoming something like this:

7 Techniques

This section just gives brief hints on how to achieve certain things. They are presented in no particular order.

7.1 Parallel operations

STEPs are performed in sequence, one after the other. If your operations can be done simultaneously (resources permitting) then define each as an assembly and combine them at the end, e.g.

```
METHOD (sequential)                    ASSEMBLY 1
      STEP 1                           ASSEMBLY 2
      STEP 2                           ASSEMBLY 3
      STEP 3                           METHOD (parallel)
                                             STEP
                                                   PART Assembly 1
                                                   PART Assembly 2
                                                   PART Assembly 3
```

It is important to appreciate how Match-IT treats STEPs when planning jobs. Consider a method that looks like this:

```
METHOD
      STEP 1
            ...
      STEP 2
            PART A
            PART B
            ...
```

From a planning point of view, the above is equivalent to this:

```
ASSEMBLY 1
      STEP 1
            ...
METHOD
      STEP 1
            ASSEMBLY 1
            PART A
            PART B
```

The significance of this is ASSEMBLY 1, PART A and PART B can all be done in parallel. The fact that PART A and PART B are defined in STEP 2 does not mean they are not considered until STEP 1 is complete.

If this is undesirable, you can use a DO record in place of the PART record. For example:

```
ASSEMBLY 1
ASSEMBLY 2
ASSEMBLY 3
METHOD (parallel)
      STEP 1
            ....
      STEP 2
            DO Assembly 1
            DO Assembly 2
            DO Assembly 3
      STEP 3
            ...
```

In this example, the assemblies will not start until STEP 1 is complete, then they will all be planned in parallel. STEP 3 will only start when all the assemblies are complete. So this is also an example of a process that starts off as common, splits 3 ways, then comes back to being common again.

7.2 Method choices

If the way you make something depends on how many you are going to make, you can define them all as choices and get the planning system to choose the appropriate one. To do this, define each method choice as an assembly, then use each as a part choice in the main method, e.g.

```
ASSEMBLY small batch
      STEP 1
      STEP 2
      ...
ASSEMBLY big batch
      STEP 1
      ...
METHOD
      STEP 1
            PART Assembly small batch
                  PART(alt) Assembly big batch
            ...
      STEP 2
      ...
```

The planning system will evaluate both the *small batch* and *big batch* options and choose the best for the situation. This technique also works using DO records in place of PART records.

7.3 Nesting (multiple outputs)

If you press/cut/mould several different components from a sheet, you can model this in your methods via multiple OUTPUT records. For each 'thing' being created, use an OUTPUT record that refers to an appropriate material record, e.g.

```
METHOD
      STEP Mould
            PART 1 sheet
            RESOURCE Moulding machine
            OUTPUT(20%) 1 Each, Left fascia
            OUTPUT(20%) 1 Each, Right fascia
            OUTPUT(60%) 2 Each, Top cover
      MADE A container
```

The method the outputs are embedded within is referred to as their *container*. This is produced as well as the outputs, although you may not have any use for it and it may not have any residual value (as in the above example).

A demand for any of the outputs of a method will cause the planning system to invoke the method for its container.

7.4 Sharing resources across multiple jobs

You can model operators that can run more than one job at a time using the 'sharing' mechanism of the planning system, e.g.

```
METHOD
      STEP Machine
            RESOURCE Machine, 5 mins cycle
            RESOURCE Operator, 1 min cycle
```

Here the machine takes 5 minutes per thing but the operator is only needed for 1 minute of that, so he has 4 minutes available to do other things. If the operator resource is marked as sharable, the planning system will run up to 5 of these jobs at once using the same operator.

Internally the planning system tracks elapsed time and allocated time within that. If the allocated time is less than the elapsed time there is spare that can be allocated to other jobs. The planning system keeps allocating time until the whole elapsed time has been used up. The elapsed time is set by the resource that requires the most time (the Machine in the example above).

7.5 Sharing work across multiple resources

If you have an operation (i.e. STEP) where the elapsed time can be reduced by throwing more resources at it, you can use one of three mechanisms to share the work across more resources. Each has slightly different effects and consequences. The three mechanisms are:

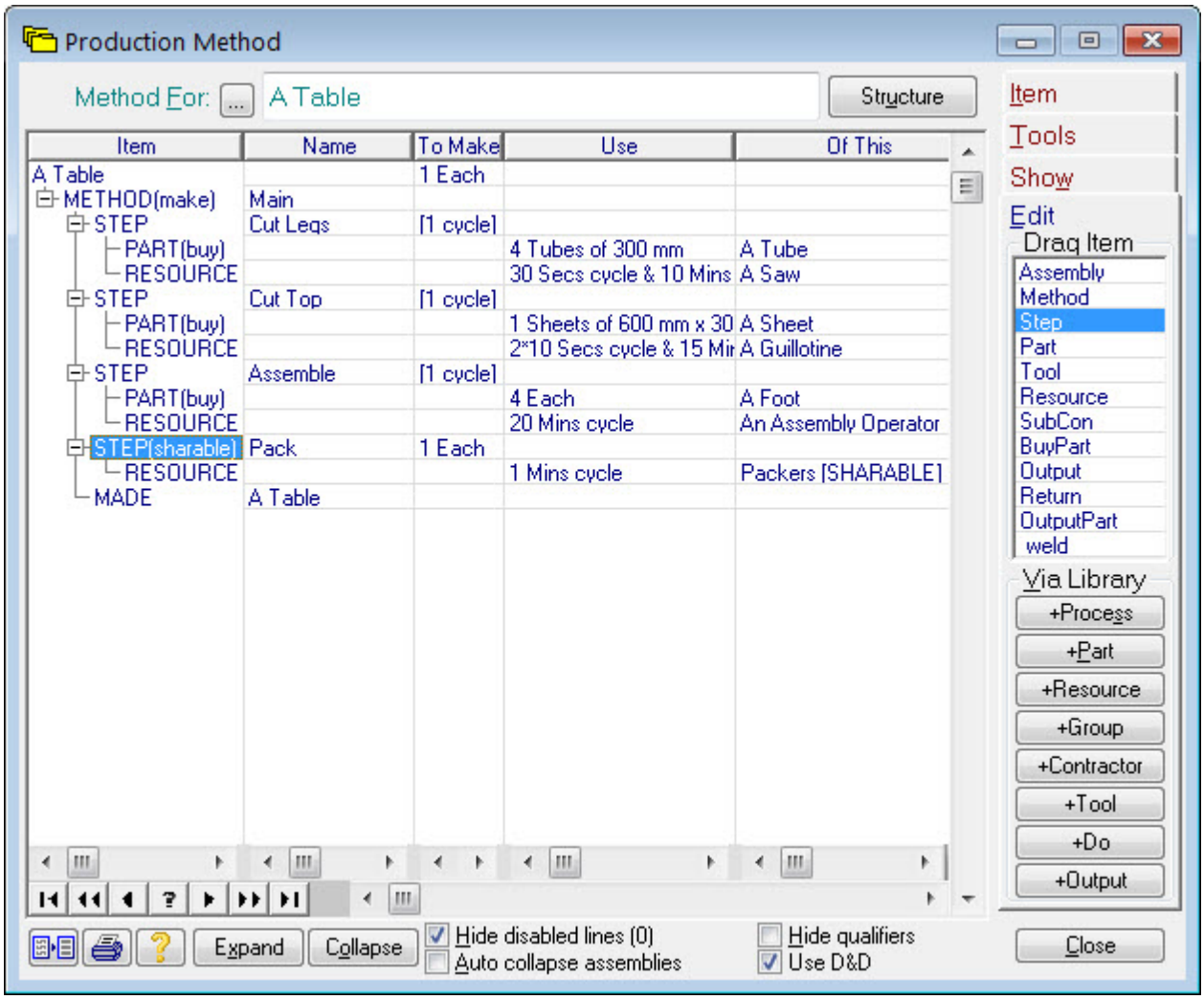
Shared steps At least one RESOURCE in the step must be sharable and have choices (either explicit or by using a group). For such resources, **every available choice is used** at once and the work shared amongst them. The share given to each choice will be proportional to their available time. This is a very dynamic mechanism because **how many resources are actually used is dependent on their availability at the time**. To make a step shared you just check the "Work is sharable" check box in the STEP record (see example below). Shared steps are most useful for operations involving just people.

Max batch size This mechanism limits the maximum quantity that can be processed in one 'lump'. The max batch size is set in the reference information for the step. To access this from a method, double-click on the STEP record, then press "Advanced Edit", then select the "To Make" tab, then press "Edit Reference Info", then select the "Stock" tab, then fill in the "Max Batch Size" field. If a max batch size is set for a step and the quantity being planned is in excess of that, then the planning system breaks the step into 'lumps' of the max size and a final lump that is the residue. For example, if the max batch size is 100 and the planning system is planning for 250, it will break it up into two batches of 100 and a residue of 50. Each split batch is created independently, **each will have its own resources, its own tools and its own kit**. This means **all the preceding steps are duplicated**, so if step 2 has a max batch there will be a separate step 1 for each split batch. This is similar to the max batch constraint on a product except all the batches are processed within the same works order and feed into the same output batches (and so have a coarser batch traceability trail). This mechanism is only useful if there are resource choices available, so at least one RESOURCE in the step should have choices, then each batch can pick up a different choice to allow the batches to be run in parallel. If there are no (more) choices, the additional batches just get serialised waiting for a choice to become available.

Lot size This is similar to the max batch size mechanism except **all batches (or 'lots') are the same size and share the same kit**. The latter means there is no preceding step duplication. Each lot is given its own resources and its own tools, but all share the same kit. This means all the kit must be available before any of the lots can start (unlike the max batch size mechanism where each batch could start as soon as their share of the kit is available). The lot size is calculated by dividing the total quantity to be done by the number of lots to spread it across. The number of lots is calculated as the total quantity divided by the lot size rounded down. If this exceeds the maximum number of lots you specify, then its limited to that. An example: if the lot size is 100 and the planning system is planning 250 then the number of lots will be 2 (250/100=2.5 rounded down to 2) and each lot will be for 125 (250/2). To set a lot size, double-click on a STEP record, select the "Use" tab and fill in the "Lot size" and "Max lots" fields. Again, to be useful, the step must contain RESOURCEs that have choices to allow the lots to be run in parallel.

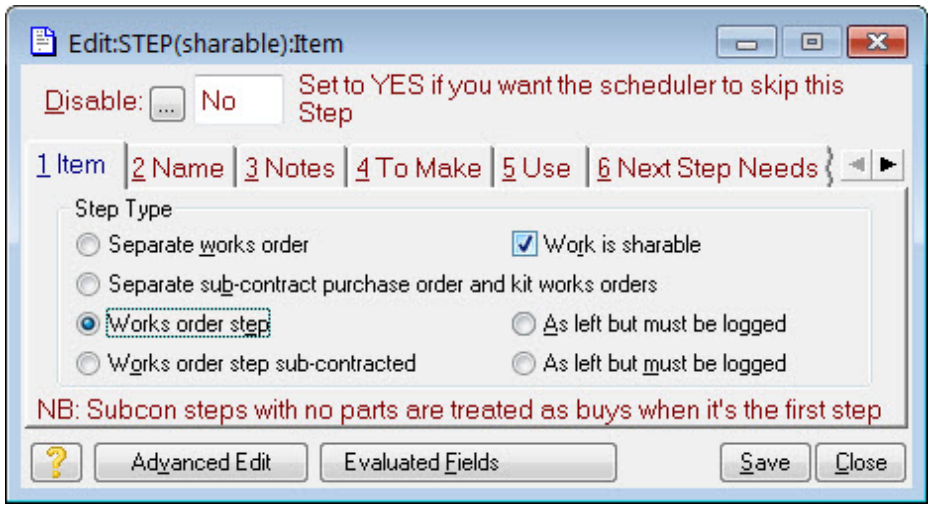
Note: In all these mechanisms, your costs will go up if any of the RESOURCEs involved have a setup time, as each batch or lot will incur its own setup overhead.

Example of using a shared step:



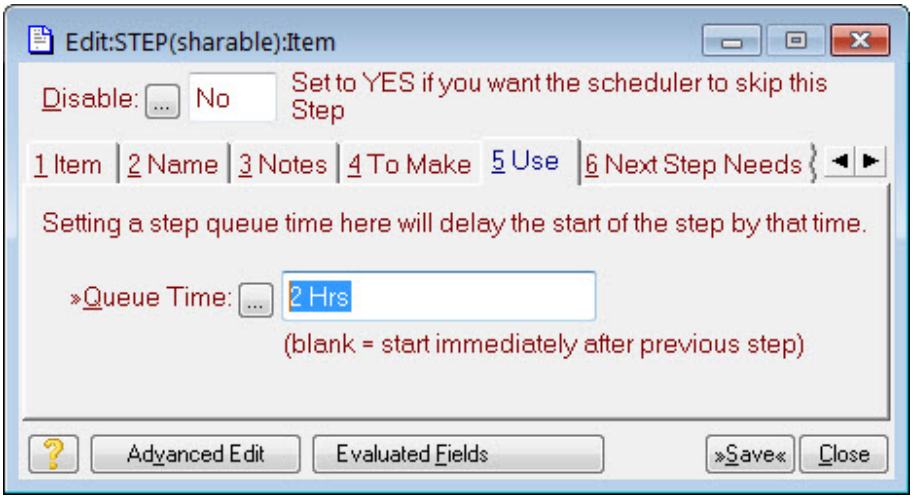
The *Pack* step here is a 'sharable' step that is using a resource group called *Packers*. Marking the step as sharable like this means the planning system will make use of every available member of the Packers group and share the work out amongst them to reduce the elapsed time of the operation.

To make a STEP sharable just check the ☑ Work is sharable option on the form that comes up when you double-click on the STEP word:



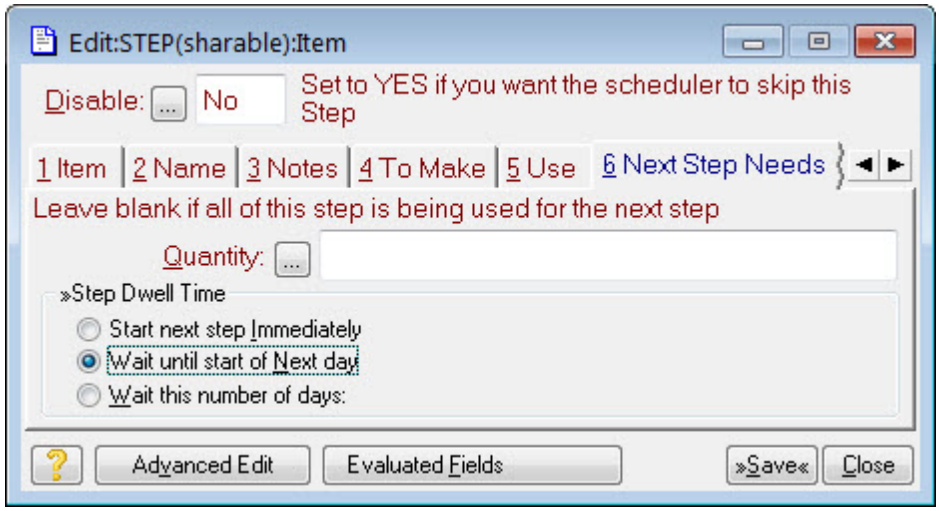
7.6 Queue time

Use a *queue time* to model logistic delays incurred when moving work from one workstation to the next.



7.7 Dwell time

Use a *dwell time* to model cool-off periods after a process, e.g. wait overnight for an oven to cool down before it can be opened.

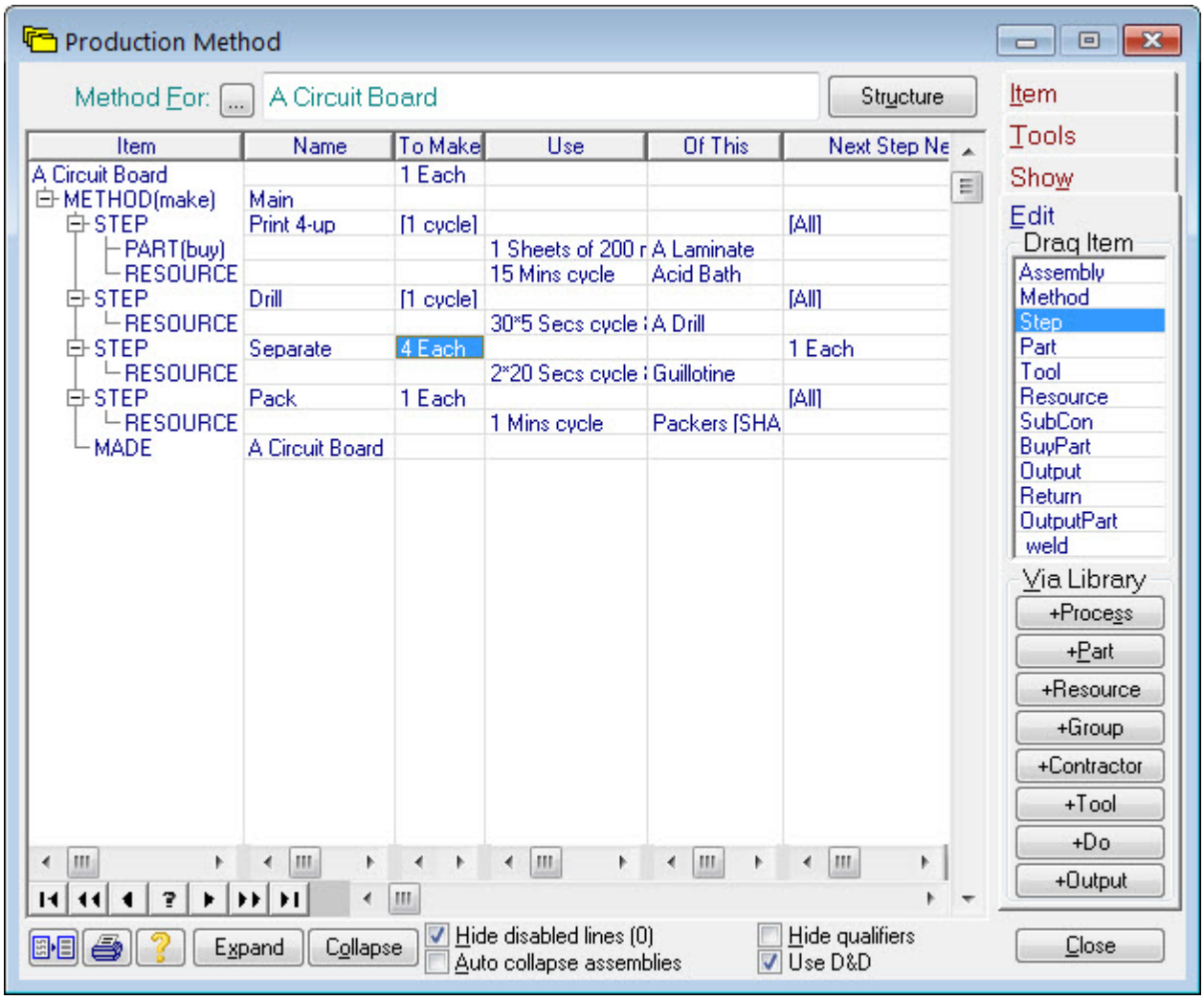


7.8 Disabling

Almost every element of a method can be disabled. A disabled element behaves as if it is not there; the planning system completely ignores it. This is useful to turn options on and off. Disabling an element will affect all jobs that are planned against the method in that condition. It will not affect jobs already planned and already started, but it may affect jobs already planned but not yet started.

7.9 Multi-up operations

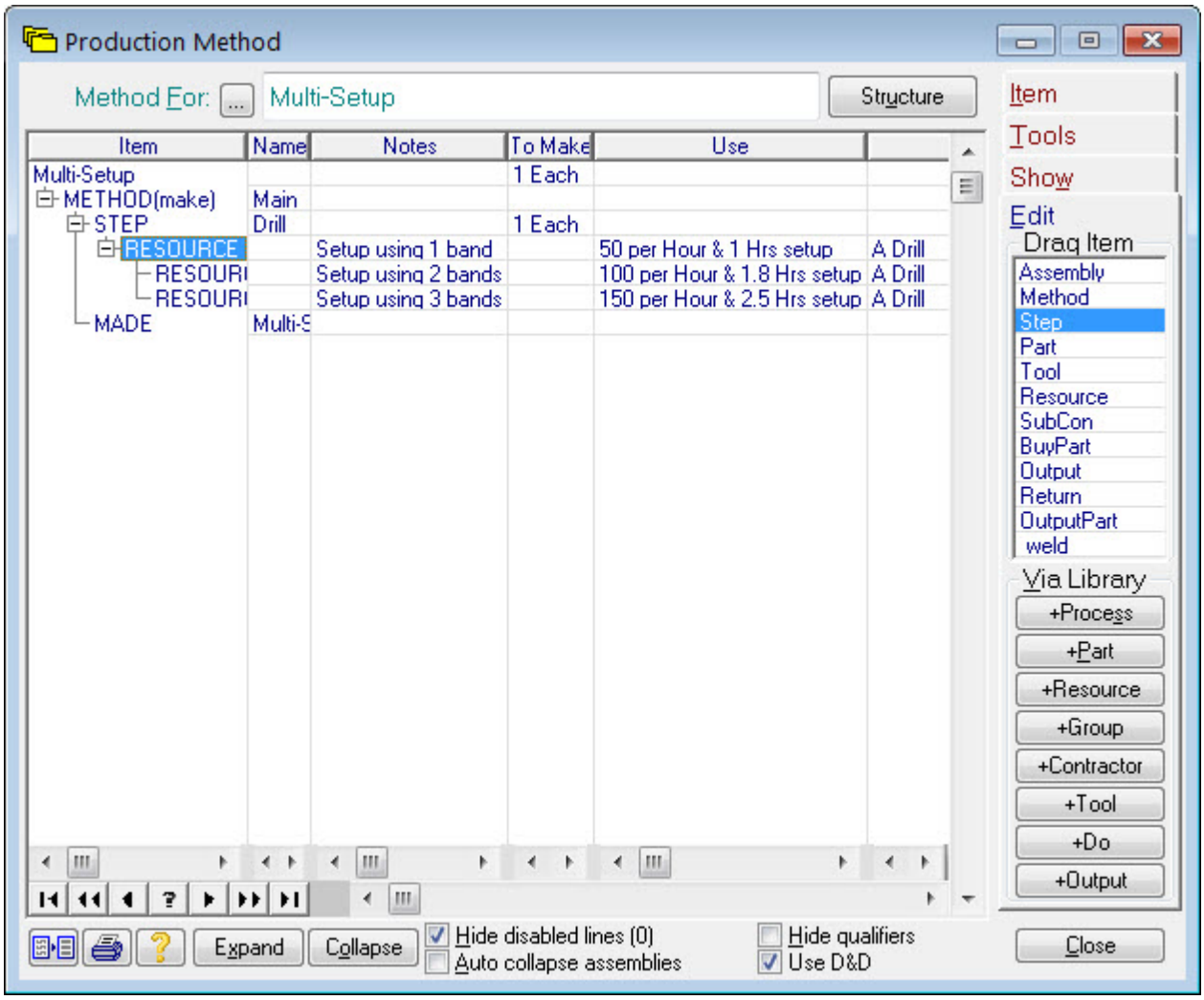
If you have an operation that produces more than one of something, e.g. a small printed circuit board where many are printed at once then finally cut up into the individual circuits, this can be modelled by specifying the *To Make* and *Next Step Needs* quantities for a step, e.g.



Here the *Print 4-up* step prints the full board, it's then drilled still 4-up, then separated into 4 separate pieces in the *Separate* step. That step also specifies that 1 of those 4 is required for the next step.

7.10 Multi-setup configurations

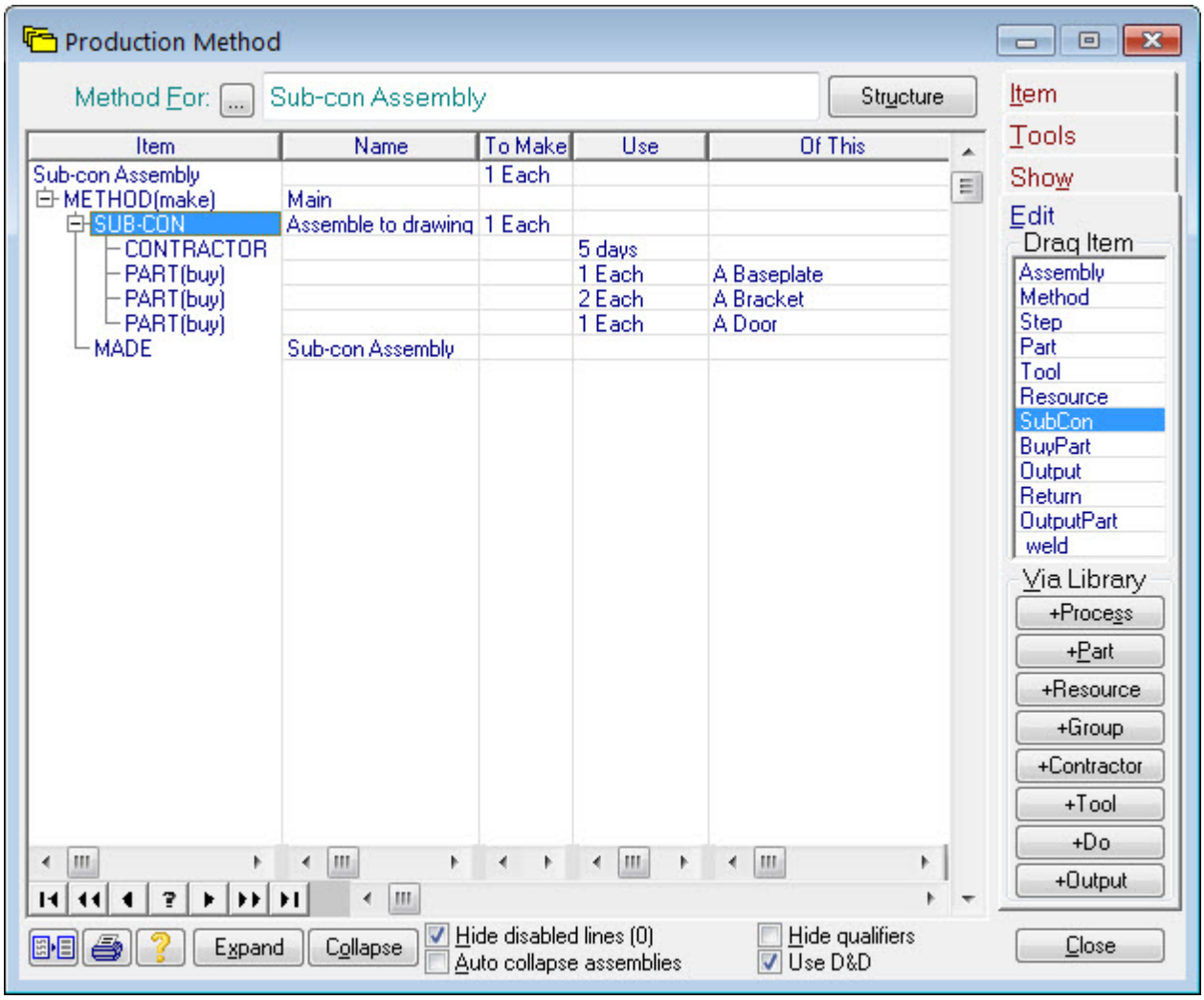
If you have a machine that can be setup in different configurations depending on the quantity being processed, this can be modelled by using the same resource as alternatives but specifying different setup and run times for each, e.g.



In the above example, the same resource is being used in each of three alternatives (A Drill), but its throughput is different in each case. In this example the higher the throughput the longer it takes to set. This means for any particular quantity to be processed there is an optimum setup. Specifying the alternatives as above, allows the planning system to find and choose the optimum.

7.11 Sub-contracted assembly

If you have a product that is assembled on your behalf by a sub-contractor and you supply the parts to them, this can be modelled by a method consisting of a single sub-con step with all the parts specified in that step, e.g.:



The product must be tagged as “Can be manufactured”, it's still a manufactured part it's just that it's being done on your behalf by someone else. Invoking this method will cause the planning system to co-ordinate the acquisition of all the parts (be they bought or made) and then it’ll raise a purchase order to the contractor describing the work to be done and listing all the parts you are free-issuing to them.

