Chapter 4 – Paramaterisation and analysis

Understanding inherent waste in a process

Many text books have already been written on the subject of waste in lean manufacturing systems. Toyota initially came up with the concept of *Muda* in the 1960s. Improvement in manufacturing performance comes from constantly attacking waste in a process.

What is waste? – Anything that a customer does not pay for? Anything that does not add value to the product? Anything that does not add perceived quality to the product? In the world of mass production of components for automobiles, Known as "Muda" in lean manufacturing , To there are 8 wastes to be combatted:

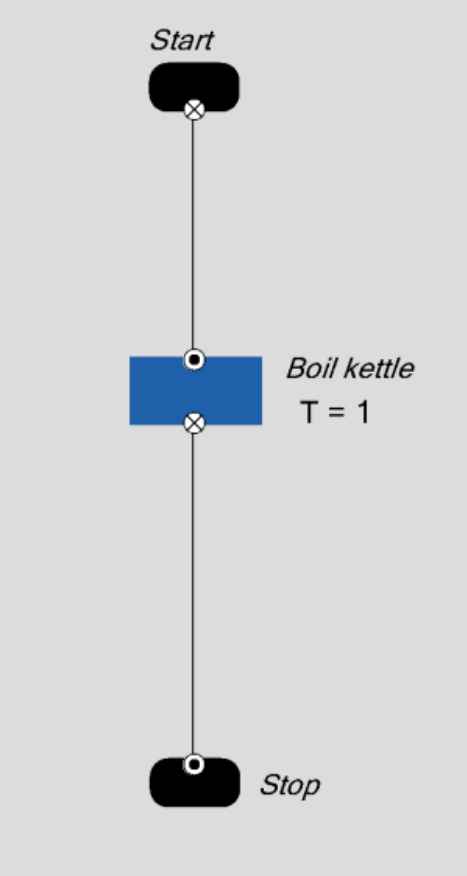
D	Defects	Effort caused by rework, scrap and incorrect information
O	Over production	Production of more than is needed or production or before it is needed.
W	Waiting	Wasted time waiting for the next step in a process
N	Non utilised talent	Not making full use of peoples talent, skills and knowledge
T	Transportation	Movement of goods between locations both within and external to a production facility.
I	Inventory	Excess products – Anything being stored anywhere
M	Motion	Unnecessary movement by people, machines, robots
E	Extra processing	Adding processing which results in a higher quality level than that desired by the customer

In this chapter we will show how Aliquis can be used to understand the extent of various wastes in a process. We will look at transport, motion, inspection and waiting.

System Attributes, User variables, Parameters Assignments and user variables.

Before we can understand the techniques used in this chapter we need to briefly discuss the meanings of these terms and how they relate to each other.

Open the model Chap4Ex1



System Attributes.

Hopefully you already understand these. System attributes are variables which are used to drive the behaviour of blocks. So, for example, in a delay block – T is a variable the defines the delay time. You have also seen that you control what you see of the system attribute. It can be invisible, or you can show any of the name, description, driving expression, value and choices.

User Variables and paramaterisation

User variables are similar to system attributes. They can be displayed or hidden from the user in the same way. You can also display different aspects of the variable as you see fit. However, they are not directly connected to the behaviour of a block. They can:

- Be assigned at certain points in the solve
- Collect Results
- Drive block attributes via paramaterisation

Here's an example for *paramaterisation* :

Let's consider the time it takes to boil a kettle for our tea – it actually can be predicted reasonably well from physics if we know the power input, the mass of water and the initial temperature of the water. We will neglect heat losses at this stage.

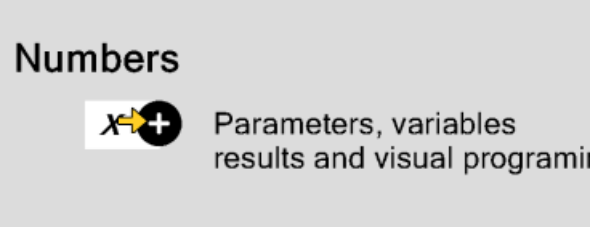
Here is the driving formula :

$$Boiling\ time(s) = \frac{4.2 * (100 - Tap\ Temperature\ (C))\ \ volume\ of\ water\ (cc)}{kettle\ power\ (W)}$$

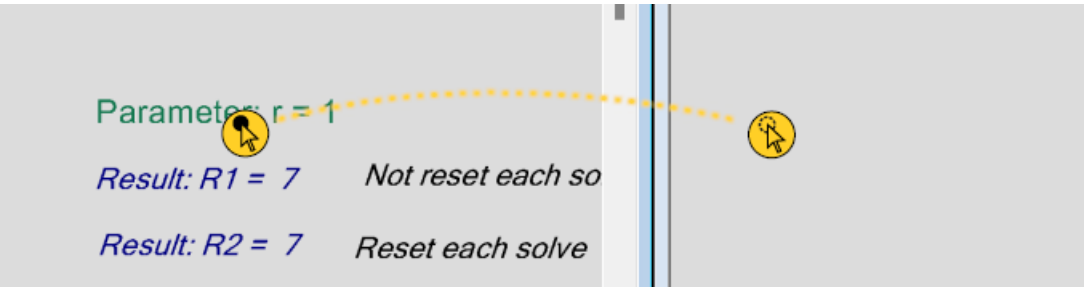
We are now going to create a user variable to serve as a parameter in the top level of our model:

In your main model – make sure you are no longer in the block. (double click in space)

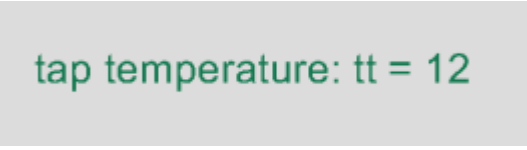
From the top-level library, find and enter "numbers":



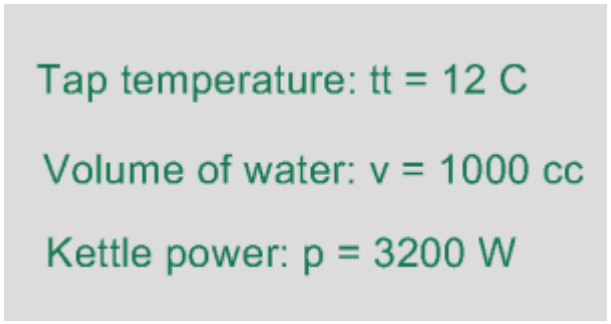
Drag one of the parameters into your model



Change the name and description as follows :

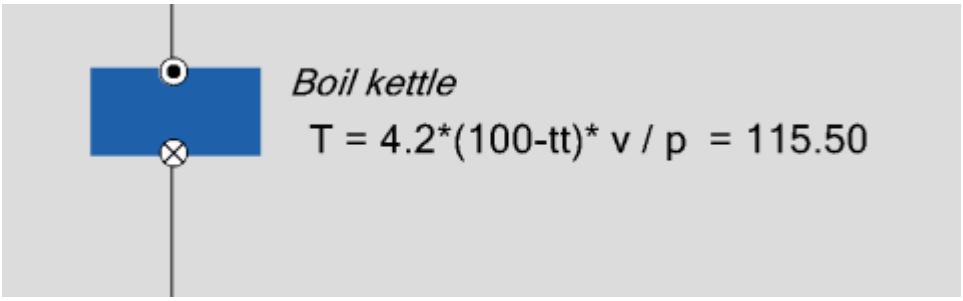


Likewise pull in new variables to make up the other parameters

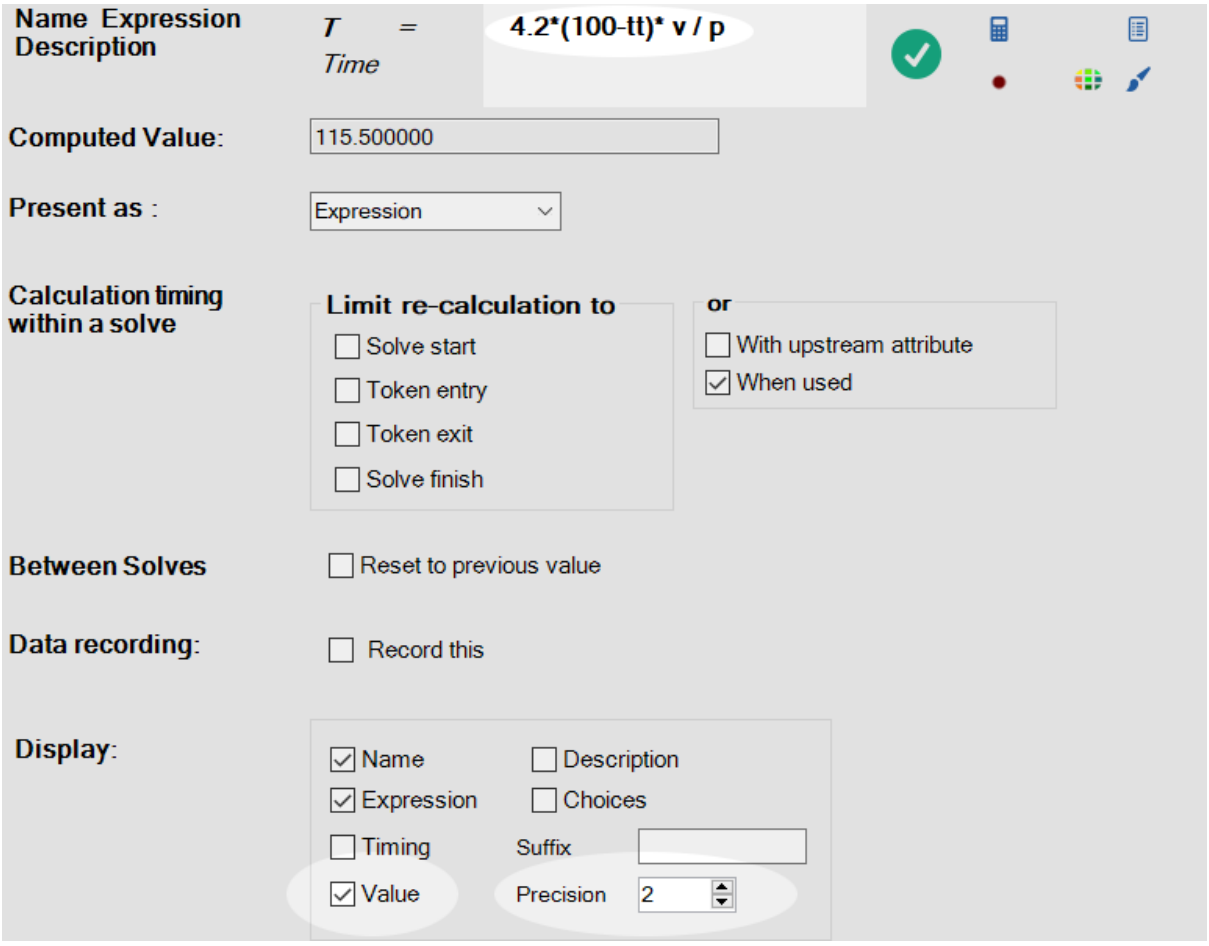


I have also used the suffix box to enter the units here. (open and expand the variable).

Now adjust the formula for the boil time block as follows :



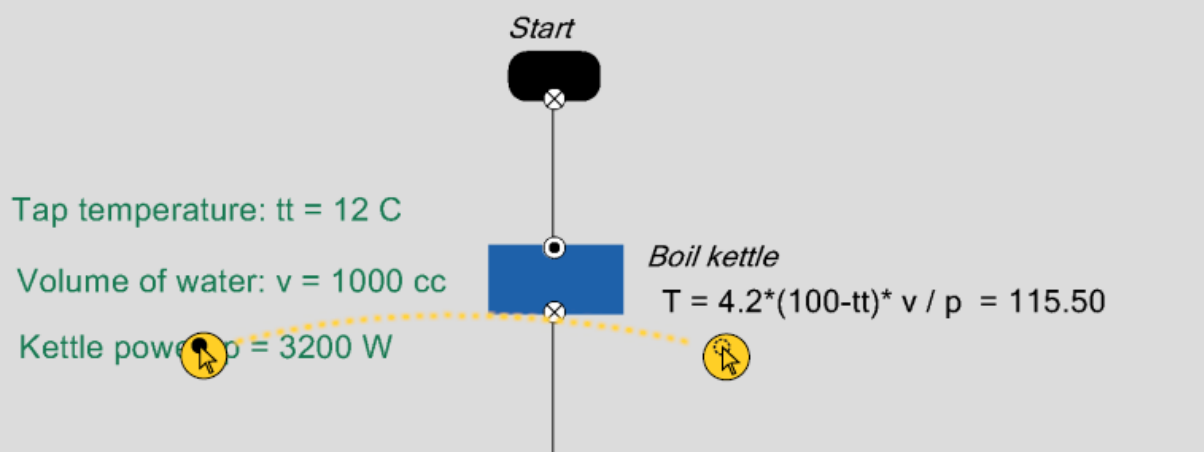
Here is the expanded variable sheet for reference :



IF you change the parameters, you will notice that the behaviour of the block (ie the system attribute T) will change accordingly.

For one final thought on paramaterisation, lets consider the logical location of the parameters. So far we have put them all at the top level of the model. However, if they are *only* going to be used within the block – it is perfectly possible to locate them within the block – Other blocks will not see them. In this case it is arguable that the temperature parameter is "global" whereas the volume and power parameters might belong better in the block.

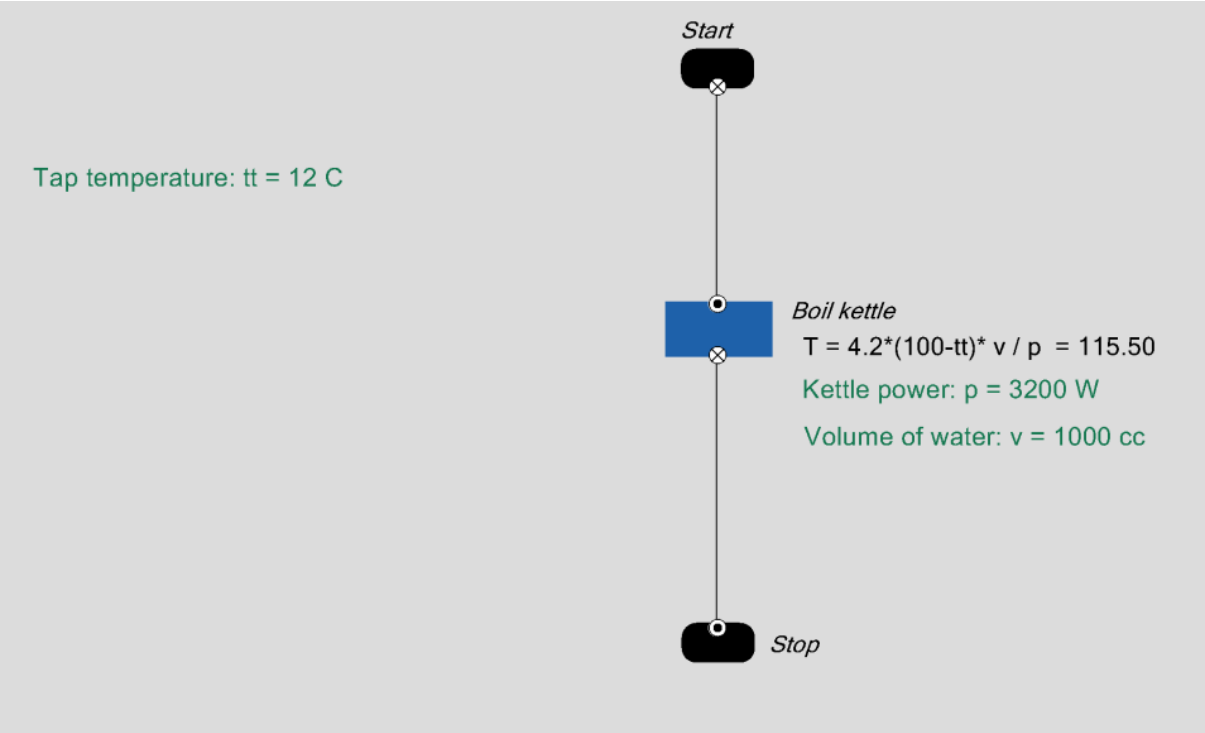
Just drag them into the block to make this happen :



The block should flash if this is done correctly.

Tip. If you are putting a parameter into the system, make sure a block does not flash – its quite easy to put parameters into blocks when you mean to put them into the top level system. If this happens just undo and try again. Drop the parameter into clear space.

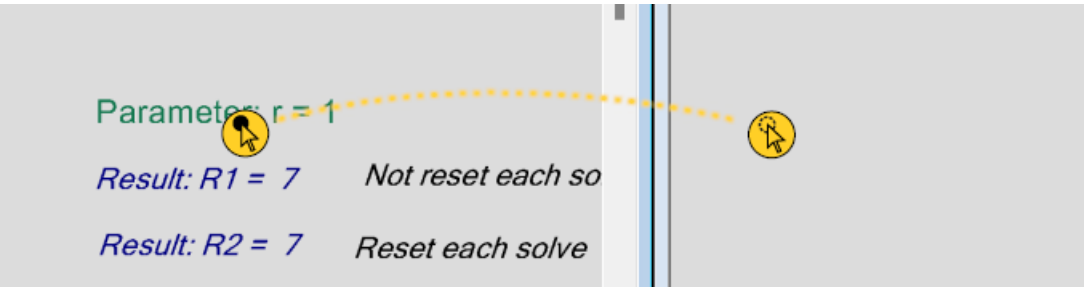
This is the result you are looking for :



Assignments

So far, we have used user variables as parameters to drive the behaviour of a block. We can also use them to hold results. In order to do this they are normally linked to an assignment which is fired at some point when the model is running.

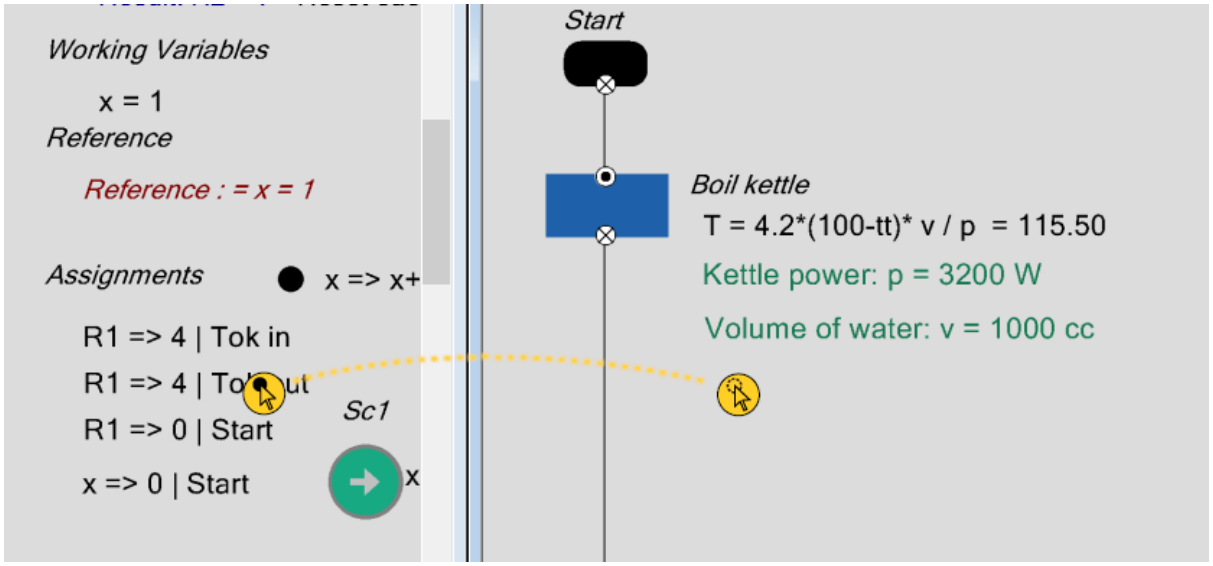
In our simple case we are going to create a result user variable to hold the time when the process finishes.



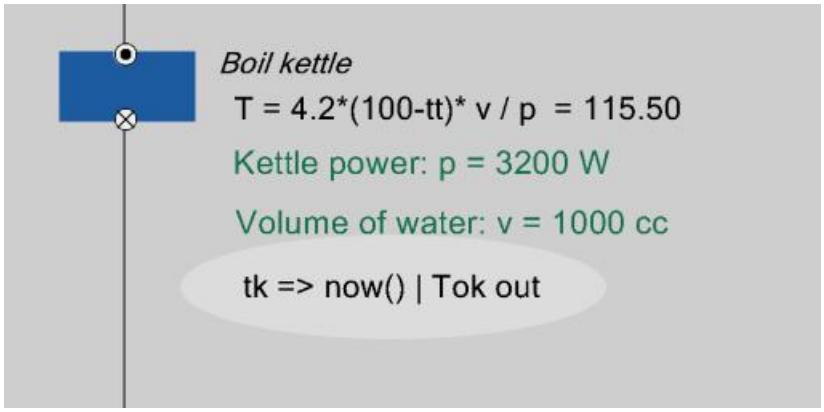
Set it up as follows :

We are now going to assign the simulation time to this variable when a token leaves the kettle.

Drag a token out assignment into the block:

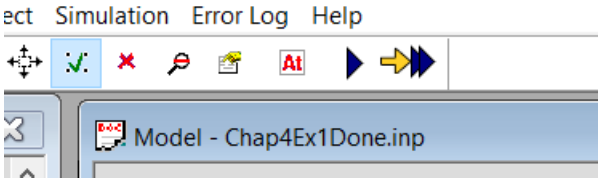


Set it as follows :

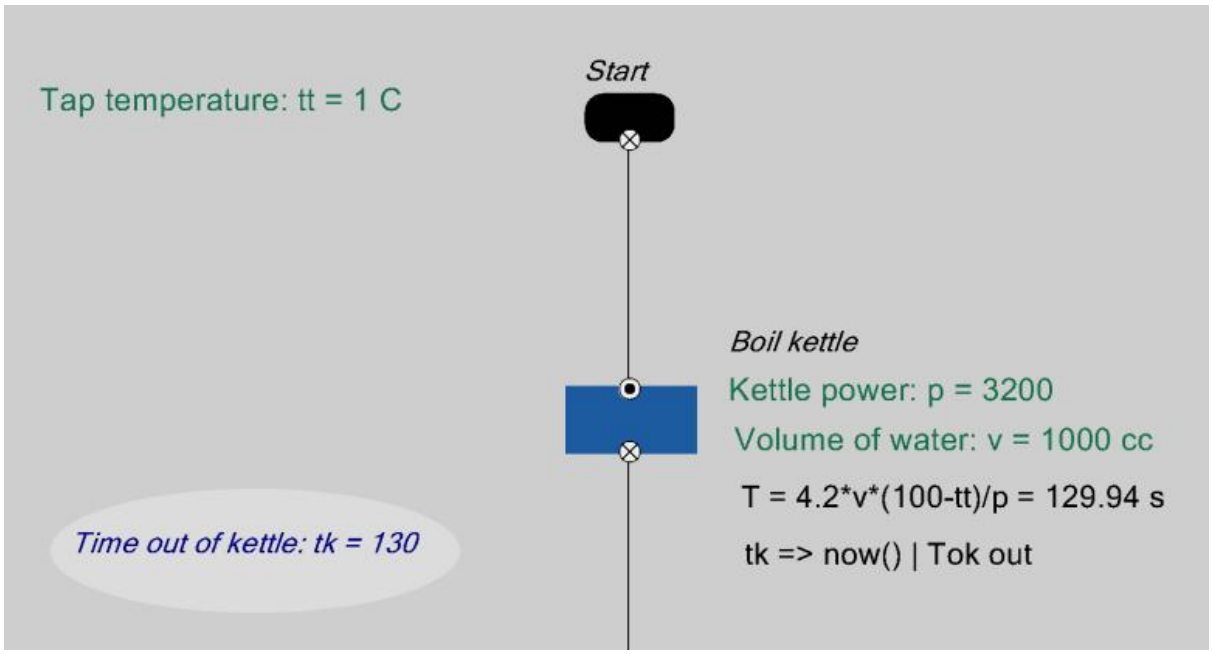


When a token comes out of the block it will trigger an assignment to tk (our result) . The value will be the value of the function now() – which means the simulation time.

A quick Run



And you should get this :



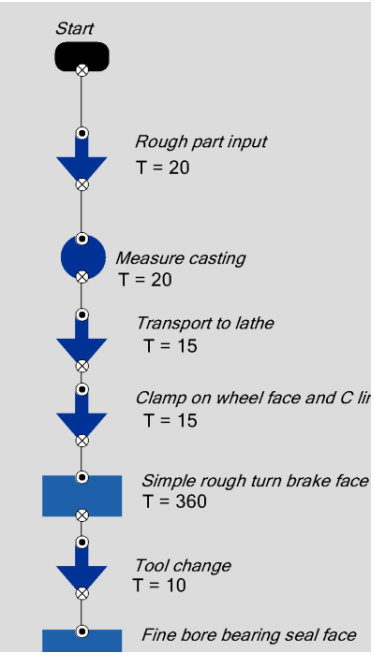
Analysing Wastes in a process

The functionality we have gone through can be used to analyse the process in many ways. The standard library has a sub library for analysing 6 wastes, - transport, movement, inspection rework , scrap and waiting. The methodology embodied in the library is not written in stone. You actually control the Aliquis Libraries and templates… However, this feature has been added to allow you to do a basic analysis easily while you are mastering the software . It also serves to show you how libraries can be tailored to fulfil standard tasks easily.

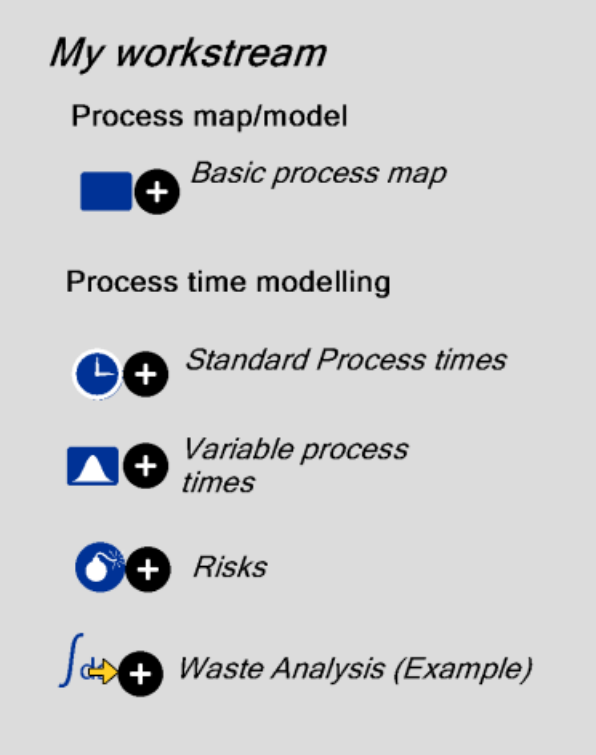
We will analyse the wastes in two stages. Initially we will look at how time is used in the standard process. We will be examining transport, movement, inspection and if we had any, waiting times.

In the second stage we will move on to a basic scrap and rework analysis.

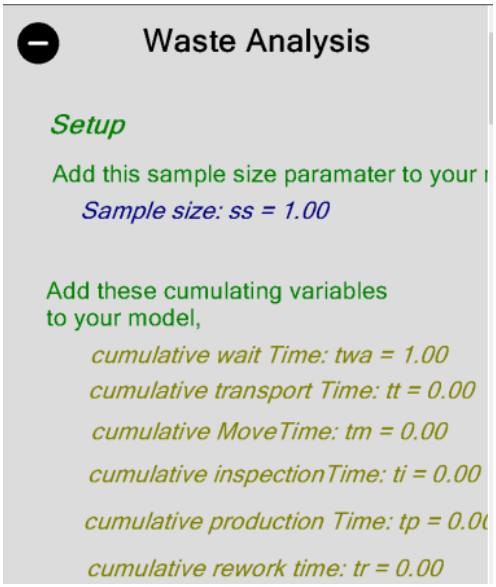
Open the model Chap4 Ex2



Now go into the waste analysis part of the library:



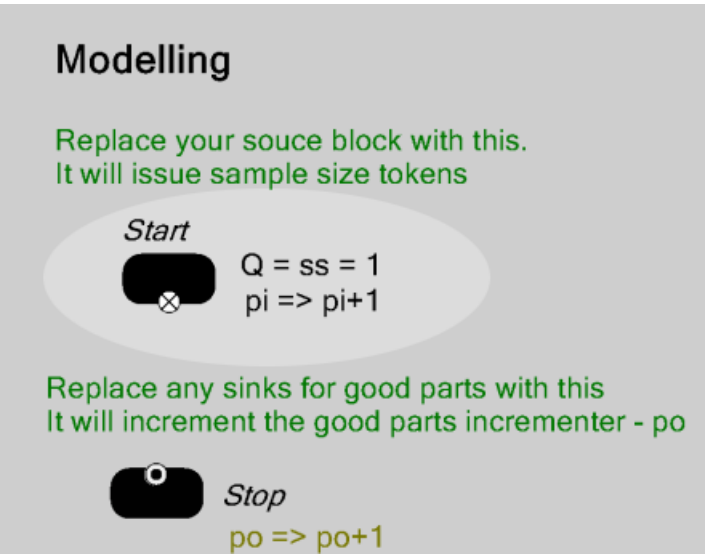
Result



This library has many variables and assignments of all forms for you to use in your model. It also has a few modelling blocks which will be useful to model scrap and rework.

First we will replace the source and sink in the model with the tailored blocks in the library :

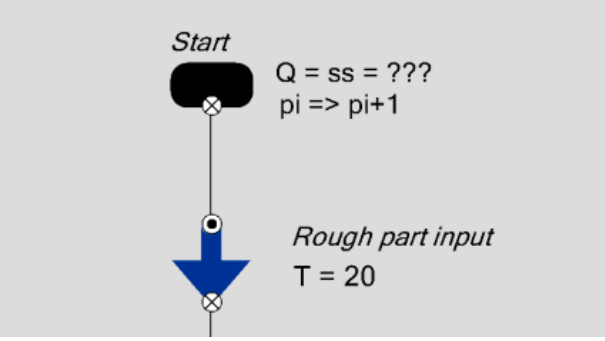
Scroll down to the modelling section you will see this :



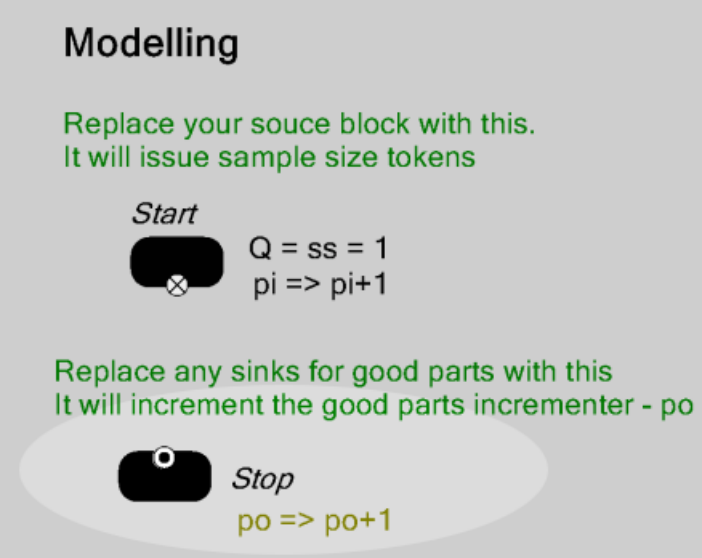
You can see that this is set to issue ss (sample size) tokens and each time it does, the variable pi (parts in) is incremented.

Replace the original source with this source..

Result :

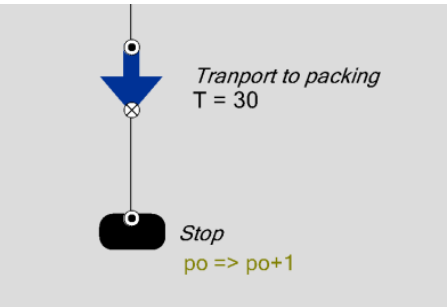


Likewise you will see a sink :



This increments the variable po (good parts out) every time a token arrives

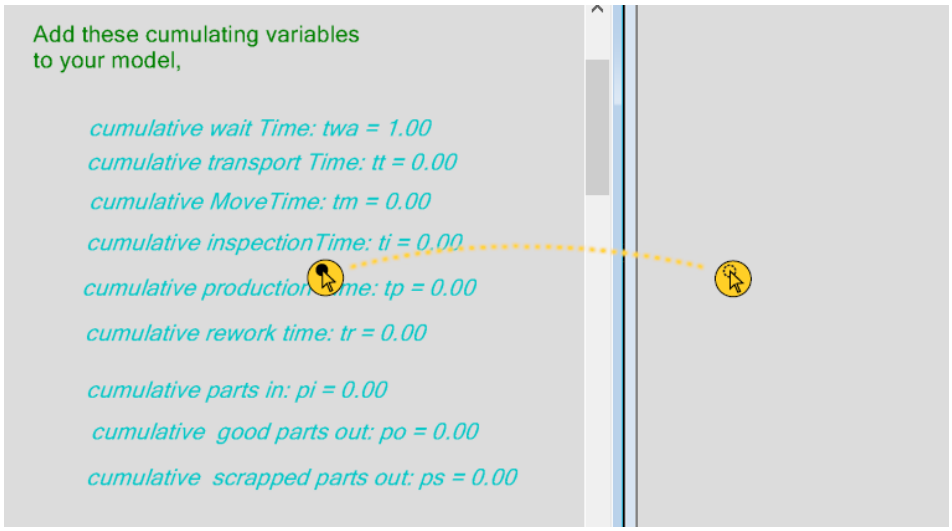
Replace the original sink with this :



Drag and drop all the variables and assignments into your model as instructed :

Don’t forget you can use a selection window to select multiple objects and drag them all over in one command.

And Drag them all into your model



These are the variables which add up the times for the various wastes. (*cumulation* variables)

In all you should drag in 1 parameter, the variables shown above, A set of final result variables and numerous assignments.The assignments initialise and the cumulation variables and finalise the results.

The left hand side of your model model should look like this :

Chapter4Ex1

Sample size: ss = 1.00

cumulative wait Time: twa = 1.00
cumulative transport Time: tt = 0.00
cumulative MoveTime: tm = 0.00
cumulative inspectionTime: ti = 0.00
cumulative production Time: tp = 0.00
cumulative rework time: tr = 0.00
cumulative parts in: pi = 0.00
cumulative good parts out: po = 0.00
cumulative scrapped parts out: ps = 0.00
cumulative actual defects: pad = 56.00
cumulative defects not detected : pdnd = 0.00
cumulative defects detected : pdd = 0.00

Wait time per part: ftwa = 1.00
Transport time per part: ftt = 0.00
Move time per part: ftm = 0.00
Inspection time per part: fti = 0.00
Production time per part: ftp = 0.00
Rework time per part: ftr = 0.00
Final Percent Production time %: pt% = 0.00
First time yield %: fty% = 0.00
Detected Defect Rate %: ddr% = 14.70
Actual Defect Rate %: adr% = 14.70

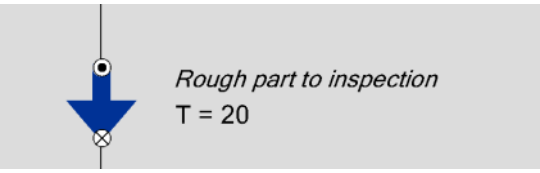
ti => 0 | Start
twa => 0 | Start
tp => 0 | Start
tt => 0 | Start
tm => 0 | Start
tr => 0 | Start
pi => 0 | Start
po => 0 | Start
ps => 0 | Start
pdd => 0 | Start
pad => 0 | Start
pdnd => 0 | Start

ftwa => twa/ss | Finish
ftp => tp/ss | Finish
ftt => tt/ss | Finish
ftr => tr/ss | Finish
ftm => tm/ss | Finish
fti => ti/ss | Finish

pt% => 100* tp /(tt + tm+ tp+twa +ti+ tr) | Finish
fty% => 100*po/pi = DIV/0 | Finish
ddr% => 100*pdd/pi = DIV/0 | Finish
adr% => 100*pad/pi = DIV/0 | Finish

We will now use assignments to build up the result variables as the simulation progresses

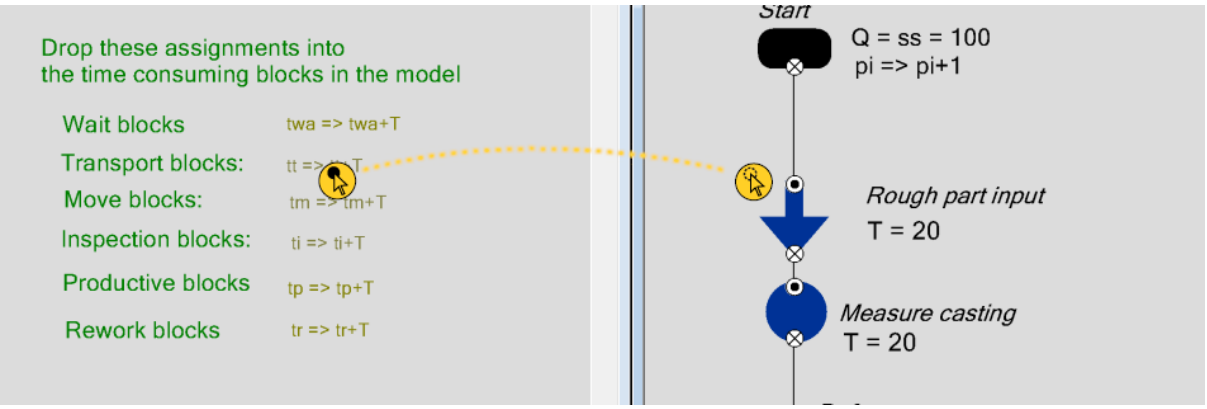
This is the first block in the model :



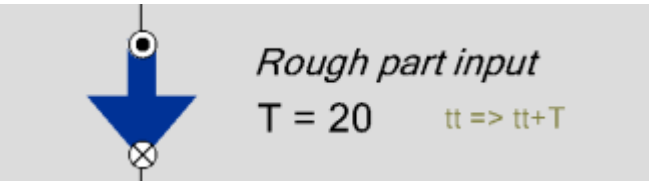
This looks like a transport block

Scroll to the bottom of the library - It contains various assignments which can be used to increment the right result variables.

Drag the transport assignment into the block:

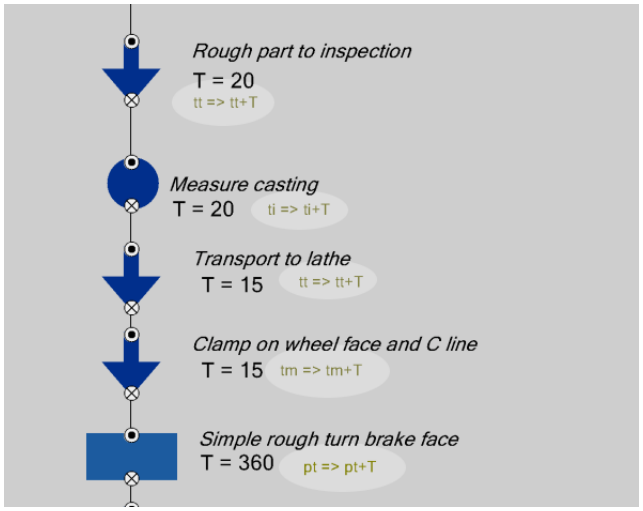


Result :

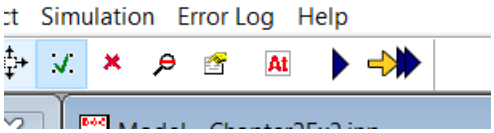


When the token goes through this block, the value of tt (transport time) will be incremented by the value of T in this block.

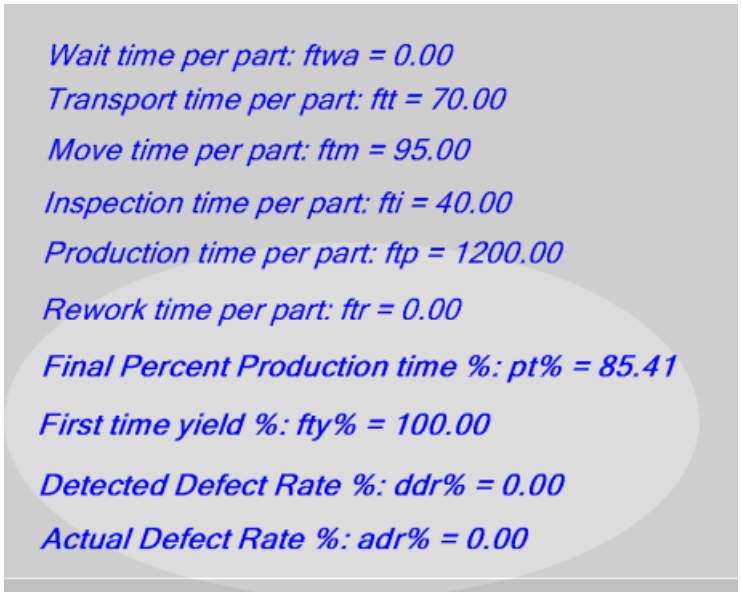
Likewise fill all the blocks with their relevant assignments :



Now solve the model – either step through or use the single quick solve button here :

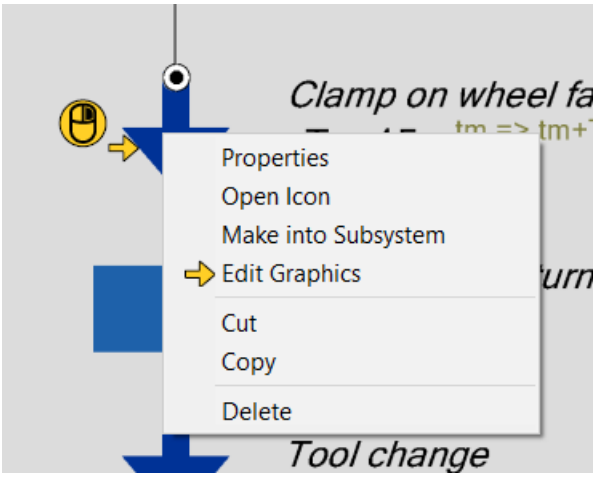


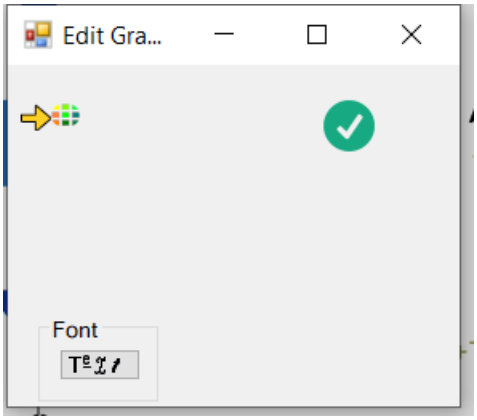
Aliquis will cumulate all the times and calculate the key results here



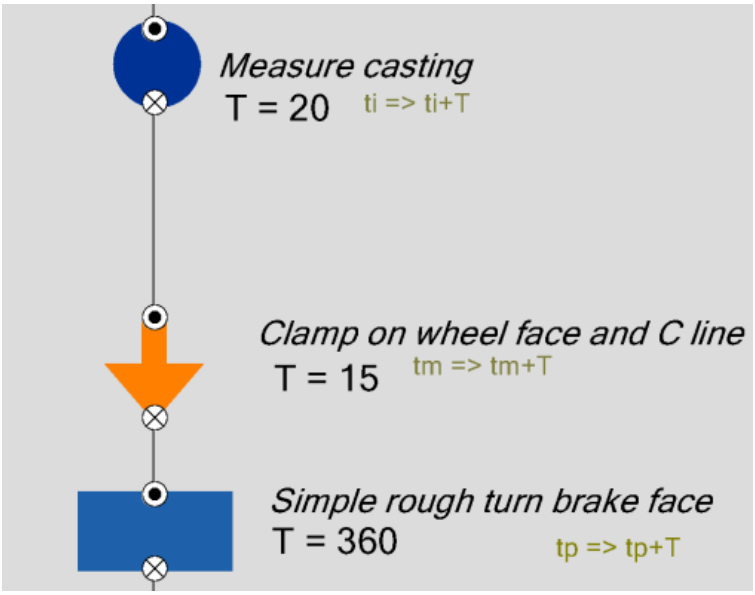
We have not done any work on scrap and good production quantities and , in this model there is no waiting – but we are already starting to see the extent of the waste time from these results.

Before we think about scrap there is one hint that you might want to take up. If you want to highlight a block for discussion, you can change its colour :





You might end up with something like this :

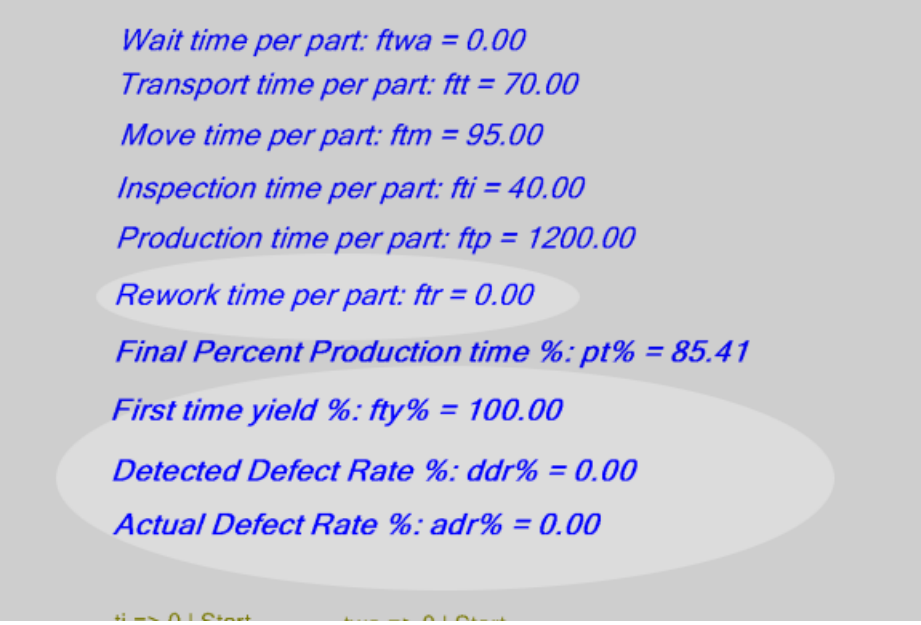


You decide how colours can be used in your model – but they are a useful communication tool.

Modelling Scrap

The biggest waste in a process like this may well be scrapped parts. In this next exercise we will model the scrap which is discovered in the inspection processes.

Before we start modelling , take a look at the missing or meaningless results:



We are going to work on the rework time in a similar way as we have already. Any activity which is the result of a defect being repaired will be deemed rework time.

First time yield is a well known manufacturing metric . It measures the percentage of parts entering the process that ends up as good output. The assumption is that the scrap rate is :

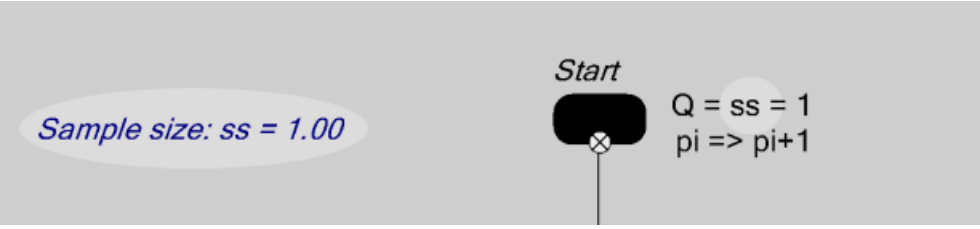
$$scrap\ rate\ (\%) = 100 - First\ time\ yield$$

The actual defect rate is the percentage of defective parts that the process creates. However, not all of these will be detected. The detected defect rate measures this.

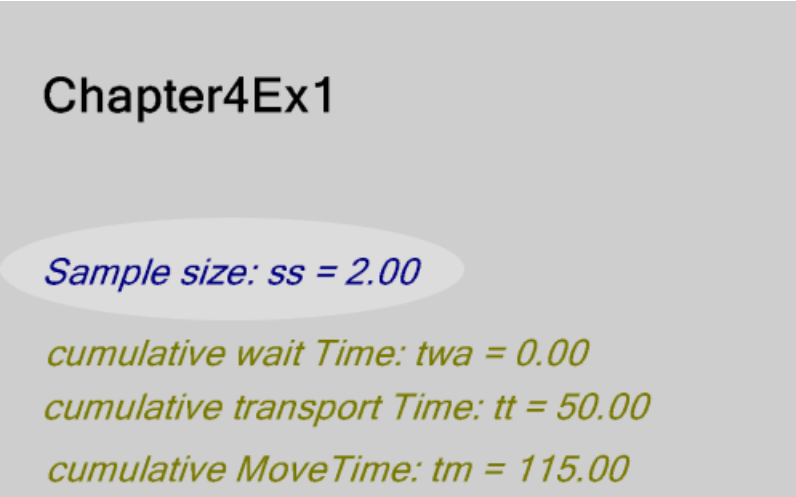
To work these numbers out there is now a need to do a large number of trials – up to now we have just done one.

This will be achieved by setting up a parameter – sample size:

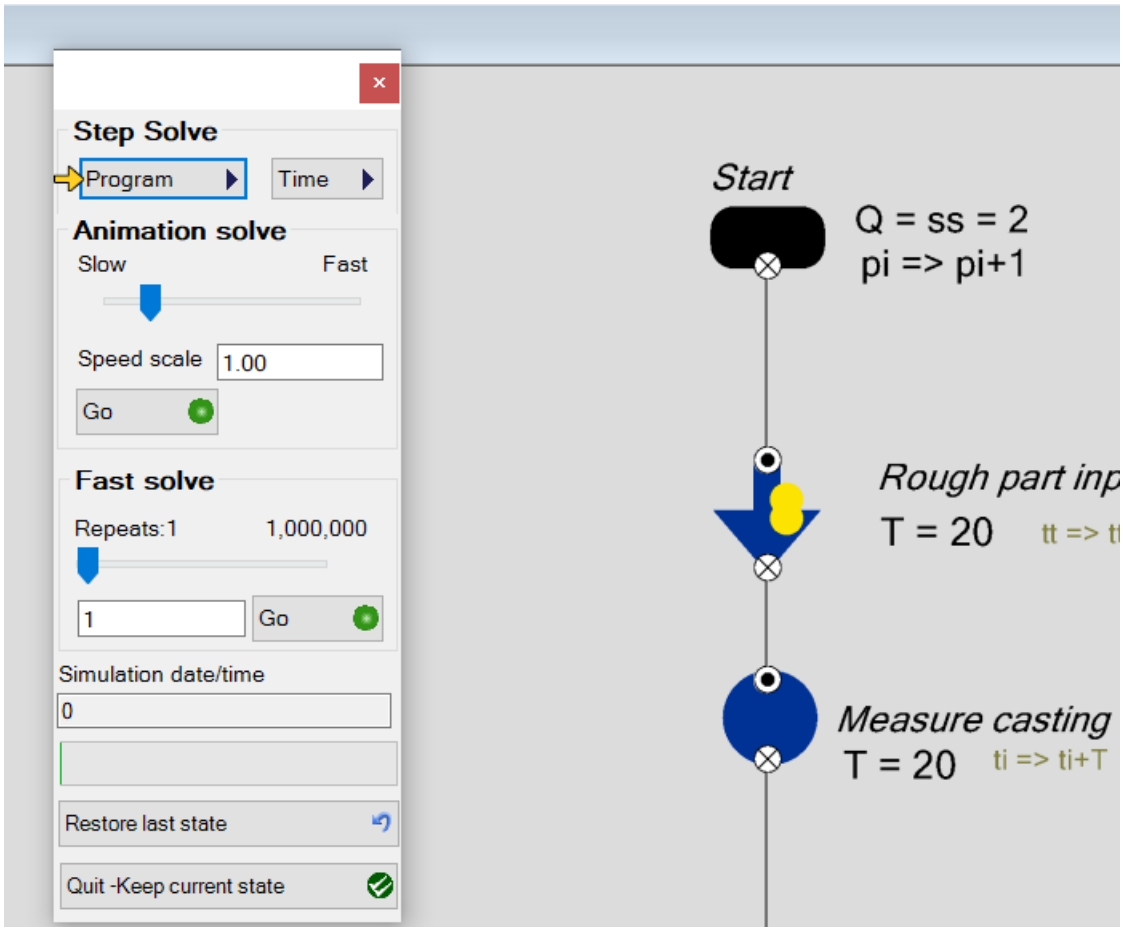
This will be linked to the source driving the process.



To make sure you understand the framework so far, set the sample size to 2:



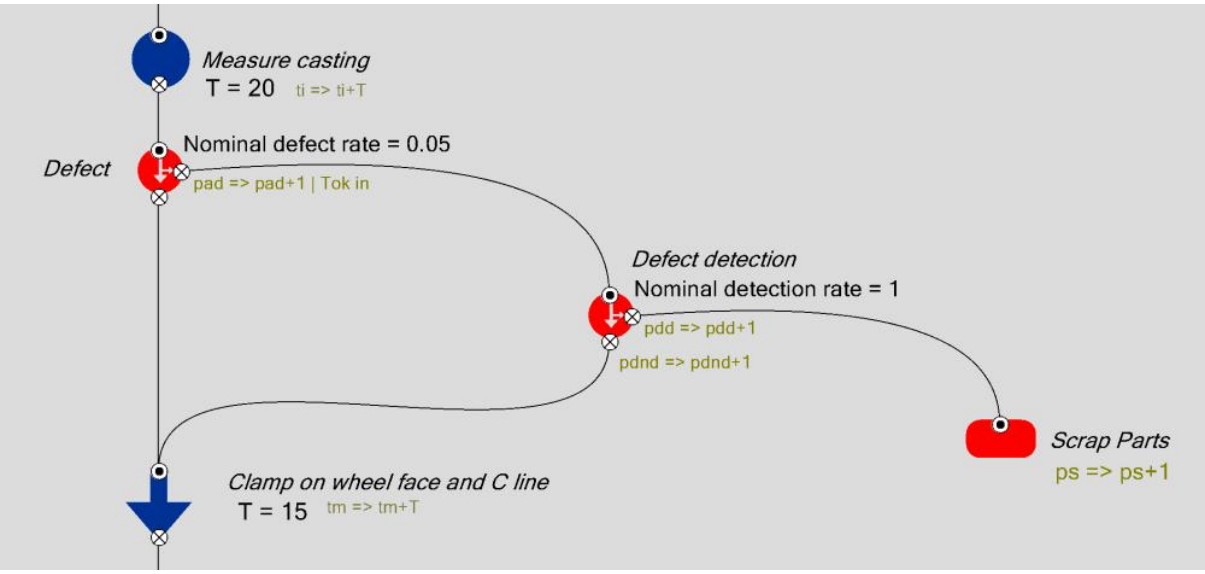
And step solve to understand that 2 tokens will now be going through the entire process.



Thee model is currently *deterministic* rather than *stochastic*. A sample of 1 will give us all the information we need. However, for our next task, we must consider that a proportion of parts may be defective at different stages of the process, we need to have a large sample to work the probabilities through.

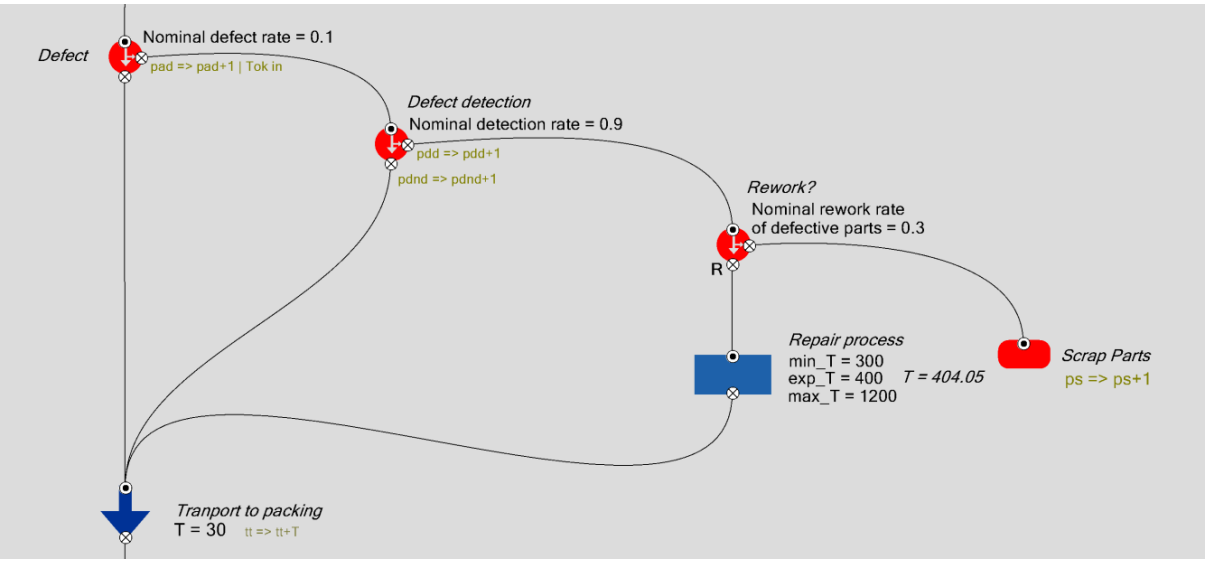
In our model, there are two inspection processes. Let's assume that the initial inspection is fairly straight forward. There is a well known defect rate (say 5%) for incoming rough parts and there are no reworking capabilities. Let us also assume that the detection rate is 100 %. Defective parts will all be logged and scrapped. The time involved will be booked to rework time as it is time used because a defect has occurred.

Using the blocks in the waste analysis sub library, build the model as follows :

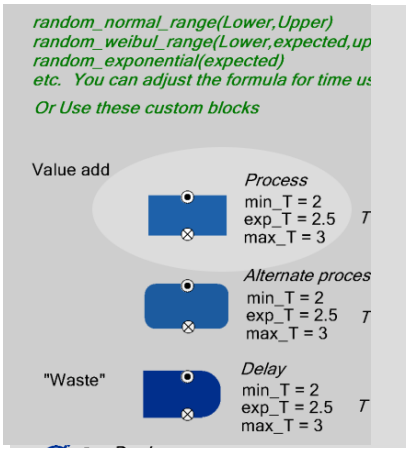


Let's assume that the second inspection process has a defect rate of 10 % but only 90 % of these are detected. Of the defects which are identified,, 30 % can be repaired. Let us also assume that the repair processes take 400 seconds on average, with an upper limit of 1200 seconds and a lower limit of 300 seconds.

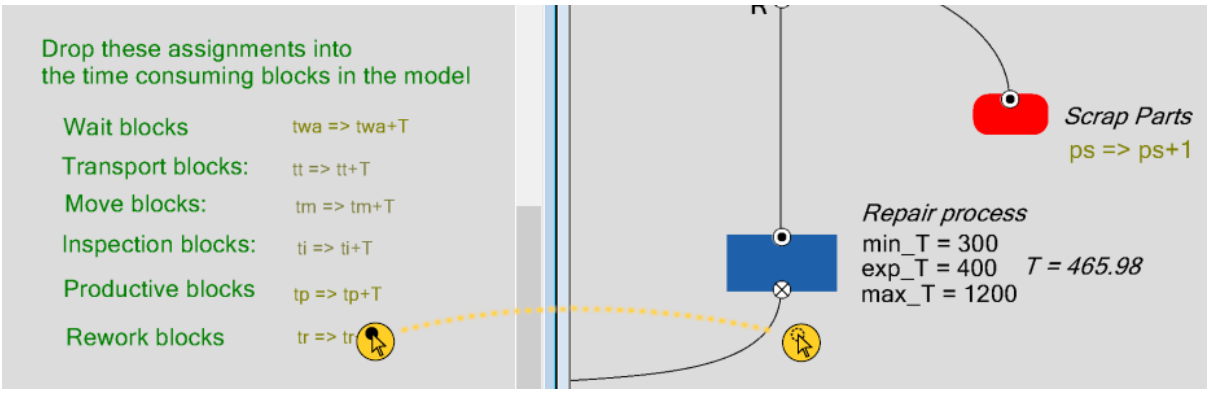
To achieve this – build the following model:



Reminder : You can get the repair processes block from the “Variable process time” sub library.

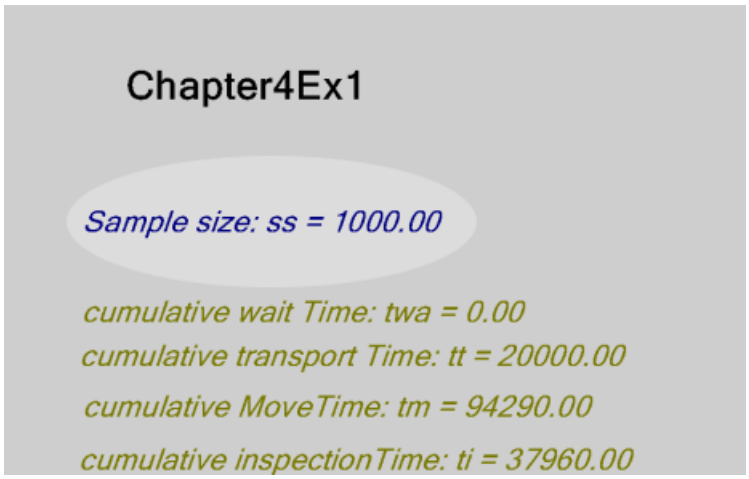


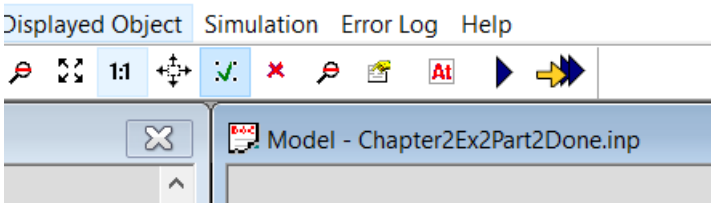
There is one last thing to do – we need to assign the time in the repair process to our rework time variable:



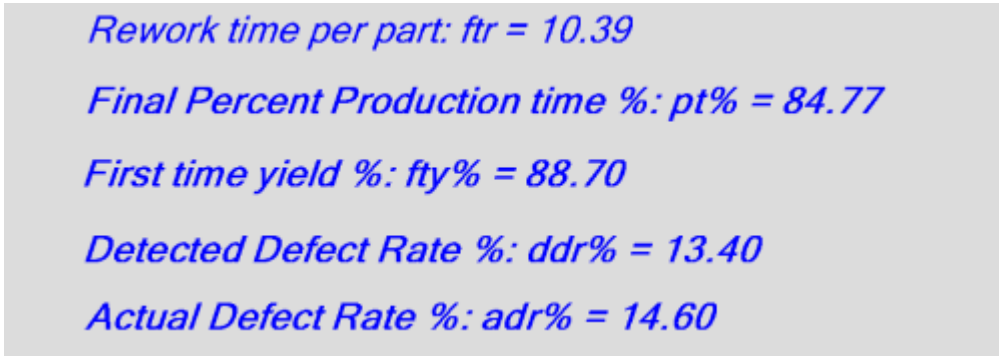
Running the model

You might want to step through with a small sample size (eg 20) to check the model works as you expect – alternatively you can go for a sample size of 1000 and do a fast solve:





This will give you the following statistics (or similar as it is now a stochastic system):



Review

So in this chapter we have focussed on time and waste analysis. In the process we have seen that you can build a mathematical framework around a simulation. By tailoring to a particular application (in this case process waste analysis) you can create a standardised system which is easy to use and will result in users understanding how the time is used and how the waste is created. From there, users can work on improved process design and , of course cost savings. In the next chapter we will notionally cost the waste and the defects getting through the system to understand the kind of savings that can be achieved.

Key Engineering concepts :

- Waste in process design

Key Aliquis concepts:

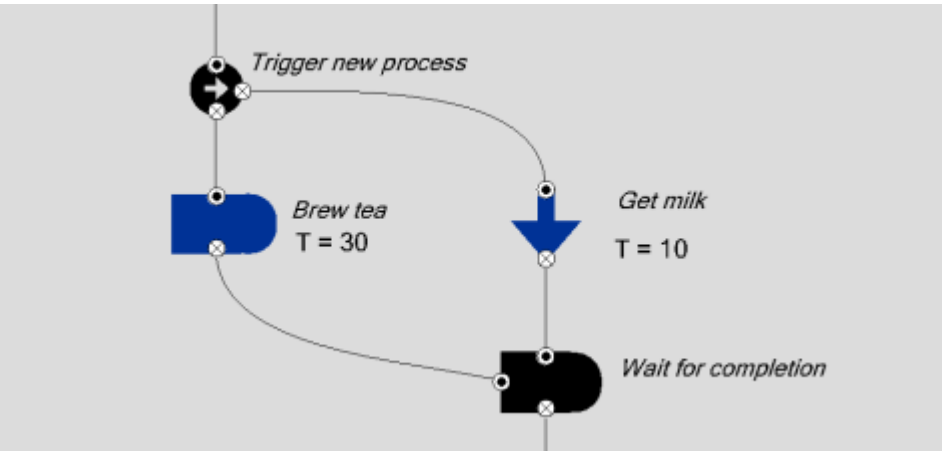
- User variables
- Assignments
- Using tailored libraries

In the next chapter we will look at how this kind of analysis can be used and extended to support decision making.

Questions :

1. Name 8 wastes of lean manufacturing? Which of them cannot be reduced by careful process design.
2. How does – MURA – variation in production processes affect these MUDA wastes?
3. Think about the relationship between these wastes. For example, if you have *defects* and defects are random, you will inevitably have *over production*.

4. In this process snippet, tea is brewed for 30 seconds, but while it is brewing we can get milk. We can see that there is an implied waiting time of 20 seconds. How would you formulate an arrangement in Aliquis which works it out?



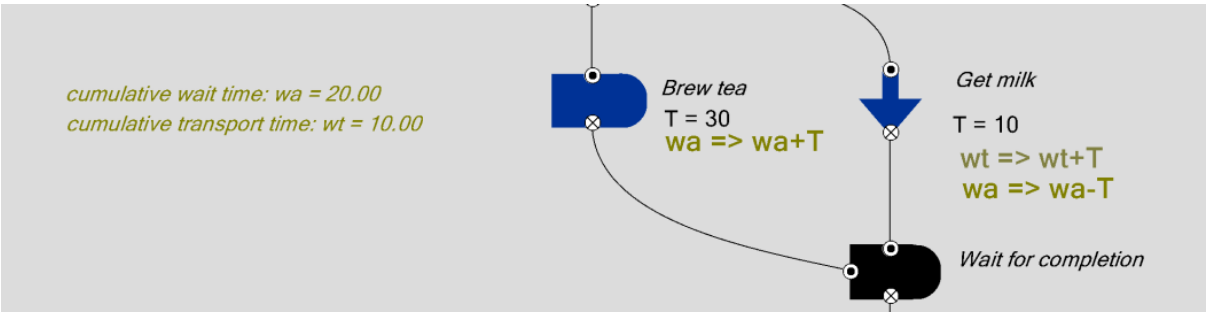
5. How do predict a defect rate for a model if the process has not yet run ? What are your general conclusions based on this thinking.

Answers

1. The 8 wastes are :

D	Defects	Effort caused by rework, scrap and incorrect information
O	Over production	Production of more than is needed or production or before it is needed.
W	Waiting	Wasted time waiting for the next step in a process
N	Non utilised talent	Not making full use of peoples talent, skills and knowledge
T	Transportation	Movement of goods between locations both within and external to a production facility.
I	Inventory	Excess products – Anything being stored anywhere
M	Motion	Unnecessary movement by people, machines, robots
E	Extra processing	Adding processing which results in a higher quality level than that desired by the customer

2. MURA or Variation will particularly affect *Inventory*. To avoid "starving" down stream elements, you will need to carry some buffer stock between stations. This is inventory. Conversely, if there is no buffer stock there will be *waiting*.
3. Defects will cause overproduction and either inventory or waiting.
4. In simple cases you can do something like this :



In the *Brew tea* block the waiting time *wa* is incremented by the time in the block, *T*. However, in the *get milk* block , the waiting time is decremented by the time in the block. Transport time *wt* is incremented. You can see from the results on the left that the correct numbers are calculated. Moreover you can adjust the process times and you will still get the correct result as long as the wait is longer than the activities to be done during the wait.

Later in the book we will look more carefully at the mathematical options to take if there is uncertainty as to whether this is true.

5. Actually it is very difficult to predict average scrap rates accurately. Engineers could look at historic data for similar processes, statistical analysis of machine capabilities

(6 sigma) , experience etc. But in the end, the input defect rates will be a best estimate and some inaccuracy would be expected. This means that the model results will also be estimates. The accuracy of any model depends on its structure and the accuracy of the data that is feeding it. In the early stages of a new venture or plan this accuracy will be significantly higher than a guess but lower than most people expect from computers! However – higher than a pure guess accuracy is still of use. Moreover, the main thing is that you can do comparisons – What is the effect of *this* improvement verses the effect of t*hat* improvement?