Chapter 5 – Pseudo costing a process design

Understanding process costs.

The costs that a company incurs in producing parts for customers depends on many factors. Not least:

The design of the part. It is the designers job to determine the material, shape and dimensional tolerances of a part. The material and shape will determine the material costs. The shape, Material and tolerances will determine the basic process – which in turn determines a large portion of the process costs.

The process design: It is the process engineers job to deliver the design defect free to the customer. Although the basic process required may largely be determined by the design. Modes of transport, automation, machine selection and parameters such as speeds and feeds are all in the hands of the process engineer. All this have impact on costs. The process choices also have a big impact on the potential for delivering close to defect free quality. Defects getting to the customer generally have massive *hidden* costs.

Production volumes : Large production volumes tend to deliver lower piece costs – for both materials and processing. In addition, fluctuating demand tends to be more costly to service than fixed rate demand.

Operations : Locations, supply chain management, distribution networks, factory design and manning levels all come under operations. Clearly they also have an effect on costs .

From these considerations, we can see that it is not really possible to get an accurate costing for a part from the process alone. However, we can adopt a scheme for *notionally* costing a process, including the hidden costs of poor quality. We can successfully compare process designs and optimise them based on this .

A note on Quality :

Modern thinking about quality would include the following questions being posed and answered:

How does the customer define quality? If our design and quality system concentrates on things which are not actually important to our customers, we will end up with over-processing one of the seven wastes identified by Toyota.

How can use our quality system to ensure that the process produces zero defects?

Modern thinking is all about process capability and ensuring that the process is controlled to the point that inspection is only necessary to ensure that a fully capable process remains in control.

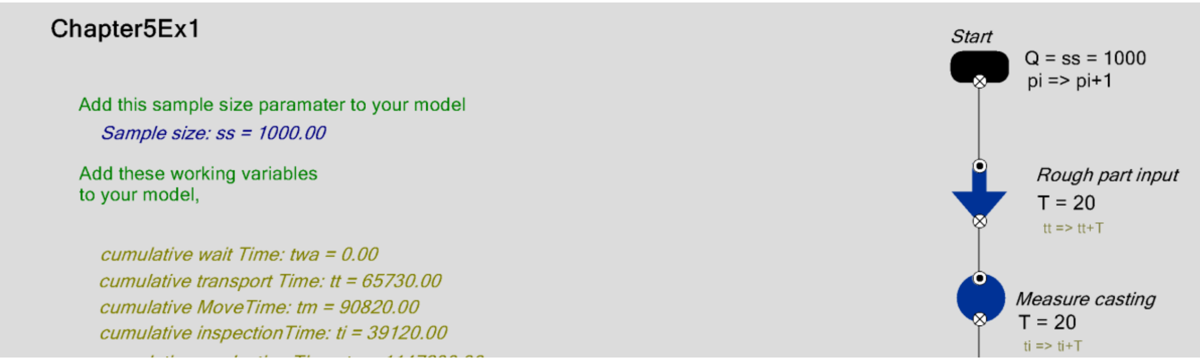
In this chapter we will take a very simplistic view of quality systems. We will make a basic and old fashioned assumption that the purpose of a quality system is to find defective parts and stop them getting to the customer… Sometimes, when there is a relatively high level of defects and complex systems – this approach is still necessary.

I would call this potato farmer quality control. It is inevitable that some of the crop will be misshapen, eaten by pests etc. 100 percent inspection for the supermarkets is something that cannot really be avoided.

So in this chapter we are going to notionally cost a process assuming 100 percent inspection and looking at the time spent on that inspection vs the hidden costs of allowing defective goods get to the customer.

Modelling

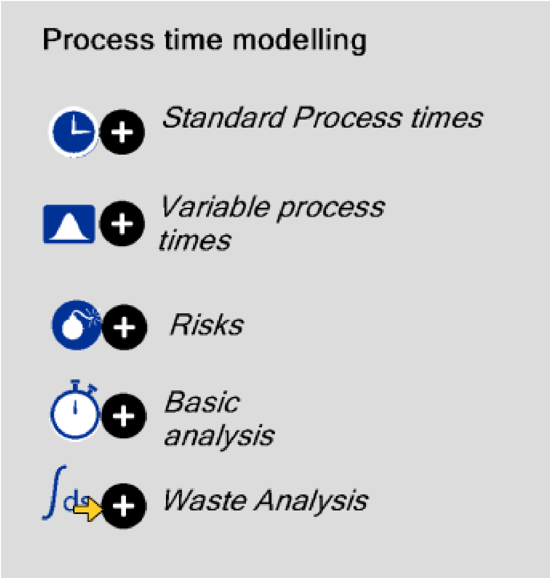
Open up the model Chapter5Ex1 from your model store:



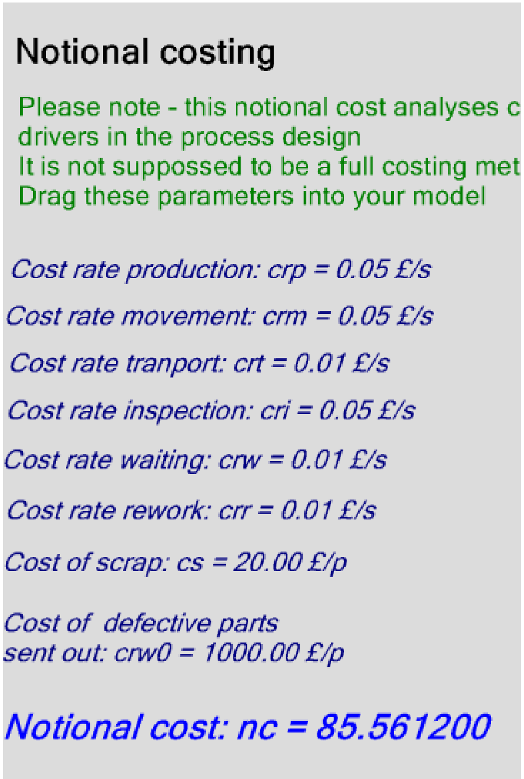
This is the result of the last lesson . Our mission now is to load it up with process costs.

Lets sort out some of the variables :

Go into the waste analysis library :



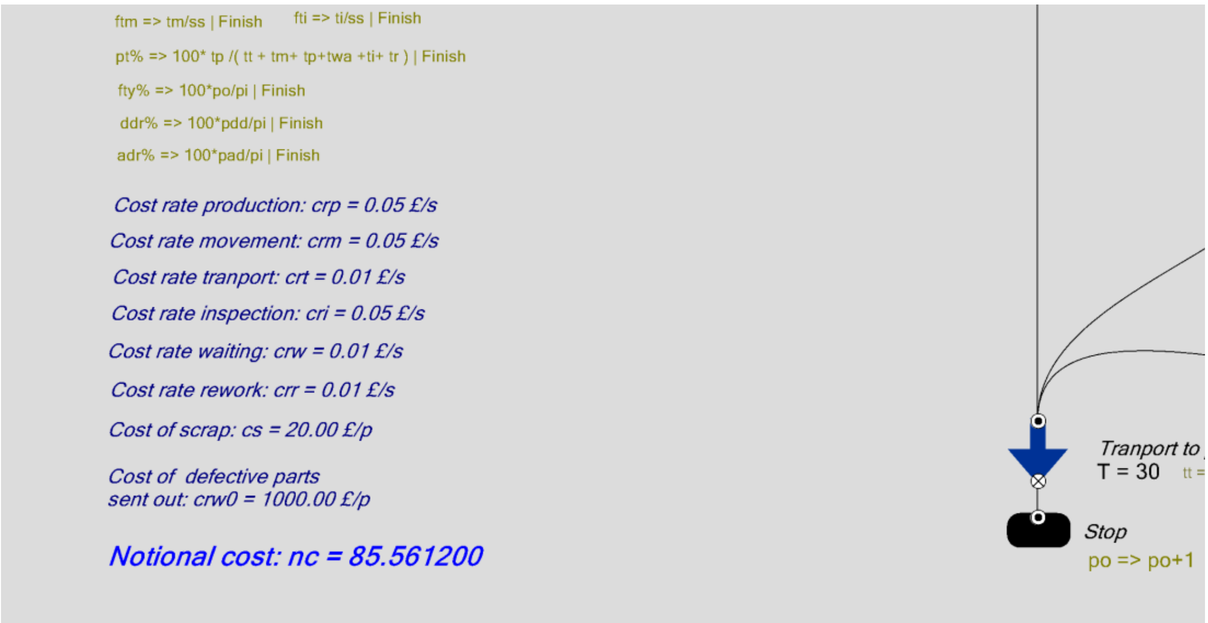
And scroll down to here :



Our approach to costing this process is to build on the analysis we have already done. We will make the cost rates for the different kinds of activities parameters in our system.

There is also a variable *nc* (notional cost) to carry the final cost result.

Drag these variables into your model :



You can set these cost rates to whatever you like. But notice that there is one cost which is currently very high. The cost of a defective part sent to a customer. This is a hidden cost but it reflects the reputational damage that poor quality can do.

We will now import the maths used to compute the final notional cost:

Looking at the formula closely you will see that the final notional cost is computed as

The sum of :

Waiting time * waiting time rate

Transport time * transport time rate

Move time * move time rate

Inspection time * inspection time rate

Production time * production time rate

The number of scrapped item * the cost of a scrapped item

The number of defects let through * the cost of a defect left through

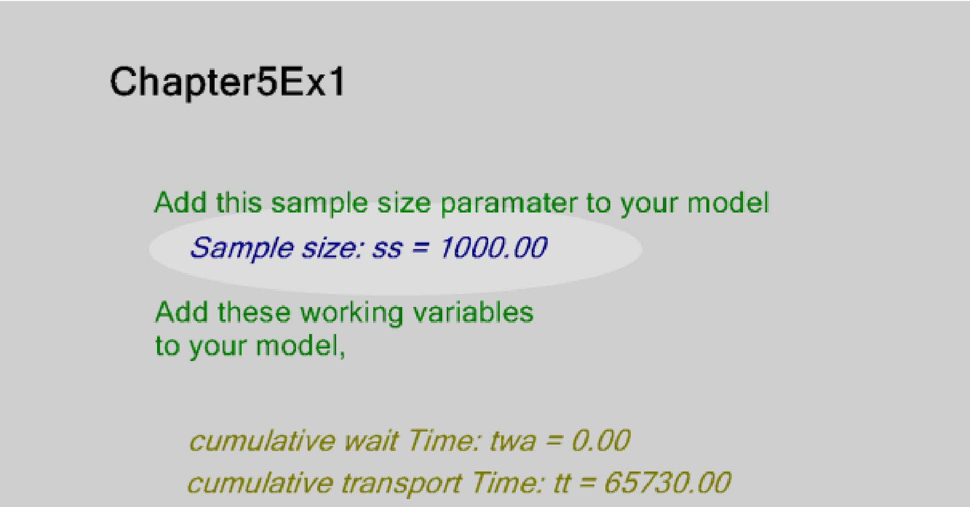
All divided by

The number of samples

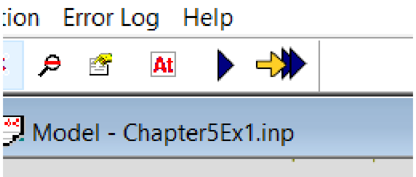
There are legitimate questions around how this could be improved. It is a little course grained. For example one production process might be done on a very costly machine where another is done on a table!   However, in this example it does give a reasonable notional cost.

Lets's go with it for now but the question and answer section of this chapter you will be given another approach which is a finer grained concept for costing.

Note that the sample size is set to 1000:



Lets do a quick solve :



Result

Notional cost: nc = 82.769000

Or similar.

Actually if you repeat the solve a few times there is quite a lot of variation in the computed notional cost.

Notional cost: nc = 73.829000

Notional cost: nc = 77.730000

If you read any statistics text book, you will understand that if you take a sample from a process, then take the average you will get what is called the sample mean. If you then take another sample and take the average of that – you will not get the same sample mean. Normally this variation is quite small – but in this case it is bigger than most people would expect because of the disproportionate cost of a defect part being allowed through the system couple with the low probability of that happening.

To reduce this variation, we have to increase the sample size - Try setting it to 10000

Repeat solve a few times and you will so that the variation has reduced – but also that it takes longer to do each solve.

As we will later be doing multiple solves and at the moment we are in educational mode, it is suggested that we keep the sample size to 1000 for the time being.

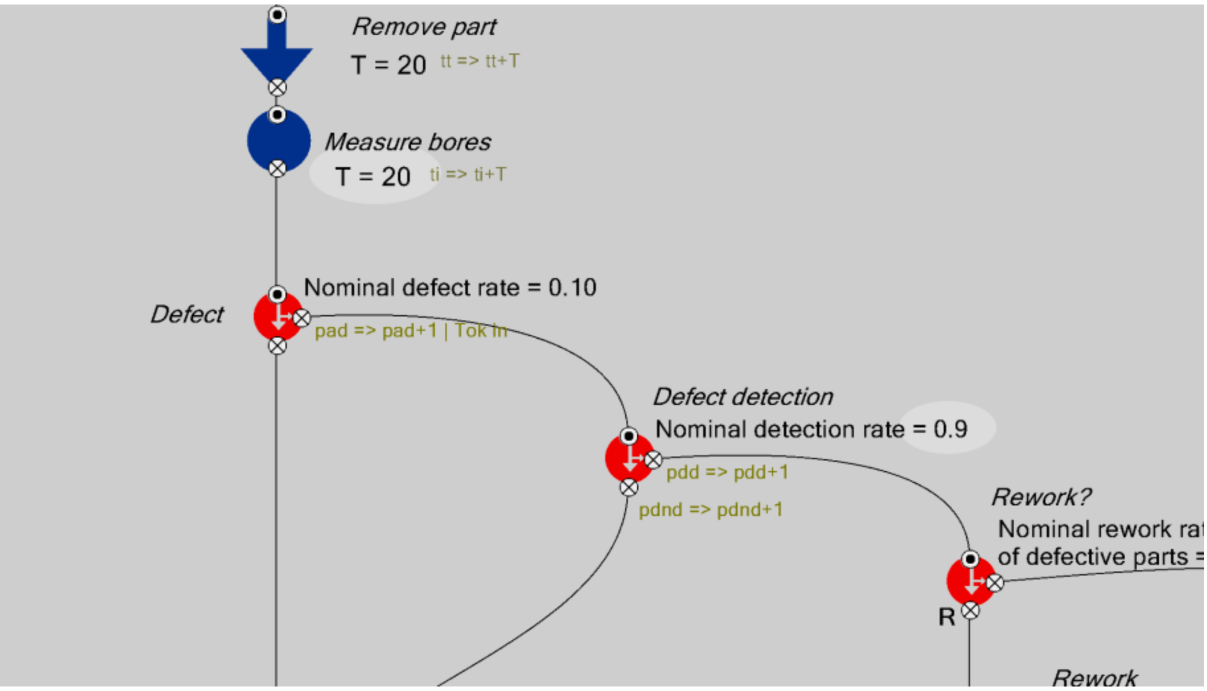
The main point is this : Like many things in life there is a trade of here between accuracy and cost (in time). We are going to examine another of these trade offs in a little more detail now.

Lets imagine a dispute between the production and quality control department:

The quality manger says that the costs can be brought down by increasing the inspection time to ensure that fewer defective parts get through to the customer.

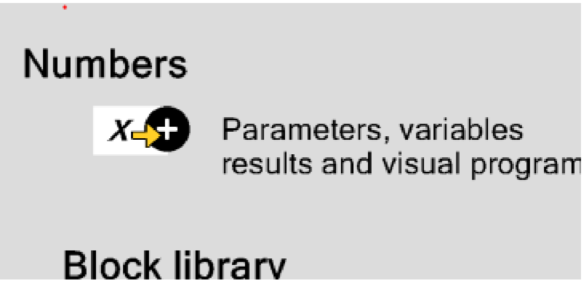
The production manager counters by saying that inspection time is waste! It should be reduced. At very least the idea of increasing it should be resisted.

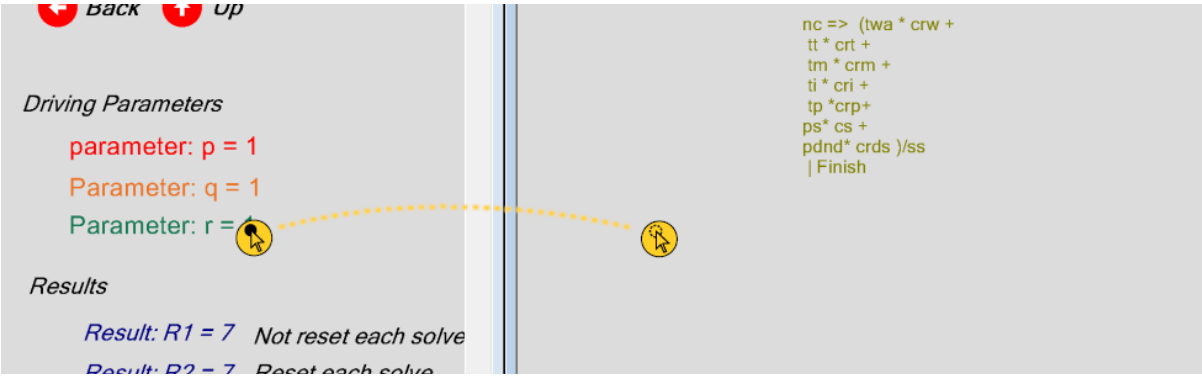
We are actually talking about these parameters in our model :



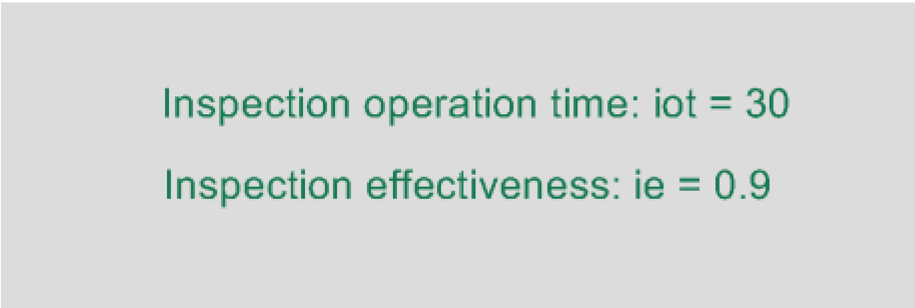
We will start by paramaterising the model and getting the current baseline figure a little more tied down :

Lets put in a system parameter for inspection time and detection rate:

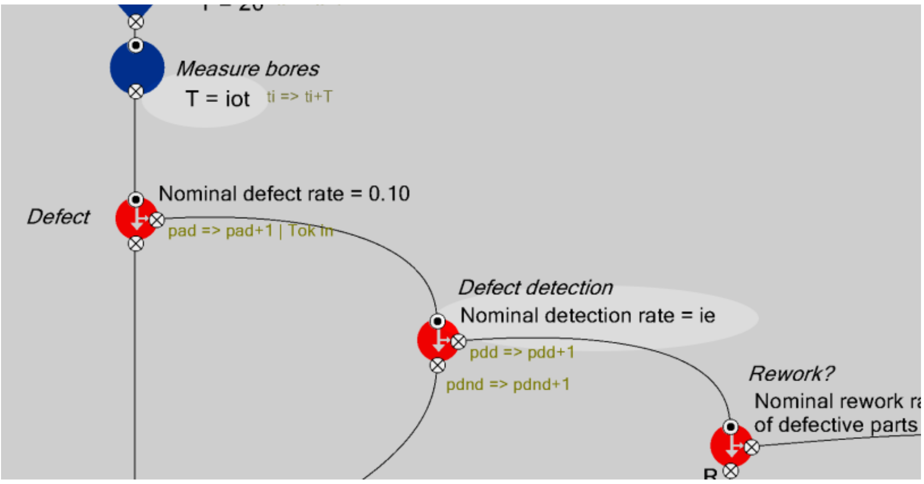




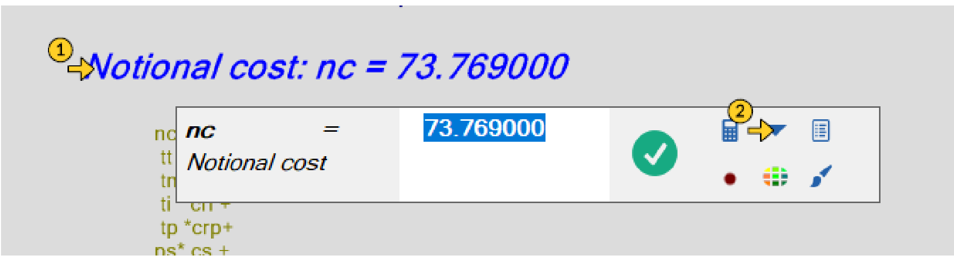
Copy two parameters in and edit them as follows:

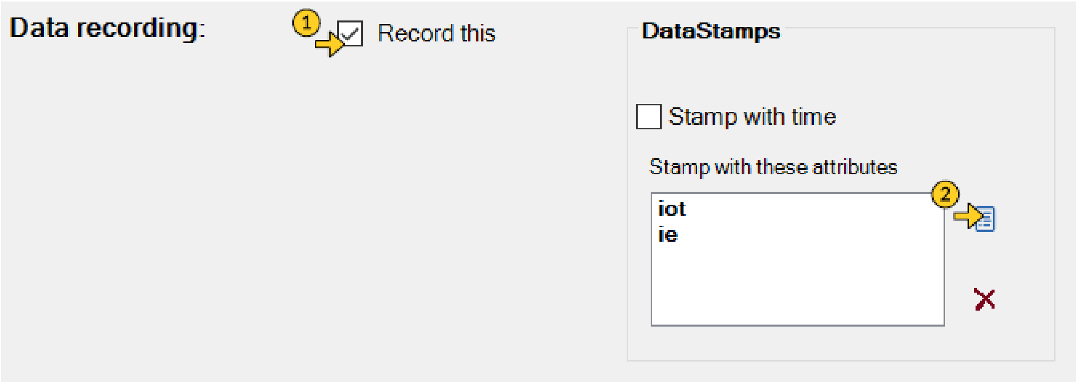


Now tie the block system attributes to these parameters :

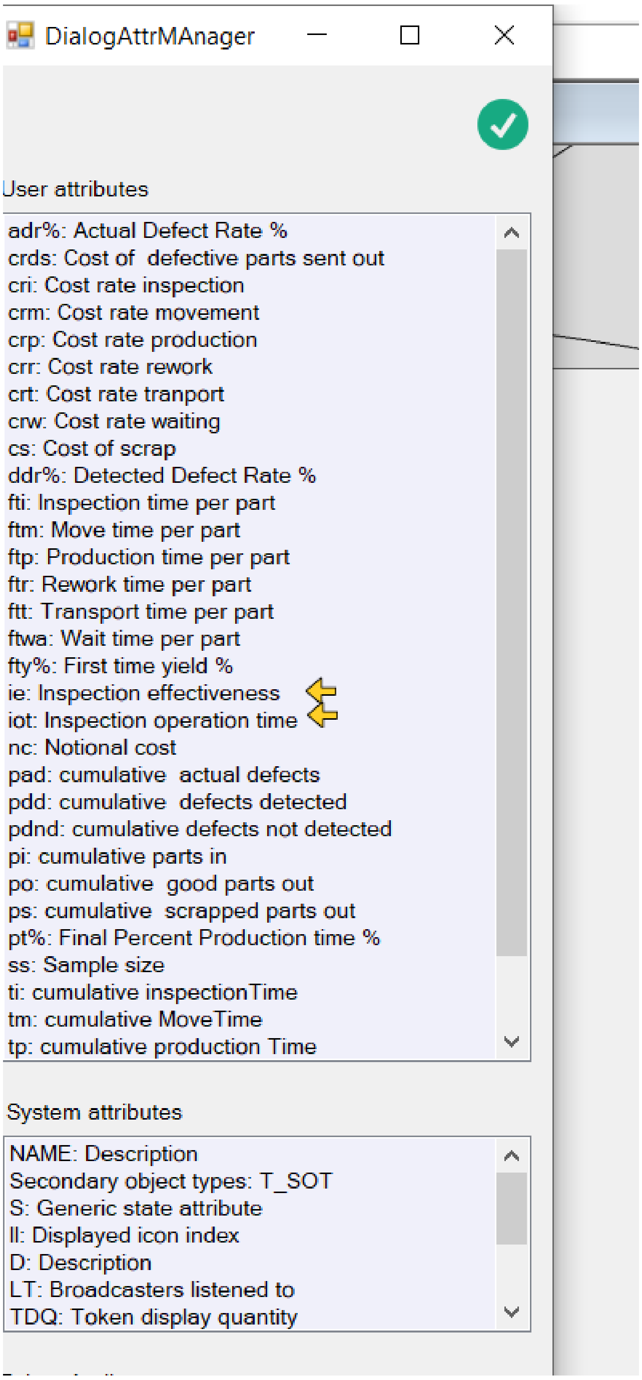


We will now save our result variable *with* the parameters:

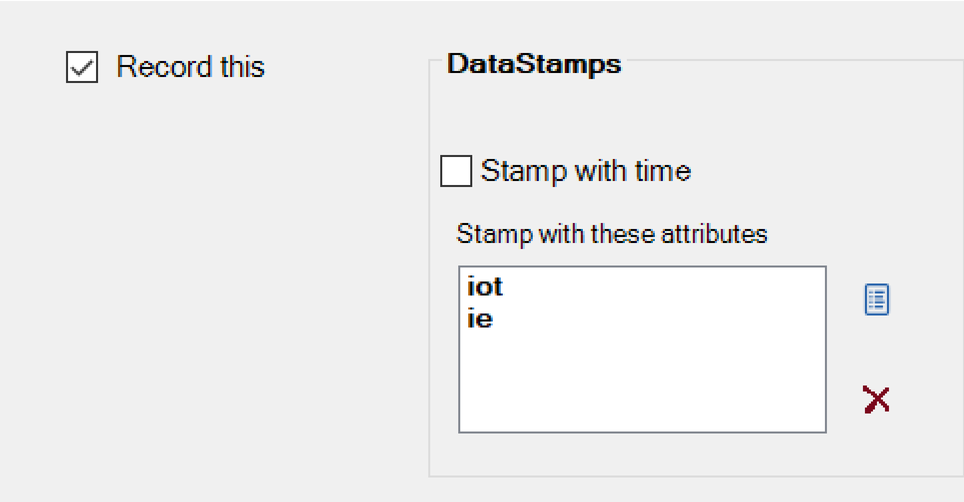




This will produce a large list of all the parameters, attributes and variables found in the top system level . Select our new parameters *iot* and *ie*

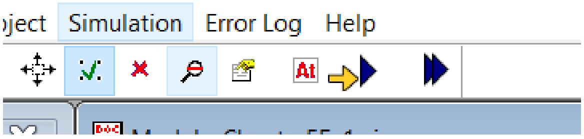

DialogAttrMAnager
User attributes
adr%: Actual Defect Rate %
crds: Cost of defective parts sent out
cri: Cost rate inspection
crm: Cost rate movement
crp: Cost rate production
crr: Cost rate rework
crt: Cost rate tranport
crw: Cost rate waiting
cs: Cost of scrap
ddr%: Detected Defect Rate %
fti: Inspection time per part
ftm: Move time per part
ftp: Production time per part
ftr: Rework time per part
ftt: Transport time per part
ftwa: Wait time per part
fty%: First time yield %
ie: Inspection effectiveness
iot: Inspection operation time
nc: Notional cost
pad: cumulative actual defects
pdd: cumulative defects detected
pdnd: cumulative defects not detected
pi: cumulative parts in
po: cumulative good parts out
ps: cumulative scrapped parts out
pt%: Final Percent Production time %
ss: Sample size
ti: cumulative inspectionTime
tm: cumulative MoveTime
tp: cumulative production Time
System attributes
NAME: Description
Secondary object types: T_SOT
S: Generic state attribute
II: Displayed icon index
D: Description
LT: Broadcasters listened to
TDQ: Token display quantity

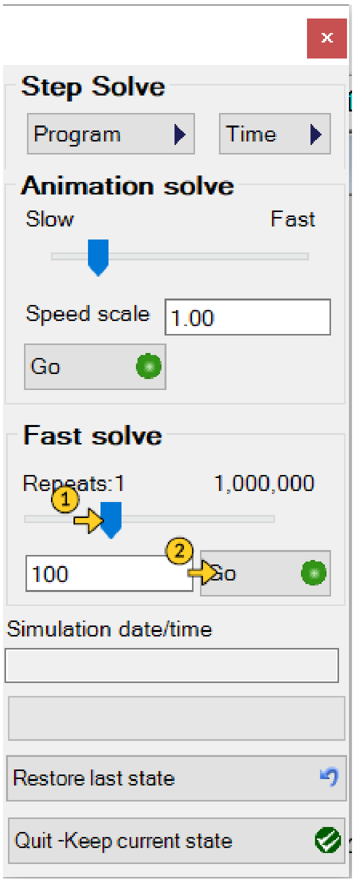
Result :



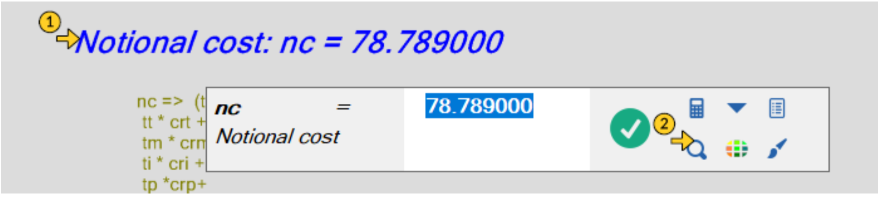
What this means is that every time *nc* (notional cost) is computed, its value will be saved along with the values of iot (inspection operation time) and ie (inspection effectiveness).

Lets give it a try with 100 solves :



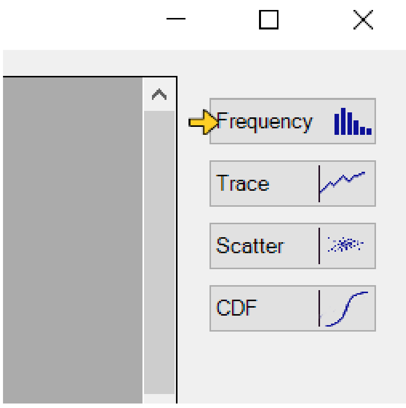


We can now examine the results :

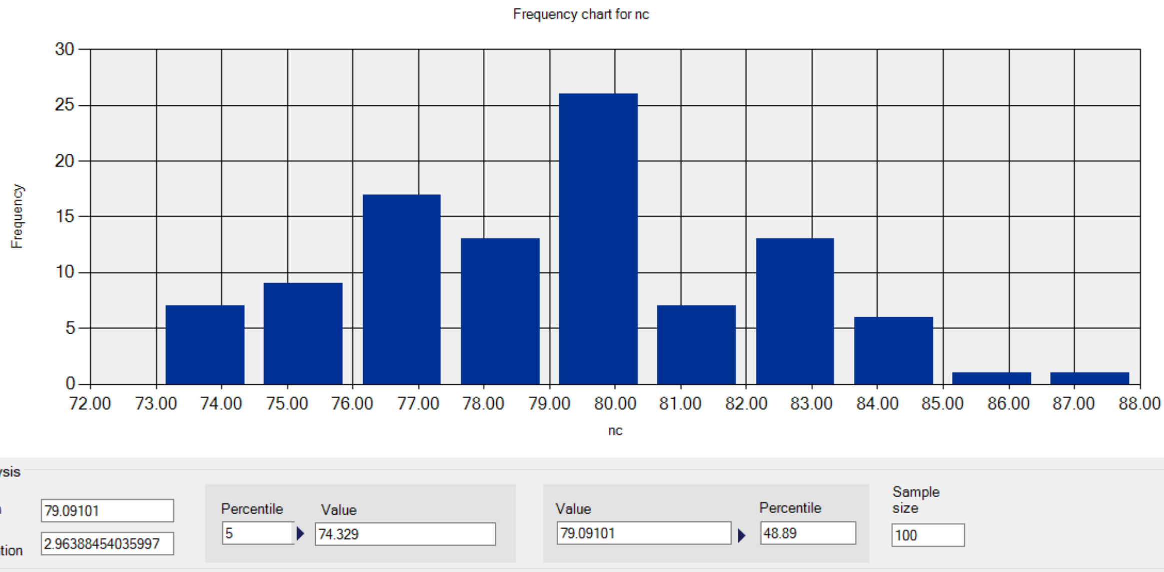


nc	iot	ie
82.013	30	0.9
81.072	30	0.9
83.151	30	0.9
74.289	30	0.9
78.427	30	0.9
76.994	30	0.9
81.21	30	0.9
78.23	30	0.9
78.033	30	0.9
77.348	30	0.9
82.132	30	0.9
84.269	30	0.9
84.269	30	0.9
77.171	30	0.9
84.974	30	0.9
75.407	30	0.9
75.935	30	0.9
76.033	30	0.9
79.21	30	0.9
80.309	30	0.9
81.151	30	0.9
82.348	30	0.9
78.289	30	0.9
73.856	30	0.9

To make sense of the variation - lets create a histogram



Result :



The base line mean is 79.09. (based on a total of a million trials)

Now we will tie the quality manager down…

What time does he want for the inspection procedure and what does he expect the effectiveness to be if he gets it?

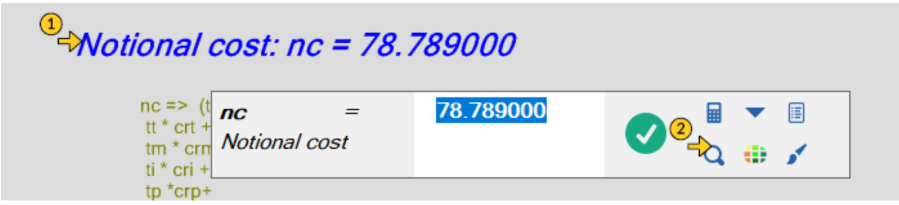
Lets assume he says he needs 100 seconds and he can get the effectiveness up to 95 percent.

Change the parameters as follows

Inspection operation time: iot = 100

Inspection effectiveness: ie = 0.95

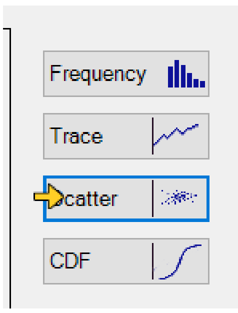
Solve another 100 times and bring up the result :

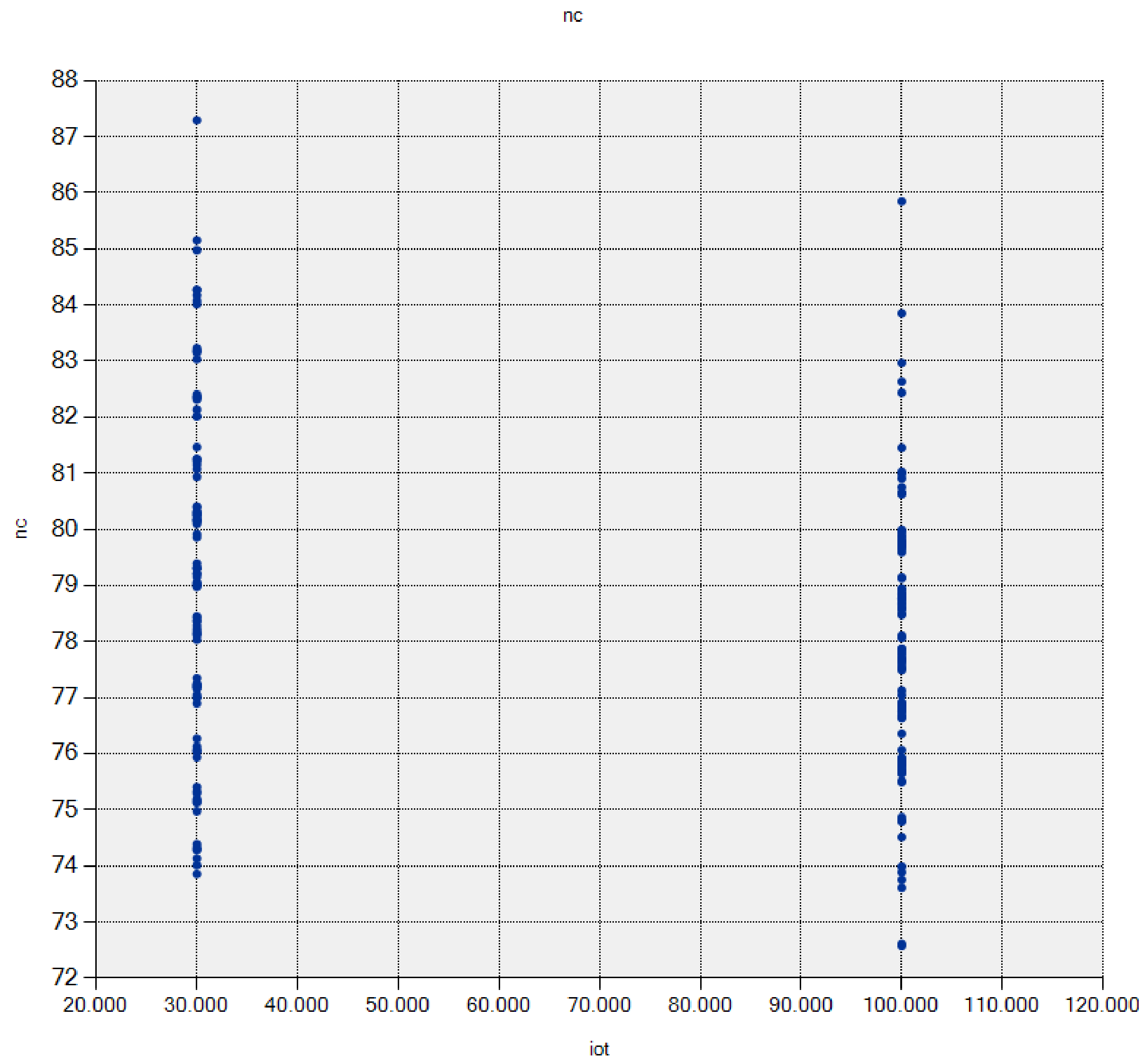


Heres the junction between the old results and the new

80.269	30	0.9
78.151	30	0.9
76.895	30	0.9
74.132	30	0.9
83.033	30	0.9
77.23	30	0.9
75.494	100	0.95
77.809	100	0.95
76.651	100	0.95
76.356	100	0.95
76.868	100	0.95

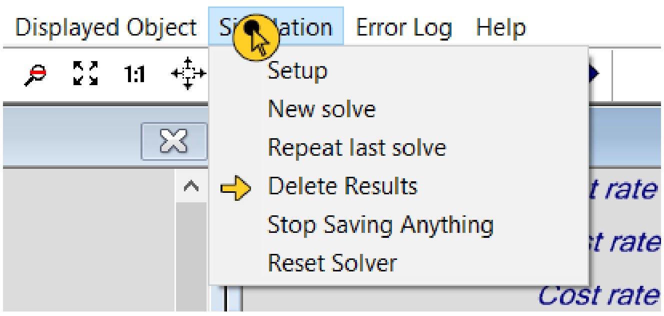
In this case a scatter graph is more useful than a histogram as there is considerable overlap in the results :



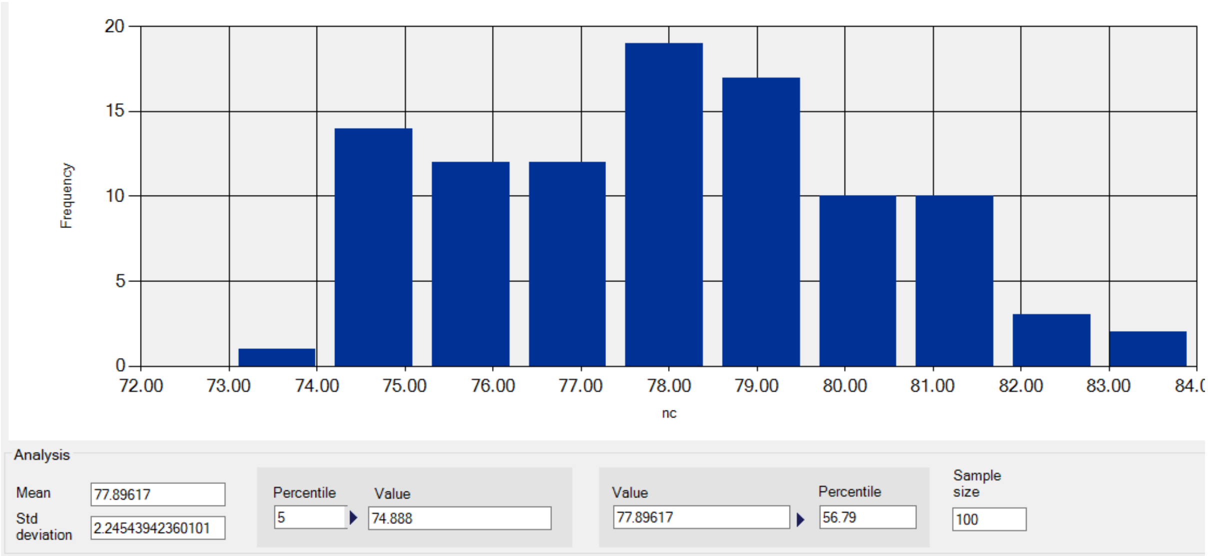


Just glancing at this we can see that the quality manager's proposal actually makes sense.

To get a quantative assessment we would have to delete the results and get a histogram just for the new parameters :



Solve 100 times gives this histogram :



The mean is notional cost is 77.90.

This means that the proposal would be expected to save 1.2 currency units.

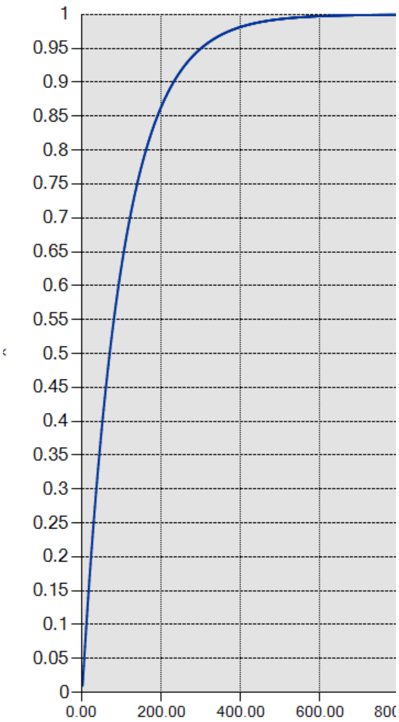
Lets take this one step further – We can postulate that the time spent on inspection affects both the inspection cost and the effectiveness of the inspection. Could there be an optimum point for overall process cost?

Well to do this we have to tie the two parameters together with some kind of formula. Getting the logic or mathematics for this formula is not necessarily straight forward but as usual a good guess is better than nothing.

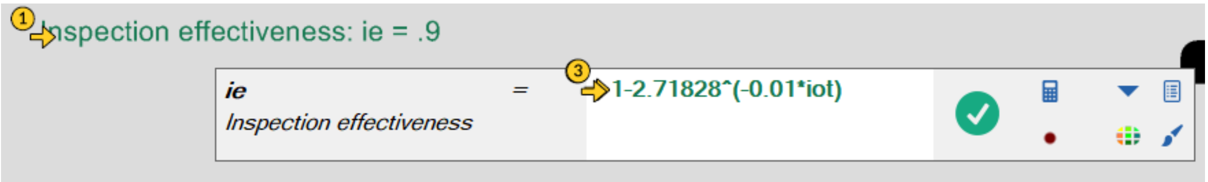
Lets assume the following :

$$inspection\ effectiveness = \ 1 - e^{-0.01*iot}$$

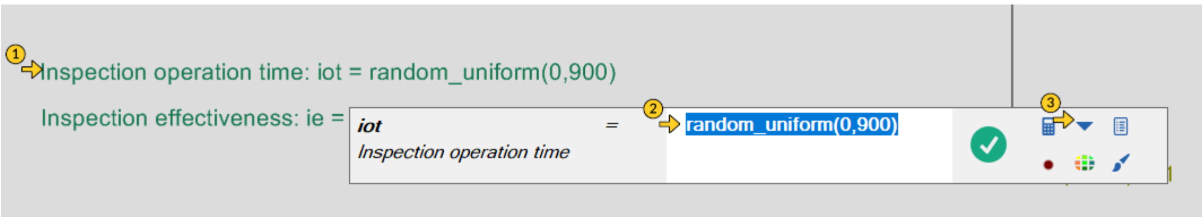
In graphical form this looks like :



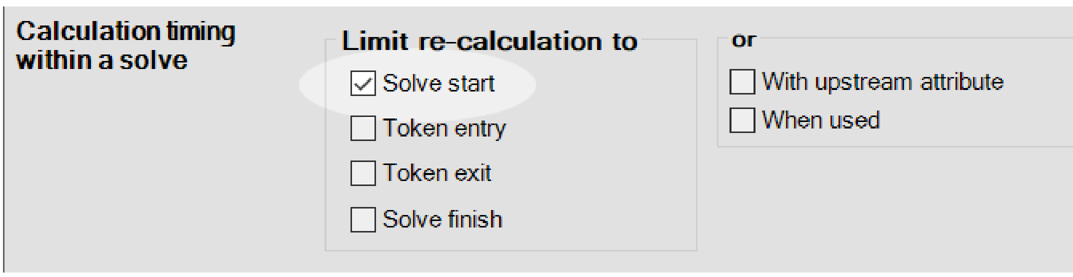
Set the parameter for inspection effectiveness to correspond to this formula



We can then set up the value of iot (inspection operation time) to be a uniformly distributed random variable between 0 and 900.



On expansion you will see this.



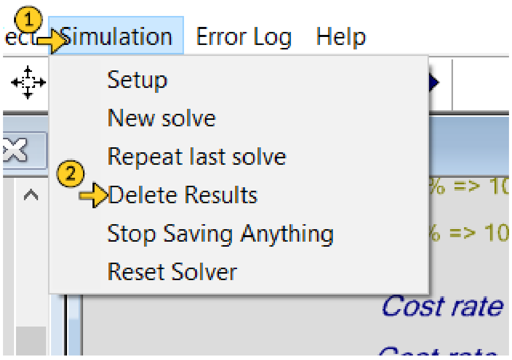
This means that the inspection operation time will be set to a uniformly distributed random variable between 0 and 900 every time a solve starts. It will take a different value for each solve.

Result :

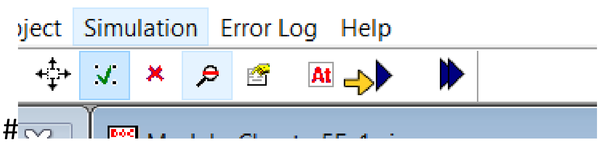
Inspection operation time: iot = random_uniform(0,900)

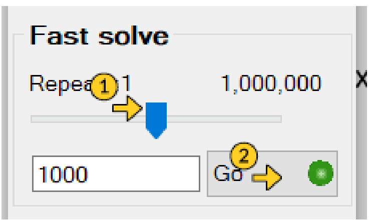
Inspection effectiveness: ie = 1-2.71828^(-0.01*iot)

Delete the results



and solve 1000 times .

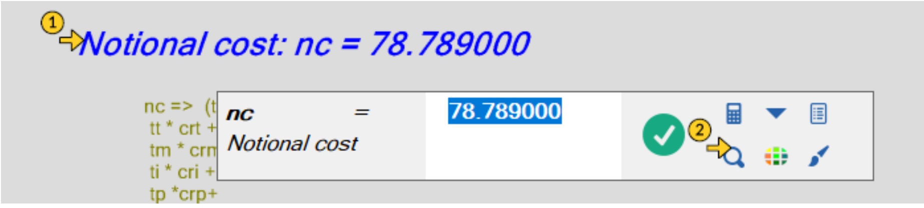


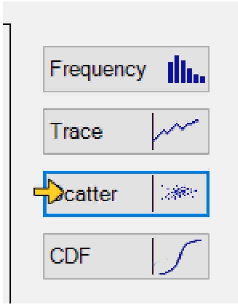


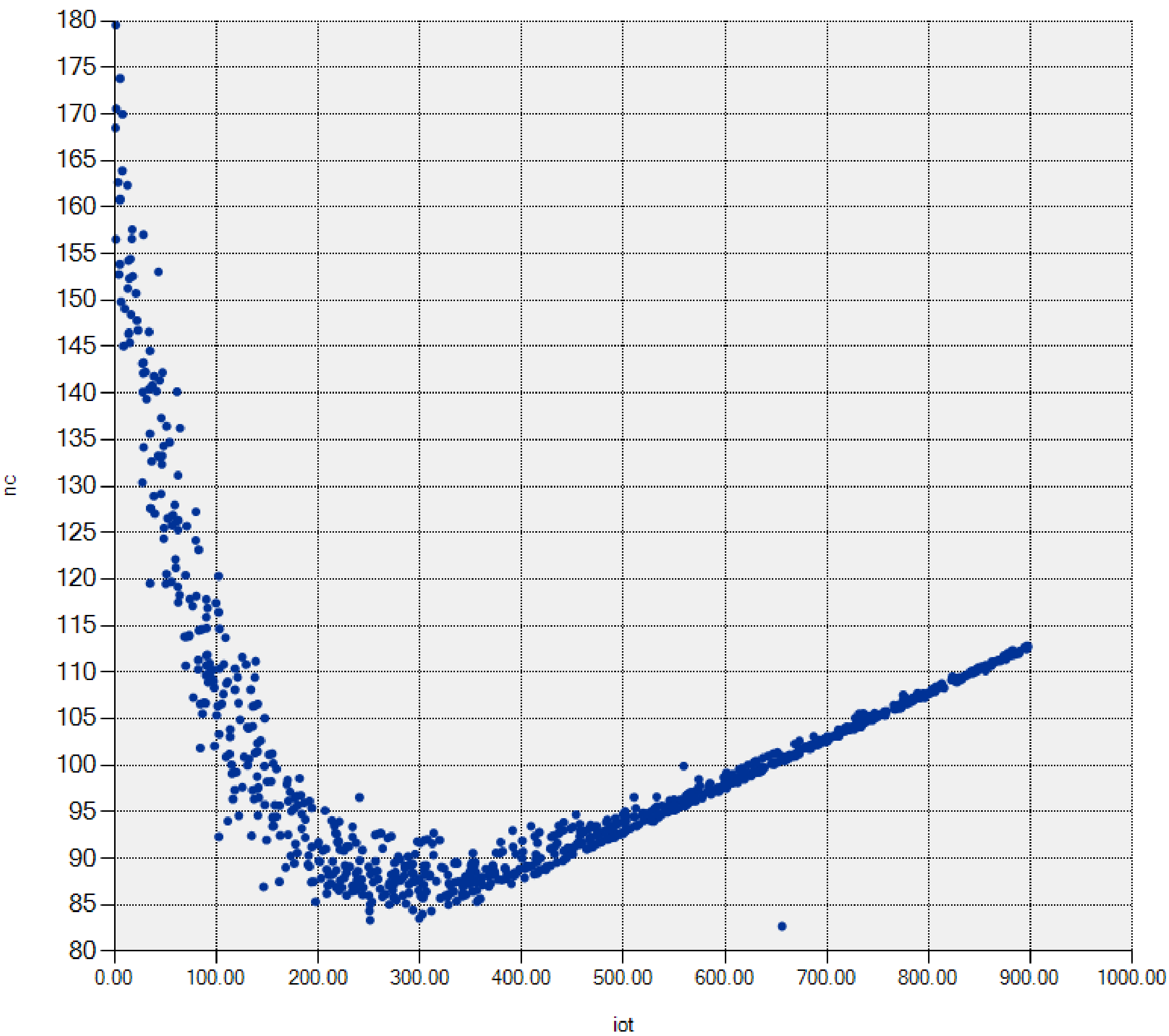
This will take a minute or two.

We are running the process 1000 * 1000 times!

Pull up the scatter graph :


Notional cost: nc = 78.789000
nc => (t
tt * crt +
tm * crn
ti * cri +
tp *crp+
nc =
Notional cost
78.789000

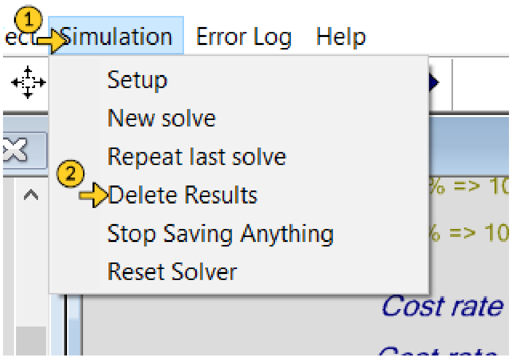

Frequency
Trace
Scatter
CDF


nc
iot
180
175
170
165
160
155
150
145
140
135
130
125
120
115
110
105
100
95
90
85
80
0.00
100.00
200.00
300.00
400.00
500.00
600.00
700.00
800.00
900.00
1000.00

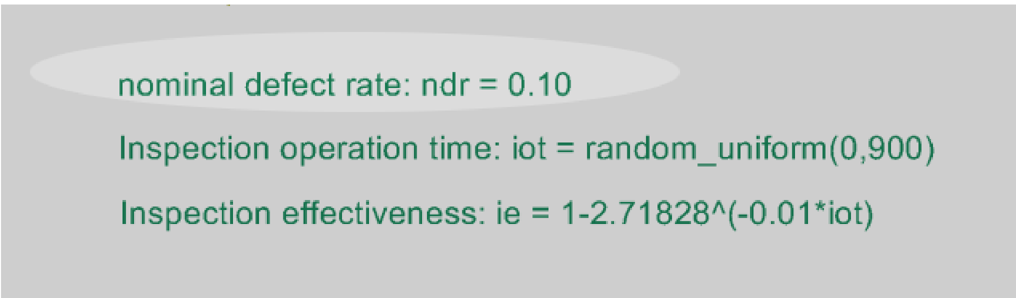
This shows that, assuming all are assumptions are correct, the optimum operating point lies between 200 and 400. It can also be noted that the costs rise much more steep to the left of this zone than to the right. If I was making a decision on this I would suggest 350 seconds is the best amount of time to invest in inspection.

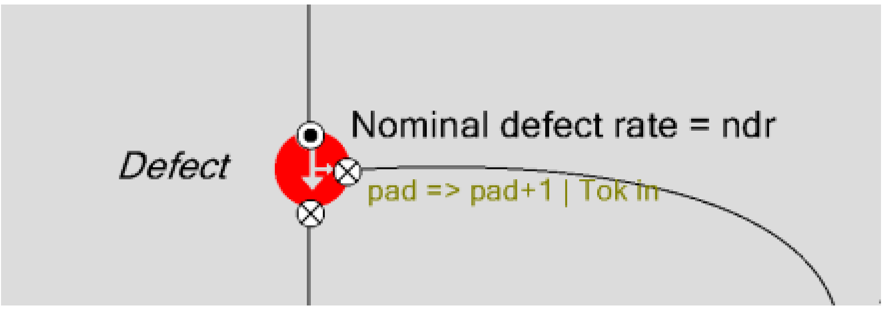
Furthermore I would also suggest from this that if the defect rate is 10 percent inspection is absolutely necessary. Lets investigate this further.

Delete your current results.

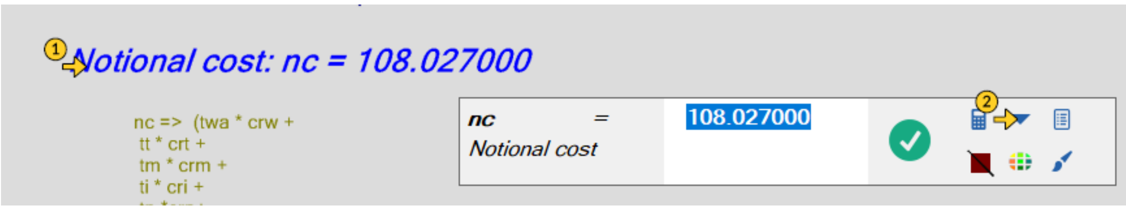


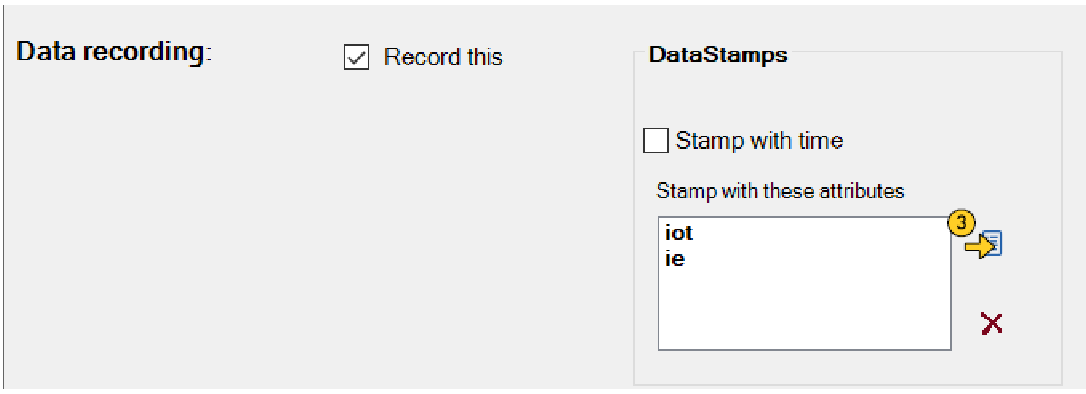
We now need to paramaterise the nominal defect rate in the block:

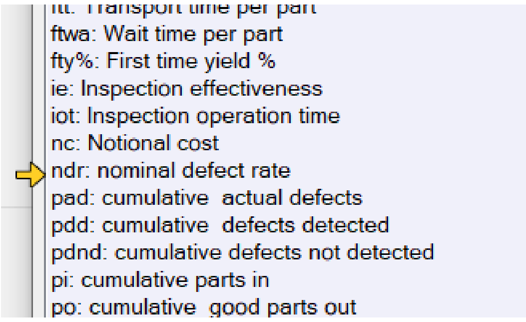




We will now add this new parameter ndr to the list of values that are saved with the computed notional cost







Result :

Ndr (nominal defect rate) is currently set to 0.1 or 10 percent. Run the solve 1000 times as it is.

Then set ndr to 0.5 and run it another 1000 times

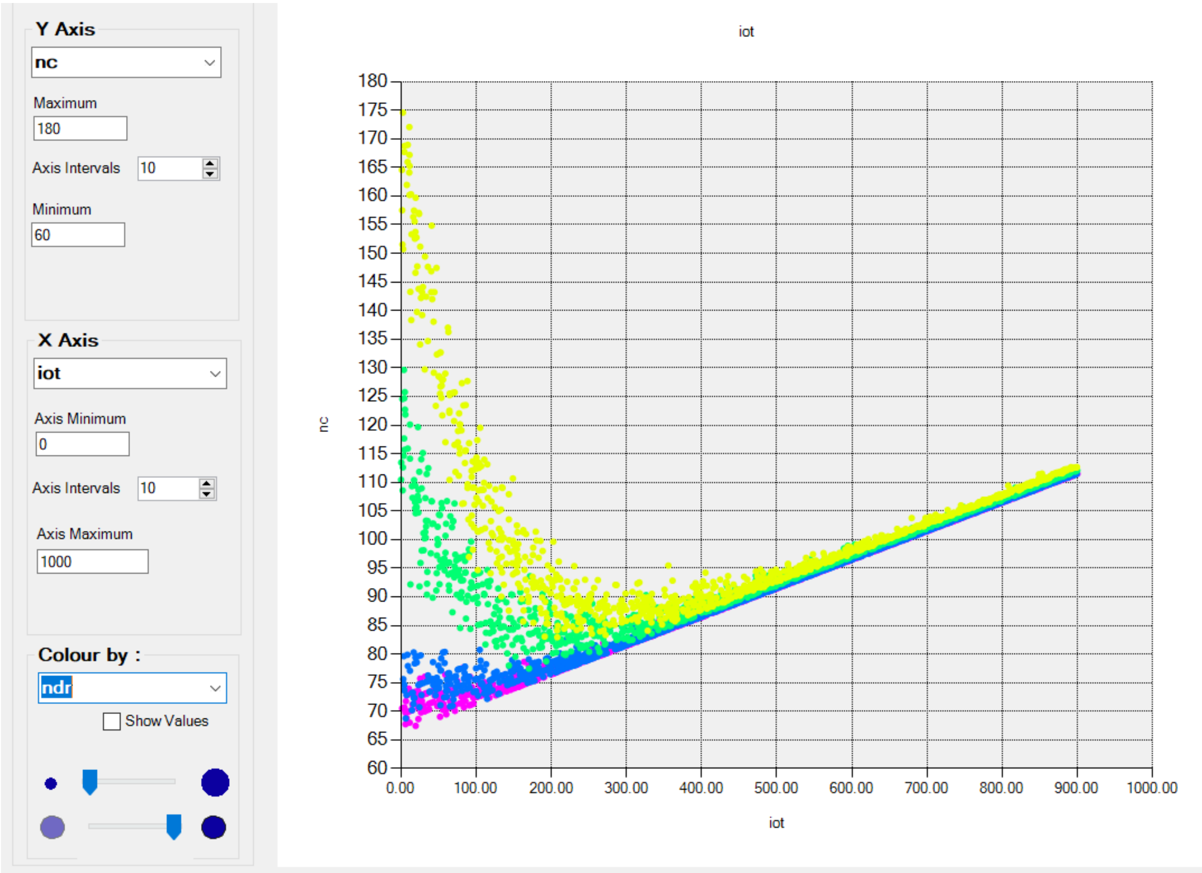
set it to 0.05. and run it a further 1000 times.

Set it to 0.001 and run it a further 1000 times.

Alternatively open up the model chapter5ex1b if you are in a hurry!

Open up the scatter graph for notional cost:

Note the X axis and colour by settings :



What does this graph tell us :

1) The minimum cost of the part is around 85 units if the defect rate is 10 percent. It is around 70 units if the defect rate is is 0.1 percent. Savings opportunities of 15 units a part are available if we can control the process to produce almost zero defects.
2) If the defect rate is 10 percent , the optimal inspection effort is 350 seconds per part. If the defect rate is 0.1 percent the optimal inspection effort is around zero. In other words if we can be sure that the defect rate is very low, any inspection simply adds cost.

This brings us on to the observations made in the introduction. Modern quality control is much more about aiming for zero defects. The aim of inspection is to check the process is running as designed (with zero or very low defects). Not to find the defects in the part.

Review

So in this chapter we have focussed on the concept of notional process costing. The aim is not to finalise a piece part cost but to establish the idea of process cost. Once the idea is established, different process designs or parameter sets can be compared with each other and optimisation schemes can be devised. The aim is to improve the quality of decision making and get a better broad idea of how the parameters affect the system.

Key Engineering concepts :

- Notional costs
- Cost minimisation

Key Aliquis concepts:

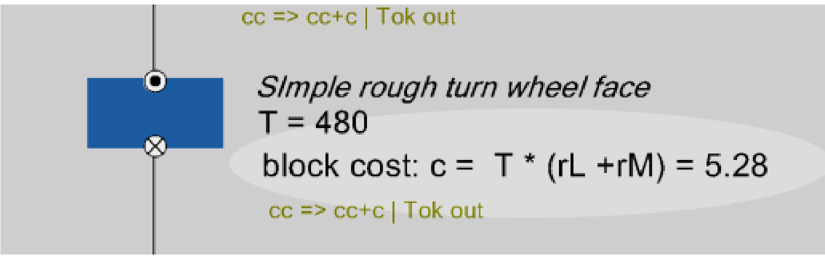
- User variables
- Assignments
- Setting up experiments
- Saving and visualising results

Questions :

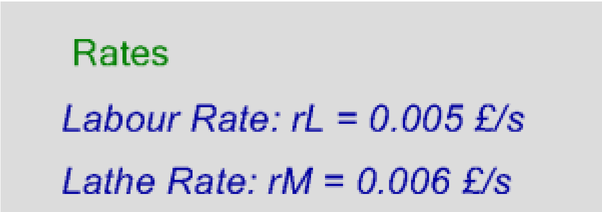
1. You have seen how the operation of the model can be tied to the idea of notional costs and waste. What other numerical results would you consider analysing in this way.
2. If you wanted to apply more control over the cost rates for different items in the model, how could you do it?
3. This chapter has shown in principle how parameters can be optimised against an output. How would the various factors inherent in the model affect where the optimisation points are found.

Answers :

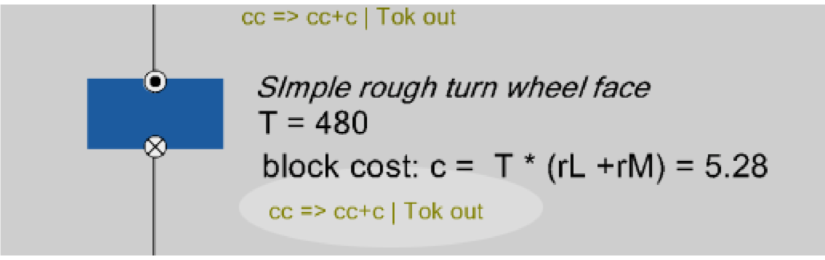
1. Here are a few ideas :
    a. Consumption of services such as electricity and water.
    b. Production of global warming gases
    c. Labour requirements
2. Actually there are numerous schemes for developing a process cost. We started with waste analysis and built cost onto that by assuming a cost rate for each type of operation. You might prefer a much more cost driven approach. Take a look at : *Chapter5Example1*. In this each block has a user variable for cost. If we look at this one :



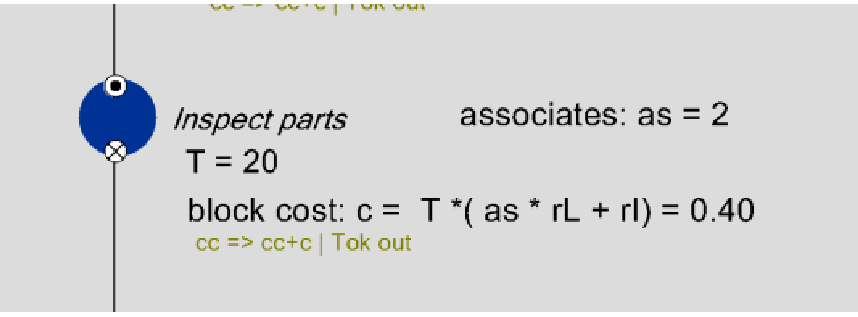
The cost is calculated from the time used and the labour plus machine rate :



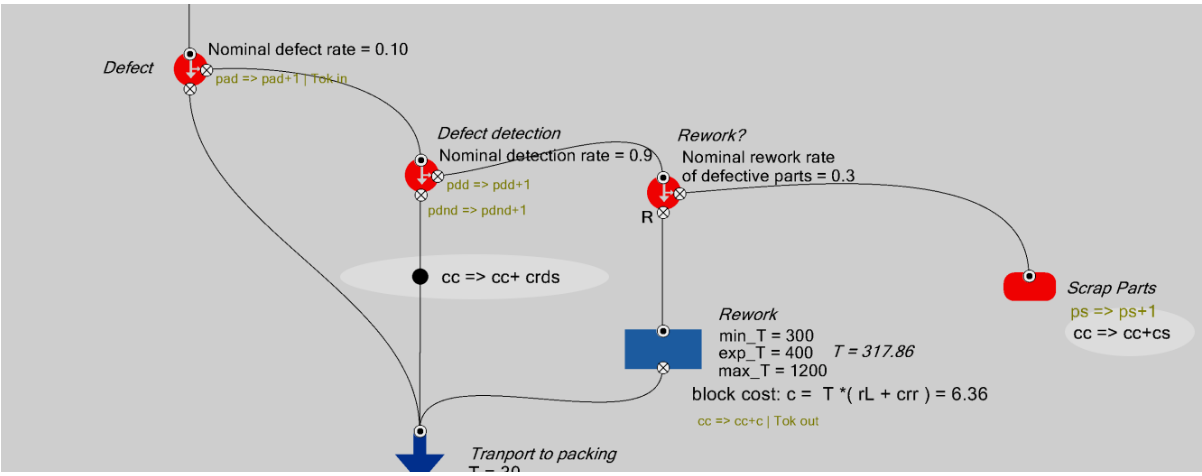
It is then cumulated to the variable cc :



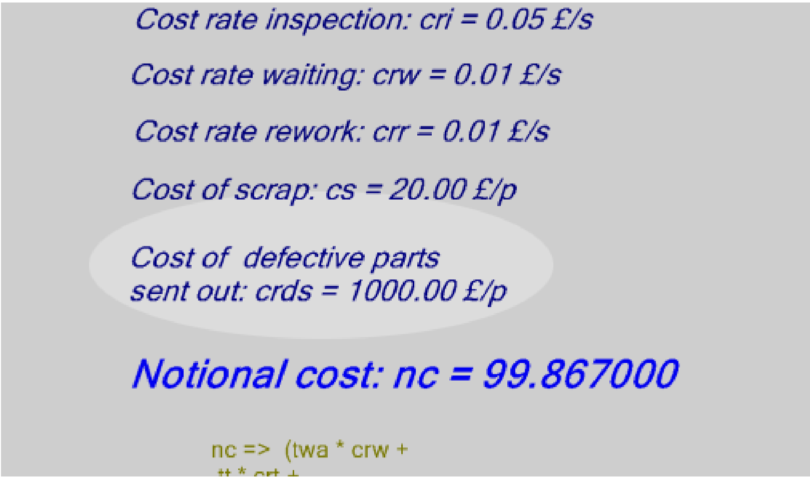
All the blocks calculate a cost and contribute to a cumulator in a similar way. Some do not include the machine rate, and the inspection block has two associates working so the cost is doubled:



The scrap is and non detected defect parts are costed as they are detected :



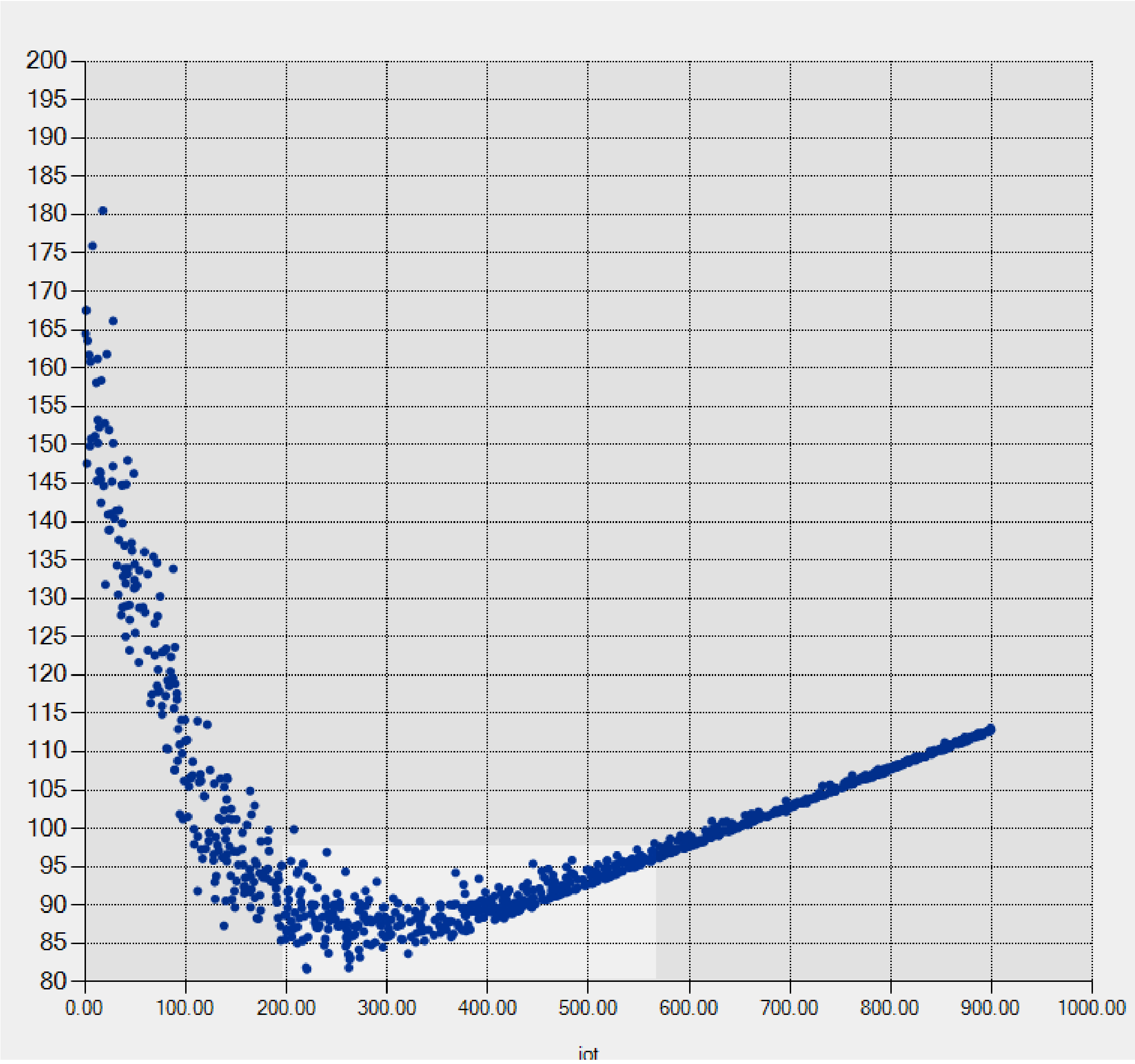
3. The optimised point in this case is strongly dependent on :
    a. the assumed cost of a defect part getting through the system – because it is high



#

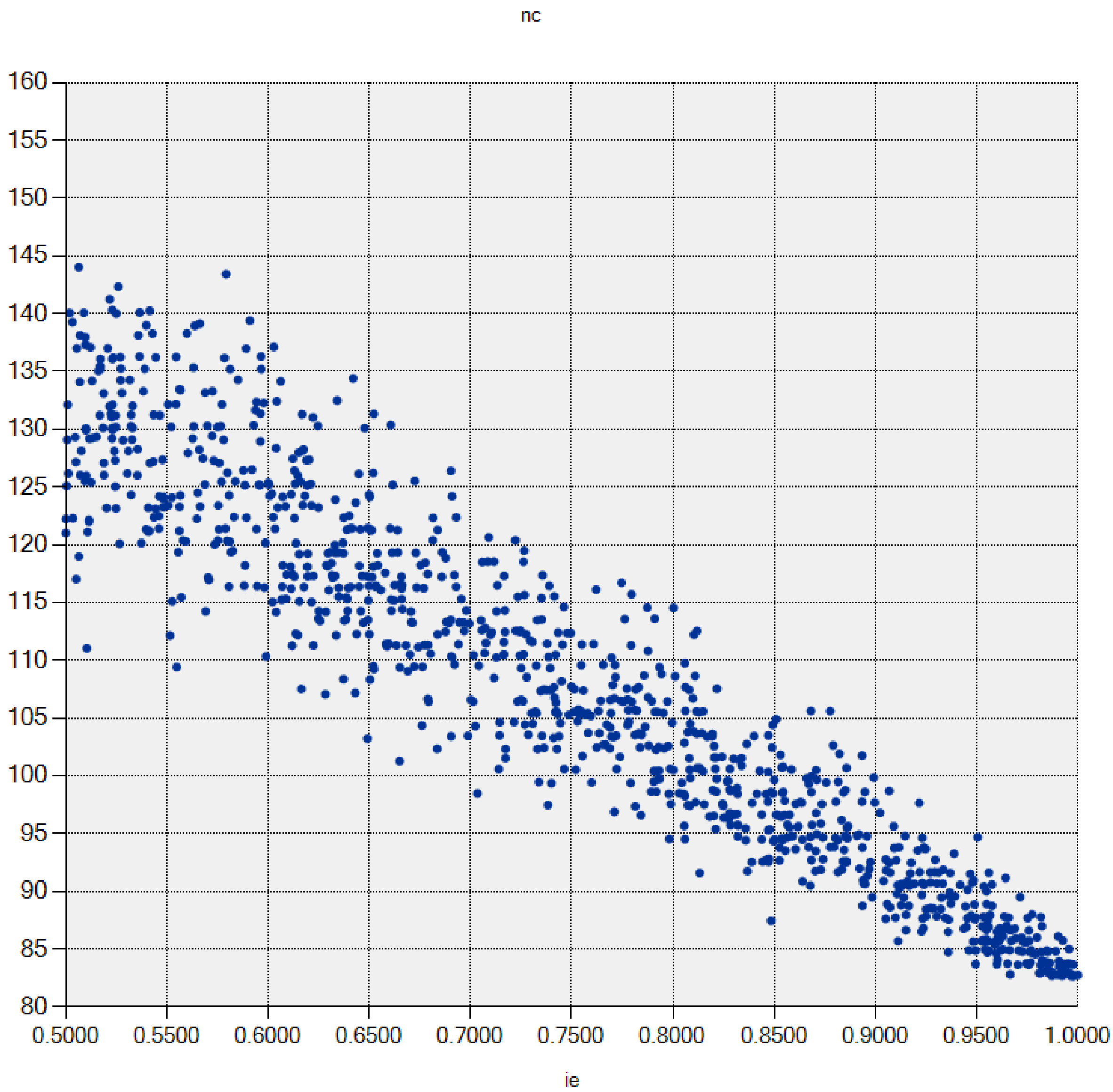
    b. the formula connecting the inspection time to the inspection effectiveness.

It has to be said that in a real world scenario these items would be hard to get right The cost of defective parts getting through is extremely subjective. The link between time and inspection effectiveness is currently arbitrary, but it could be tied down much more by modelling alternative actual inspection processes. Does this matter? Well looking at the result scatter graph we really should talk about an optimised region rather than an optimised point:



What is clear from this analysis : If you have a ten percent defect rate and if defects sent to customers are expensive – you need to put a good deal of effort into 100 percent inspection. But not a ridiculous amount.

Another useful graph is this one :



The x axis here is the inspection effectiveness which is allowed to vary between 0.5 and 1 with a fixed inspection time. In other graph shoes that if you could directly control inspection effectiveness directly, what could you gain as it approaches 1. Or what is the cost of poor inspection effectiveness. (130-82) . This is called a parameter sensitivity study. We just allow one parameter to float while holding everything else constant. We then see the effect of the isolated parameter on system performance. I would suggest that in these conditions , this is a very important parameter and it should be given due attention!