Manufacturing systems

They typically are complex systems in which the change of the performance of one component will influence the performance of other components, not necessarily in a desired way.

Example: in an assembly line, station S_3 is the bottleneck, slowing the final output by 30%. We add another station S_{3a} to remove the bottleneck. Unexpectedly, the output of S_4 (next in line) falls by 40% because the increase of output from S_3 and S_{3a} overflows the storage capacity of parts waiting to be handled by S_4.

Benefits of simulating

- Increased throughput.
- Reduced inventories of parts.
- Increased utilisation of machines and employees.
- Increased on–time deliveries of products to customers.
- Reduced expenses (capital or operating).
- Confidence that a proposed system will work as planned.
- A better understanding of the way a complex system operates (might result in new ideas).
- Identify weaknesses in a proposed system without having to build it.

As always there are three components involved in simulation:

- Modelling the non–deterministic aspects of a system (“input”).
- Modelling properly the system’s behaviour (more randomness).
- Interpreting simulation output.

The first two categories share some properties and (e.g. interarrival time and service time) but differ in methodology.

Modelling system's behaviour

There are many sources of randomness in a system:

- Obviously, input to the system.
- Machine failures (these may be exponential but they have more complex effects on subsequent stations).
- Handling times (processing, inspecting, etc.).
- Handling failures and rework times.
- Times to fix failures.
- Times to reset a machine to do a different task (including a pdf of the "human error" variable).

Machine failures

consider the following scenario:

> A company is going to buy a new machine tool. The vendor claims that the machine will be down 10% of the time. The mean–time–between–failures is unknown.

Naive analysis suggests that one should simply subtract 10% from the machine productivity. It works fairly well in the case of average throughput but not so well for other, possibly crucial, measures.

Three models are considered: **(a)** 1h breakdown every 10h, **(b)** 6min breakdown every hour and **(c)** no breakdowns but machine works at 90% capacity. All three are "legal" models reflecting the vendor's claim.

Measure	MTBF		
	60/600 min	6/60 min	**–10%**
Throughput	1908	1913	1915
Ave Time in system	35.1	10.3	5.6
Max time in system	256.7	76.1	39.1
Average queue length	27.2	7.3	3.6
Max queue length	231	67	35

The queue length measure indicates how many work orders were waiting to be handled.

The three scenarios considered in the previous table illustrate how large the difference between reality and model can be.

The table suggests that a **120/1200 min** model would give much higher (possibly 2–3 times than 60/600) waiting times. And what about randomised failure times?

Failure time

A machine can be in one of three states:

Busy

Idle

Down

When a machine is idle, it is hard to imagine that it will break down (unless there are severe maintenance problems). When it already is down, it will certainly not break down (“human error” being an exception).

Assuming exponential (or Erlang) arrivals of failures, one is tempted to program the operation of a machine as:

```
UP:
    EQ.Insert( NextFailure( Now ) , DOWN ) ;
    IfThereIsWork()
            EQ.Insert( Done( Now ) , FINISH ) ;
    else ;
    break ;
DOWN:
    Do something about that FINISH event
    EQ.Insert( Repair( Now ) , UP ) ;
    break ;
```

```
FINISH:
    IfThereIsWork()
        EQ.Insert( Done( Now ) , FINISH ) ;
    else ;
    break ;
```

- A machine usually breaks down when operating. That leaves unfinished work that has to be discarded or restarted (depends on the application).
- Repairing a machine is a service and should be modelled as such: the repair team may be busy at the time of failure and will not be able to start repairing the machine immediately.

```
DOWN:
    Do something about that FINISH event
    EQ.Insert( Now , FAILURE.ARRIVAL ) ;
    break ;
FAILURE.ARRIVAL:
    if RepairCrewIdle
        EQ.Insert( Repair( Now ) , UP ) ;
    else
        RQ.Add( ThisMachine ) ;
    break ;
```

Here, **RQ** is the queue of machines waiting to be repaired (it probably is a FCFS queue, but not necessarily).

Still leaves the problem of the unfinished work.

To solve problems like failure–completion interaction one has to change the design of a simulator.

One method is to model all activities as pairs **service queue**+**event** to allow to specify the next event related to a particular server. Note that the EQ is not a **service queue**; it is merely a sorting device (to give the next event in time ordering).

Another is to allow to modify the descriptions of pending events (i.e. events stored in the EQ).

```
ThisMachine⟶DOWN:
    NEv = Deletemin( ThisMachine ) ;
    ThisMachine.WQ( NEv⟶Order ) ;
    EQ.Insert( Now , FAILURE.ARRIVAL , ThisMachine ) ;
    break ;
```

Here, **WQ** is the queue of orders waiting to be fulfilled by **ThisMachine** (can be adapted to model a pool of machines).

Two programming points are important in this case: (1) never to ‘preload” events past the next of a kind (2) the code works only if idle machines never break down.

If the order is to be discarded instead, the insertion of the work order should be removed.

```
ThisMachine→DOWN:
    NEv = Deletemin( ThisMachine ) ;
    EQ.Insert( Now , FAILURE.ARRIVAL , ThisMachine ) ;
    break ;
```

Another way of looking at machine failures is to assume (usually correctly) that machines fail as a result of being used. Then, the proper unit of time to measure the MTBF is *the number of parts produced* rather than conventional time.

Having several time axes simultaneously (one “real” time and one for each machine) is a major challenge.

Work stoppages etc.

Work stoppages are an integral part of the manufacturing process. Trying to incorporate them into a comprehensive industrial simulator is not simple.

A crew unit that stops working can be modelled as “DOWN” in a way similar to broken machinery, but with one distinction: there is no way to propose a pdf for stoppage time.