Documentation of DES Models for Manufacturing System Life Cycle Simulation

Jan Oscarsson
Matías Urenda Moris

Agenda:
- Life Cycle Simulation
- User requirements
- Model documentation
- Conclusions
A simulation model

4 Procedure my_simulation()
  5 Var
  6  container_pclass : Part_class
  7  container_part : Part
  8  partC_pclass : Part_class
  9  partC : Part
  10  partM_pclass : Part_class
  11  partM : Part
  12  elem : Element
  13  req : Request
  14  EndVar
  15 Begin
  16  VWait until (elem->in[1])>num_out_parts > 0 IN STATE WAIT_BLOCK
  17  container_part=elem->in[1]->out_parts[1]
  18  container_pclass=container_part->pclass
  19  if (req->pclass) then
    20    order_fulfilled(req)
  21  Endif
  22  if (container_part->in[p]>0) then
    23    work normal[33,3,1]
    24    container_part = elem->in[1]
    25    partC = container_part->parts[2]
    26    remove_part_from_container_part(partC)
    27    transfer partC thru output 2
    28    delay 0.1
    29    pass()
  30  else
    31    VWait until (elem->in[2])>num_out_parts > 0 IN STATE WAIT_BLOCK
    32    if (container_part->out_parts<3) then
    33      work normal[33,3,1]
    34    else
    35      work normal[43,3,1]
    36      transfer partC thru output 2
    37      delay 0.1
    38    endif
    39    Endif
    40    delay 0.1
    41    partM<elem->in[2]->out_parts[2]
    42    if (req->pclass) then
      43      order_fulfilled(req)
    44    Endif
  45  endif
  46  Add_part_to_container_part (container_part, partM)
  47  transfer container_part thru output 1
  48  delay 0.1
  49  pass()
  50  delay 0.1
  51 END
Manufacturing System Life Cycle

Manufacturing systems undergoes changes from time to time.

20-30 years!!
Digital Factory

Digital factories are used by production engineers to support engineering work.
The System and Time

Changes are not made continuously, only when required.
The Model and the Life Cycle

The model ought to be:

- Re-used – during new projects
- Updated – when required
- Understood – by all people involved
- Inherited – by others than the initial developer
Life Cycle Simulation

- Simulation models may be highly complex
- Simulation models contain much information
- People tend to retire or move to new posts
- It is costly to recapture knowledge about the model

Life cycle simulation and re-usability makes documentation of simulation models increasingly important.
Who needs the documentation?

- The programmer (original Simulationist)
- Engineers or new simulationist who will study or re-use the model
- Others who need to understand the main frame, but perhaps not all the details
Criterions to Consider in the Selection of Notations

Neutral notation
- Not limited to any specific languages, software or systems.

Generic notation
- Able to describe different systems of various purpose, complexity and scope.

A Recognised notation
- The notation should to some degree be accepted and recognised, something that will improve communication and support maintenance during its life cycle.
Criterions to Consider in the Selection of Notations

User friendly
- Should be intuitive.
- Should help in overcoming the difference between the abstract code level and the natural language.

Descriptive in several levels
- Give the possibility to modulate and describe different levels of abstraction and a top-down or bottom-up approach.

An in-house competence
- Companies may have certain standard notation.
Recommendations

Low-level documentation
- Flowcharts
- Code comments
  - UML diagrams, structure and behavioural diagrams.

Conceptual documentation
- IDEF 0
  - Describe flow, choose and adequate abstraction level.
- Flowcharts
  - Helps to visualise a more physical structure

General documentation
- Use model animation when accessible or flowchart.
A Flow chart is defined as a "pictorial representation describing a process being studied"

Strengths:
- Descriptive and intuitive syntax.
- Well recognised.

Weakness:
- Difficult to track in large diagrams.
- Ambiguous interpretation
IDEF0

IDEF0 - *Describes systems in terms of their activities and information flows*

Strengths:
- Flexible (decomposition)
- Recognised in engineering
- User friendly

Weakness:
- Does not define structures
UML - Unified Modeling Language

Strengths:
- Complete notation
- Object oriented

Weakness:
- Complex
- Not widely used by manufacturing engineers
The documented model
-- Description: Process Logic for Machine Assemble
-- The process logic for the Machine Assemble contains both the process and route logic for
-- the element. It contains three different process, one for the fixture that is going to be
-- changed, the second for the fixture that is empty. The third process is the regular
-- process which is the longest process and demands both that the harden cabinet is removed
-- and the new cabinet put in place.
-- Called by: Machine_Ascsemble_1 and _2
-- Version: QUEST 5.3
-- Date: October 2001
-- Author: Mateus Orlando Moris

Procedure my_machine_proc()

Var
container_pclass = Part_class
container_part = Part
partC_pclass = Part_class
partC = Part
partD_pclass = Part_class
partD = Part
elem = Element
req = Request

Begin
wait until <elem->in[1]-->num_out_parts > 0 IN STATE WAIT_BLOCK
container_part = <elem->in[1]-->out_parts[1]
container_pclass = container_part->pclass
require part container_pclass thru 1
req = match_order(container_part) -- the request attached to the part is match against\nif(req=NULL) then
  order_fulfilled(req)
endif
if (container_part->byte="true") then -- if the part tool is going to be changed new cabinet
  work normal(35, 3, 1)
  container_part = <elem->in[2]-->parts[1]
  partC = container_part->parts[1]
  remove part from container_part(partC)
  transfer partC thru output 1 -- the hardened cabinet is placed on the conveyor system
  transfer container_part thru output 1 -- the fixture/tool is removed to the next station
  delay 0.1
  pass()
else
  wait until <elem->in[2]-->num_out_parts > 0 IN STATE WAIT_BLOCK
  if(container_part->num_parts < 2)then -- If the fixture is empty, the process is shorter that the regular one
    work normal(35, 3, 1)
  else
    partC = container_part->parts[1]
    remove part from container_part(partC)
    work normal(45, 7, 1)
    transfer partC thru output 2
    delay 0.1
  endif
  delay 0.1
  pass()
  partC=<elem->in[2]-->out_parts[1]
  require part partC->pclass thru 2
  req = match_order(partC)
  if(req=NULL) then
End procedure my_machine_proc;
Conclusions

- What is a good notation? One can never be totally objective.
- Technical perfectness is not always the best choice. Usefulness and acceptance is important.
- The proposed notation has proven to work in industry.
Thank you for listening!
Questions?