The Impact of Simulation on the Design of Logistic Systems

G. Lodewijks

Inaugural Smart Workshop, Wollongong, Australia, September 15, 2009
Contents

1. Introduction
2. Logistics of Baggage Handling Systems
3. New Concepts for City Logistics
4. Logistics of Marine systems
5. Logistics of Bulk Handling Systems
6. Logistics of Container Handling
1. Introduction
Introduction

Recipe Proces-Inter

**Modeling Recipe:**

- What are the system elements?
- Describe the behavior (processes of the living ones)
- Carry out the runs
Conceptual process interaction modeling

**Modeling Recipe:**
- What are the system elements (syst. Boundary)
- Describe the behavior (processes of the living ones)
- Carry out the runs

**Elements:**
- Cranes
- Ships with containers
- ‘Ship generator’

**Crane-Process:**
Repeat the following actions:
- Wait while the ShipQueue is empty
- Remove the first ship from the ShipQueue
- Work the unload time of the ship
- Remove the ship from the system

**Shipgenerator Process:**
Repeat the following actions:
- Wait until the next ship should arrive
- Create a new ship with its load
- Put the ship in the ShipQueue

**Modeling Recipe:**
- What are the system elements (syst. Boundary)
- Describe the behavior (processes of the living ones)
- Carry out the runs

**Configure a model with 3 cranes**
**Vary the arrival pattern and the load distributions**
**Run 100.000 time-units**
**Read the average ships waiting time**
Processes

**Crane-Process:**
Repeat the following actions:
- Wait while the ShipQueue is empty
- Remove the first ship from the ShipQueue
- Work the unload time of the ship
- Remove the ship from the system

A process describes the **behavior** of a class.
=> For each class only one process description needed

The *Class definition* + the *(filled)* **Process** + *user defined items* give the ‘blueprint’ of the *structure and behavior* of each member of the class
Pre-defined classes

For process interaction modeling we make use of *pre-defined* basic classes with a standard *functionality* and *extendibility*.

Some pre-defined classes are
- `SimElement`
- `Queue`
- `Distribution`

Model elements inherit all attributes and methods of a basic class.
Simulation Case

A simulation case is constructed by creating the necessary elements according to the class definitions and starting the relevant processes.
Model design in 3 stages

- Intuitive Conceptual model
  - Process-Interaction method
  - Plain English/Dutch

- Process Model
  - limited Vocabulary
    - E.g. 'Hold'
    - 'EnterQueue'

- Computer Model
  - Tomas/Delphi
  - Or other P.I.supporting language
From Conceptual to Process model

Intuitive Conceptual model

Process Interaction method
Plain English/Dutch

Process Model
limited Vocabulary
E.g. 'Hold'
'EnterQueue'

Computer Model

Tomas/Delphi
Or other P.I.supporting language
In Process model: Only words from vocabulary!

<table>
<thead>
<tr>
<th>simElement</th>
<th>Queue</th>
<th>Distribution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create</td>
<td>Create</td>
<td>Create</td>
</tr>
<tr>
<td>Process</td>
<td>Length</td>
<td></td>
</tr>
<tr>
<td>Hold</td>
<td>FirstElement</td>
<td>Sample</td>
</tr>
<tr>
<td>Standby</td>
<td>LastElement</td>
<td>Mean</td>
</tr>
<tr>
<td>IsIn</td>
<td>MeanLenght</td>
<td></td>
</tr>
<tr>
<td></td>
<td>MeanWT</td>
<td></td>
</tr>
<tr>
<td>EnterQueue</td>
<td>AddToTail</td>
<td></td>
</tr>
<tr>
<td>LeaveQueue</td>
<td>Remove</td>
<td></td>
</tr>
</tbody>
</table>

And many more!
Conceptual process interaction modeling

**Modeling Recipe:**
- What are the system elements (syst. Boundary)
- Describe the behavior (processes of the living ones)
- Carry out the runs

**Elements:**
- Cranes
- Ships with containers
- ‘Ship generator’

**Crane-Process:**
Repeat the following actions:
- Wait while the ShipQueue is empty
- Remove the first ship from the ShipQueue
- Work the unload time of the ship
- Remove the ship from the system

**Shipgenerator Process:**
Repeat the following actions:
- Wait until the next ship should arrive
- Create a new ship with its load
- Put the ship in the ShipQueue

Configure a model with 3 cranes
Vary the arrival pattern and the load distributions
Run 100,000 time-units
Read the average ships waiting time
From Conceptual to Process model
Element (class) Definition section

Classes:
Shipgenerator = SimElement
  - InterArrTimeDistr = Distribution
  - HelpTimeDistr = Distribution
  - Process
QuayCrane = SimElement
  - Process
Ship = SimElement
  - HelpTime
ShipQueue = Queue

Elements:
- ‘Ship generator’
- Cranes
- Ships with containers
- Waiting Queue
From Conceptual to Process model
ShipGenerator Process

Shipgenerator Process:
Repeat the following actions:
  Wait until the next ship should arrive
  Create a new ship with its load
  Put the ship in the ShipQueue

ShipGenerator.\textit{Process}
Repeat
\textit{Hold} (InterArrTimeDistr.\textit{Sample})
newShip=Ship.\textit{Create}
newShip. HelpTime= HelpTimeDistr.\textit{Sample}
ShipQueue.\textit{addToTail}(newShip)

NB: ‘dot’ notation:
newShip\textbullet\textit{HelpTime} equals “Helptime of newShip”
newShip=Ship\textbullet\textit{Create} means “use Create function of ‘Class’ ship to create a ship”
ShipQueue \textbullet\textit{addToTail}(newShip) : the ‘Class’ Queue owns an addToTail function
**Crane-Process:**
Repeat the following actions:
- Wait while the ShipQueue is empty
- Remove the first ship from the ShipQueue
- Work the unload time of the ship
- Remove the ship from the system

QuayCrane.*Process*

*Repeat*

While ShipQueue.*Length* = 0
Standby
myShip=ShipQueue.*FirstElement*
myShip.*LeaveQueue*(ShipQueue)
Hold(myShip.HelpTime)
myShip.*Leave*
From Conceptual to Process model
make a case and carry out a run

Configure a model with 3 cranes
Vary the arrival pattern and the load distributions
Run 100,000 time-units
Read the average ships waiting time

ShipQueue=Queue. Create
Repeat(number of QuayCranes)
  QC=QuayCranes. Create
  QC. Start

ShipGen=ShipGenerator. Create
ShipGen. InterArrTimeDistr=Distribution. Create
ShipGen. Start

Startsimulation
From Process model => computer model
(=language specific! We use Delphi / Tomas)
Definition of elements classes

TYPE
  QuayCrane = class(TomasElement)
    published
      procedure Process; reintroduce;
    end;
  Ship = class(TomasElement)
    public
      HelpTime: Double;
    end;
  ShipGenerator = class(TomasElement)
    public
      InterArrTimeDistribution : TExponentialDistribution;
      ShipHelpTimeDistribution : TExponentialDistribution;
    published
      procedure Process; reintroduce;
    end;
  ShipQueue : TomasQueue;

classes
Shipgenerator = SimElement
  – InterArrTimeDistr = Distribution
  – HelpTimeDistr = Distribution
    – Process
QuayCrane = SimElement
  – Process
Ship = SimElement
  – HelpTime
ShipQueue = Queue
From Process model => computer model: Crane Process

QuayCrane. **Process**

**Repeat**

**While** ShipQueue. **Length** = 0  **Standby**

myShip=ShipQueue. **FirstElement**

myShip. **LeaveQueue**(ShipQueue)

**Hold** (myShip.HelpTime)

myShip. **Leave**

```pascal
Procedure QuayCrane.Process;
var Myship:Ship;
begin
  while true do
  begin
    standby;
    Myship:=ShipQueue.FirstElement;
    ShipQueue.Remove(Myship);
    hold(Myship.HelpTime);
    Myship.Destroy;
  end;
end;
```
From Process model to Computer Model make a case and carry out a run

```pascal
(* MAKE the ELEMENTS and START THEIR PRODUCTION *)

Procedure InitializePORT;

var
  Seed1,Seed2 : Integer;
  newQuayCrane : QuayCrane;
  Generator : ShipGenerator;

Begin
  seed1:=123456;    //seed for InterArrTimeDistribution
  seed2:=123457;    //seed for ShipHelpTimeDistribution
  decimalseparator:='.';
  tomasform.Trace:=true;
  runTime:=strToFloat(Form1.Edit2.Text);
  For i:= 1 to NCranes Do
    newQuayCrane:=QuayCrane.Create('QuayCrane');
    newQuayCrane.start(TNow);
  Generator :=ShipGenerator.Create('shipGenerator');
  Generator.InterArrTimeDistribution:=TExponentialDistribution.Create(Seed1,10)
  Generator.HelpTimeDistribution:=TExponentialDistribution.Create(Seed2,8);
  Generator.start(tNow);
  ShipQueue :=tomasQueue.create('ShipQueue');
End;
```
Extending model with Stack and Automated Stacking Cranes (2)

Simulation demo: What is the difference with the former model?
Animation Screen
Several representations of a simulation run of ECT Delta Terminal
2. 

Logistics of Baggage Handling Systems
Next generation baggage loading systems for narrow body aircraft
• Bagloader (+/- 1960)

• Conveyor to the plane (Arlanda 1992)
Baggage truck with storage capacity
With built-in Rampsnake®
Rampsnake®

Pictures actual Rampsnake®

Now working at Zaventem, Belgium
Simulation model for AAS
Next generation baggage loading systems for wide body aircraft
Transport in x-, y-, en z-direction to reach all spots in a ULD

- Gantry-robot
- Scara-robot
- Spherical robot

- Telescopic belt conveyor in portal
3. New Concepts for City Logistics
Intra/inner City Logistics
• Amsterdam city center
• 11 outlets
• Dist. to DC 13.7 km
• Dist. to outlet 2.5 km
City Logistics using Inland Waterways
450 pallets in

520 pallets out

500 pallets in

560 pallets out

590 pallets in

580 pallets out

410 pallets in
Conventionele overslag

afstand

Direct met laad/los vloer
Overslag met Tussenopslag in magazijn

Overslag met Tussenopslag in Trailers
4.

Logistics of Marine Systems
Design of a Ferry Service
Riverpusher
5.

Logistics of Bulk Handling Systems
Logistics of bulk terminals
Maintenance Logistics
100 km Phosboucraa system build by Krupp in the 70’ties
Option 1: *tag to tag communication*
Option 2: *tag to router communication*
Option chosen: *tag to tag communication*
PC

Modbus Master

Enterprise or Automation Network

server
Automatic Network Set-up
Logistics of Container Handling
Automated Transport Systems

A system with multiple Automated Guided Vehicles

Traditional:
- Central control
- Fixed guide path layout
  - hardware imposed
  - reduction of complexity
- Safety
  - sensors
  - semafore-based control
Fixed path design: single loop

Sensor based safety
Container terminal (1)

Sensors + Semaphore based safety

Network example
Container terminal (2)
Container terminal (3)

Free ranging

NOMAD

• Pedestrian behaviour model
• Cost minimizing methodology

Characteristics

• 3 layer model: strategic, tactical and operational level
• Individual (non-cooperative) route optimization
• Multiple static and dynamic ‘obstacles’
• Arrival time is crucial
• Proven performance
• Cost model can be adapted for AGVs
DEFT trajectory planning example

AGV destination duetime : $T = 45$
Time: 45
Time: 35
Time: 24
Time : 9

Costs

Time: 9.99 (8/10)
Time : 4
Time: 0

Costs

X

Y
Simulation

- Area of 6 x 8 meter
- 2 static obstacles
- 2 x 7 destinations

- Random selection of destination
- Ignoring end orientation
- Compared with mesh and loop layouts

*150.00 AGV simulation*
Application in AGV laboratory
Thank you, any questions?