COMBINING SCRIPTING AND COMMERCIAL SIMULATION SOFTWARE TO SIMULATE IN-PLANT LOGISITICS

Tim Govaert, Sven Neirynck, Sofie Van Volsem and Hendrik Van Landeghem
Department of Industrial Management
Ghent University
Technologiepark 903
BE-9052 Zwijnaarde, Belgium
e-mail: {Tim.Govaert,Sven.Neirynck,Sofie.VanVolsem,Hendrik.VanLandeghem}@UGent.be

KEYWORDS

simulation, automatic model generation, Python, Plant Simulation

ABSTRACT

In this paper we describe the use of a commercial discrete event simulation package (Siemens 2008) combined with a custom program, written in the programming language Python (Martelli 2006). Combining these two makes it possible to automatically generate a model for assembly line logistics simulation. The different stations of the assembly line, their connections and the storage near the assembly line were generated within seconds. A huge amount of time was saved compared with manual generation.

INTRODUCTION

In the truck assembly industry, in-plant transportation should be handled in the most cost-efficient way. Taking into account the fact that forklifts fail when it comes to efficiency, the truck industry started investigating the use of automatic transportation systems such as overhead conveyors or Automated Guided Vehicles.
Implementing an automatic system on factory scale requires extensive research. Cottyn et al. (2008) executed a feasibility study to investigate the possible gains and necessary investments. They suggested to build a simulation model to discover possible pitfalls of the system and to be able to dimension more in detail the amount of drop-off stations, pick-up points and carriers.
In the digital factory concept, the product and process planning can be designed and improved on all levels, by using various simulation processes. A broad overview of applicable areas for simulation is given by Kühn (2006).
In order to create a simulation model that is easily adaptable and flexible in use for the particular problem of simulating the in-plant logistics processes for truck assembly, two problems were encountered:

Modeling the assembly line A factory can contain a huge amount of workstations, which can change very often. Therefore, it was decided that the generation of the workstations in the model should be automatic and very flexible. In this way, different factories or subparts of one factory can be easily generated and simulated. A big increase in model flexibility can thus be obtained.

Modeling storage buffers at the border of line The huge diversification of the customers needs results in a huge variety of parts. These parts need to be stored at the line, in order to be consumed when the corresponding chassis passes the workstation. The parts can be bulk fed or brought at line in kits (Limère and Van Landeghem 2009). In each situation, the amount and configuration of buffers will be different. An automatic generation of these buffers could drastically decrease the modeling time.

In the next paragraph, we present the solution method. Thereafter, preliminary results results of a practical case study in the truck industry, using the proposed method, are given.

METHOD

Simulation environment

Two options exist for implementing simulations:

- develop a dedicated computer program that implements a specific simulation problem
- use a commercial simulation package to model the simulation problem at hand

In an effort to try and combine the advantages of both approaches, we propose to use a commercial simulation package[1] to simulate the plant (Siemens 2008), while generating large parts of the model via a custom program. This program is written in the programming language Python (Martelli 2006); ASCII files are used to interface between the two distinct environments.

[1]Tecnomatix Plant Simulation

Model Components

A production plant as described in the introductory section can be divided into objects belonging to four categories:

- line supplier
- transportation system
- border of line (BOL) storage
- assembly line

Line supplier
A line supplier is an entity that delivers parts to the transportation system. These entities can be warehouses, pre-assemblies or supermarkets. A supermarket is a logistical area where kitting takes place.

Transportation system
The transportation system takes care of the transport of "parts" between buffers. The origin buffer is always the output buffer of a *line supplier*. The destination buffer is the *border of line*. Identical parts, i.e. parts with the same part number, are put in a container. The transportation system transports this container over transportation tracks.

Border of line storage
The border of line (BOL) is modeled as different buffers. The parts stored in these buffers are ordered[2] (FIFO queues). On the transportation track inside the station an "offload" point is present where the container on transport can be moved to a buffer.

Assembly line
An assembly line is modeled as a series of connected stations. A station is where the assembly of the trucks takes place. A station receives a partially completed chassis from the preceding station. At the station parts are added on to this chassis. These parts are retrieved from BOL buffers locally to that station. After a fixed amount of time (the takt-time), the chassis is moved to the next station.

Model generation

Two types of objects are automatically generated: stations and buffers. To automatically generate an assembly line not only the stations need to be created but also the connections.

Assembly line generation: creating stations
Added functionality can be programmed in Plant Simulation by using so-called "methods" Siemens (2008).

[2]Parts are ordered in a container, and containers are ordered in the buffer.

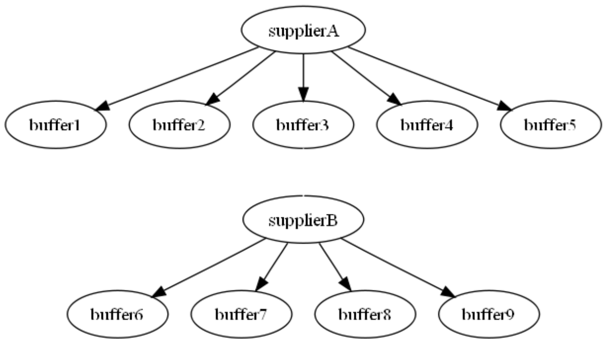

Figure 1: Relation Buffers and Suppliers

Methods are like functions: they can have input, output, program logic and can perform various tasks.
In Plant Simulation data objects exist such as tables and queues. A table object called *FactoryLayout* is created. This table is a database describing all stations. For each station there is an entry with the following information:

- station name
- x and y coordinate of the station's location center
- type of station, determined by number and place of BOLs
- preceding station
- a boolean value indicating whether transportation tracks have to be generated

A method was written which uses the *FactoryLayout* table as input. For each entry in the *FactoryLayout* table, a station at the given coordinates is created. If necessary, transportation tracks are also created.

Assembly line generation: creating connections
Two types of connections need to be created:

- Connections between output of one station to the input of the next station to model the flow of the chassis through the factory.
- Connections between the tracks of the transport system:
  - The tracks need to form a closed loop
  - Junctions to reach the drop-off points inside the stations need to exist

Not all connection information can be stored in the *FactoryLayout* table. This table only stores the stations predecessor. Examples of extra connections are: feeder lines, merging of two lines, connectivity of the last station to the exit of the factory. The remaining connections are put in another table called *Connections*. Another "method" uses this table to create the remaining connections.

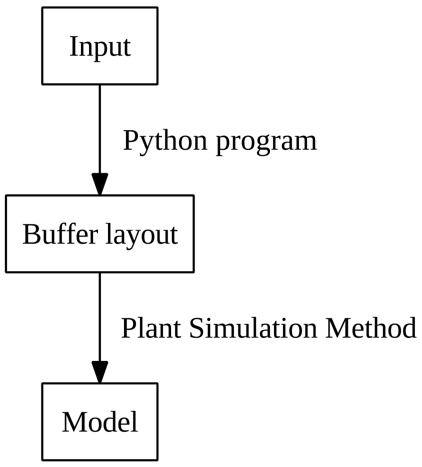

Figure 2: Data flow

Buffer generation
The buffers from the border of line as well as the assignment of these buffers to the different parts is dynamic. It varies between different simulation runs.
One of the goals of the eventual model is to explore different "kitting" combinations. Each combination has its impact on the buffers. A suitable model therefore needs to be dynamic in buffer allocation. To assign buffers to the stations we need to have the information about what trucks need which parts at which station. We need to process this information and calculate the buffer assignment. This buffer information is then fed into the model.
Information about the trucks is needed. A Python program transforms this into output which contains information about the buffer layout. A Plant Simulation "method" will use this buffer layout information to create the model. (See Figure 2)
During the simulation, parts will be consumed at the stations. As a consequence buffers need to be refilled. Some parts will come from a pre-assembly, others will be a kit from a supermarket and some can just be retrieved from a warehouse. Thus, information is needed on the in-plant origin of the different parts.

Input: The Python program requires the following input:

- *Parts Requirements List:* A file describing for each truck all the parts needed for assembly and the station where the assembly takes place
- A file describing the in-plant origin for each part

Output: The external[3] program written in Python transforms this information into the layout of the buffers. It generates the following output:

[3]External to Plant Simulation

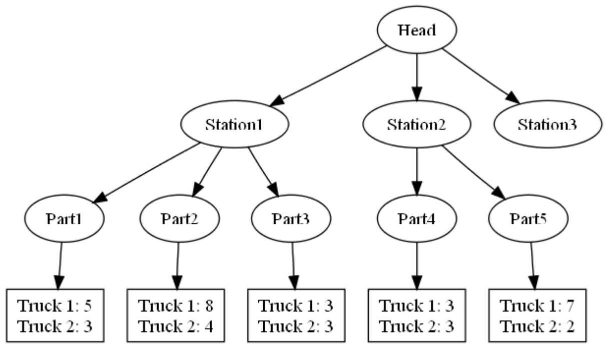

Figure 3: Example tree

- for each station: the number of buffers;
- for each buffer: a partnumber, supplier and the number of parts to be assembled for each truck.

Algorithm: The algorithm to calculate the buffer layout consists of two phases. In phase 1 the input is parsed into a tree. In phase 2 the tree is written out into individual files which will serve as input for Plant Simulation.

Algorithm 1 The two phases

```
Create empty tree
for Each Line in Parts Requirements List do
  Parse station name
  Add truck, station, part to the tree
end for
for Each station in the tree do
  Count parts
  Write number of parts to file
  for Each part in the station do
    Create buffer output file
    for Each truck do
      Write truck and number to the output file
    end for
  end for
end for
```

We use the following I/O files:

"Stationname"."L/R".txt contains the number of buffers at this station. (Some stations have a left and right side indicated by an extra character L or R).

"Stationname"."L/R" "buffernumber".txt is the table stating the quantity used for each different truck.

"Stationname" "buffernr".supplier.txt is a file with the in-plant supplier's address and the partnumber for that buffer.

Inside the station there is a “method”, *configure*, which configures the station. This method looks for the files *“Stationname”.“L/R”.txt* and creates the necessary number of buffers. For each buffer it reads the file *“Stationname”.“L/R”“buffernumber”.txt.* It also reads in the *partnumber* and *supplier.* This information is needed by the model during simulation.

CASE STUDY

We implemented these two techniques to create models for the supply chain logistics for a large European truck factory.

Assembly line layout

The *FactoryLayout* table consists of 300 entries. Three different Plant Simulation station objects are used. The *connections* table consists of 75 extra connections. Some tracks and warehouses are manually placed in the model. Generation of the model takes only a couple of seconds on a standard desktop.

Buffer layout

Using the information of the last 10 trucks assembled at the plant, the Python program created 8793 files in less than a minute. The Python program creates a file for each station and a file for each buffer in the stations.

CONCLUSIONS

We have successfully created an interface between a commercial simulation package (Plant Simulation) and a scripting language (Python).
Within Plant Simulation we are able to generate the factory layout in a very flexible and cost effective way.
Using the scripting language Python we are able to dynamically create a model (buffer configuration) using a few input files. This would have not been possible using a conventional approach without resorting to an external programming language.

REFERENCES

Cottyn J.; Govaert T.; and Van Landeghem H., 2008. *Alternative line delivery strategies support a forklift free transition in a high product variety environment.* In *Proceedings of the International Workshop on Harbor Maritime and Multimodal Logistics Modeling and Simulation.* Campora S. Giovanni, Italy.

Kühn W., 2006. *Digital factory - simulation enhancing the product and production engineering process.* In *Proceedings of the 2006 Winter Simulation Conference.* 1899–1906.

Limère V. and Van Landeghem H., 2009. *Cost model for parts supply in automotive industry.* In *Proceedings of the 16th European Concurrent Engineering Conference.* Eurosis, Bruges, Belgium, 120–125.

Martelli A., 2006. *Python in a Nuthsell.* O’Reilly, 2nd ed.

Siemens, 2008. *Tecnomatix Plant Simulation 8.2 Step-by-Step Help.* Siemens.