An Object-oriented Approach to ARGESIM Benchmark C14 'Supply Chain' using MATLAB

Matthias Wastian[1*], Stephan Reichl[2]

[1]dwh Simulation Services, Neustiftgasse 57-59, 1070, Vienna; *Matthias.Wastian@dwh.at*

[2]MMS- Mathematical Modelling and Simulation, Inst. of Analysis and Scientific Computing, TU Wien, Wiedner Hauptstraße 8-10, 1040 Vienna, Austria;

SNE 28(1), 2018, 35 - 38, DOI: 10.11128/sne.28.bn14.10407
Received: August 10, 2016; Revised August 15, 2017;
Accepted: December 10, 2017
SNE - Simulation Notes Europe, ARGESIM Publisher Vienna,
ISSN Print 2305-9974, Online 2306-0271, www.sne-journal.org

Abstract. ARGESIM Benchmark C14 'Supply Chain Management' allows different modelling approaches, from classical simulation approaches with discrete event systems or process modelling to directly programmed system evaluation. The tasks require a classical feedforward planning mechanism, without implicit feedback loops. An intrinsic property of the supply chain is a bi-directional flow, a material flow from factory via distributor to wholesaler, and an order flow from wholesaler via distributor to factory. Some simulation systems provide special modules for supply chain integrating these bidirectional flows, otherwise 'twin' modules must be defined, with reverse flow and intracommunication. Directly programmed modules for the flows may be an efficient approach, but the model flow descriptions are hard to read and not really suitable for understanding of the process. This solution tries an alternative approach using a fully object-oriented modelling approach for the modules fatory, distributor, and wholesaler – defined in MATLAB in order to make use of efficient vector and matrix structures.

1 Simulator

MATLAB – **MAT**rix **LAB**oratory – is a commercial platform independent software by *MathWorks Inc.* and used for solving mathematical problems, rather numerical calculations in the context of matrices. The syntax of MATLAB is adapted to common object oriented programming languages like JAVA or C#.

Hence, it is easy for programmers, who are used to common object oriented languages, to get familiar with the syntax.

MathWorks introduced class and method structures, combined with property definitions. However, MATLAB must not be mistaken with these programming languages. The underlying concept is different and designed for performing matrix calculations. The simulator program has to be implemented by capitalizing from these advantages.

2 Modelling

According to task assignments a supply chain management problem has to be simulated. There are three different types of active participants who are categorized to factories, distributors, and wholesalers.

While the number of the first two groups is fixed to four in each case, the number of wholesalers is summarized in a group. Figure 1 shows the simulation setup, to be mapped into object-oriented MATLAB structures.

Classes are defined for factories, distributors and wholesalers containing properties for matrices representing the stock as well as ordering status. The general structure is:

```
classdef Factory < handle
% FACTORY
   % - constructor
   % - order(factory object, and the curren order)
   % - produce (factory object and the current
             simulation tic)
   % - proof (factory object and current order)
   % - storage (factory object, MAT on
        finished products)
% getter/setter methods
```

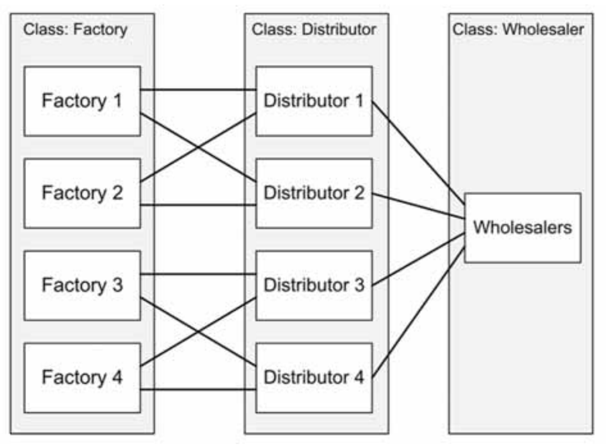

Figure 1. Simulation setup for chain supply with classes.

In general the simulation is tried to be implemented by considering the matrix theorem of MATLAB in order to get higher performance.

In order to boost the simulation speed, the simulation setup is utilized. It defines intervals of seconds which are multiples of 100. Hence these intervals are discretized to intervals of 100 seconds. This method reduces the simulation time by accepting slight changes in the randomized results which can be disregarded.

The basic supply methodology is the following. Factories produce day and night 12 types of products. Each factory produces only six products of the assortment. The choice for the produced product as well as the inter-arrival time at the distributor randomly distributed (uniform and exponential). In the simulation the inter-arrival time is calculated and after is passed the distributor's storage gets filled.

The distributors have specific factories assigned which they order goods from. The orders are placed at the beginning of the day. The group of wholesalers order from the distributors. The order time and the selected distributor are randomly chosen (uniformly distributed). The ordering mechanism is varied to three types `Simple`, `Demand`, `Order-Del`ay ordering mechanisms.

The production matrix defines the products which are produced by certain factories:

```
ProdMAT=
[[1,1,1,1,1,1,0,0,0,0,0,0]; [0,0,0,0,0,0,1,1,1,1,1,1];
 [0,0,0,1,1,1,1,1,1,0,0,0]; [1,1,1,0,0,0,0,0,0,1,1,1]];
```

The production matrix is allocated to a specific factory instance:

```
FA=Factory(ProdMAT(1,:), 'A');
```

The distributors instances are initialized with the certain run initial values like, time of distribution, the factory matrix where goods can be ordered and the ordering strategy:

```
DistributorA= Distributors([16,22,20,12],
                FactoryMAT, 'A', OSTRATEGY);
```

The wholesaler instance receive the distributor matrix on the distributors:

```
Wholesaler(DistributorMAT);
```

3 Results Task a – Simple Order Strategy

In Task a – Simple Order Strategy - the simple ordering mechanism is used in between the distributor and the factories after the preordering – 10 pieces per product. If the distributor sells at least one piece of a product he orders two pieces of the same product.

In Figure 2 the evolution of distributor 1's stock is shown during 30 days. On the seventh day the distributors start to order 10 pieces of every product which arrive until day 8. Hence, distributor 1 has a stock of 120 products. On day 9 the wholesalers start to buy products which result in a reduction of the stored products. This is marked by the red line in Figure 2. Afterwards the stock increases due to the ordering strategy.

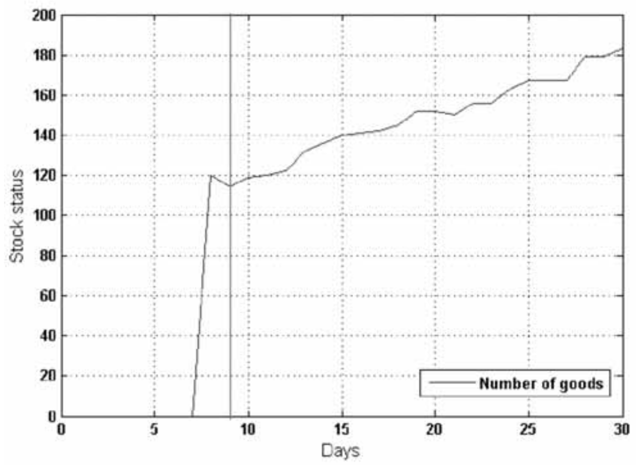

Figure 2. Stock status evolution for distributor 1 - Task a – Simple Order Strategy.

The following Matlab code above shows the procedural calls of the functions for this task:

```
for
 DAYSEC=1+((DAY1)*DAYLENGTH):1:DAYLENGTH*DAY
 factoryProduction(FA,FB,FC,FD,DAYSEC);
 factoryOrder(DistributorA,DistributorB,
         DistributorC, DistributorD,DAYSEC);
 wholesalerOrder(WholesalerG, DAYSEC);
end
```

First the factory production is triggered that starts the day production. Second, the distributors submit their order of the day. Last, the wholesalers send their orders. FA, FB, FC, and FD represent the product matrix for each factory.

The distributor orders a constant number of products without regarding the real requirements. Hence he does not sell as many products as he orders which results in an increasing stock status. As it is shown in Table 3 the simple ordering strategy is the most expensive one. Table 1 lists the maximum and minimum values of the total cost C, the number of delivered products N, and the relative costs R.

	C	N	R
max	12032	246	58.90
min	11141	199	45.24

Table 1. Min and max values of total costs C, delivered products N, and relative costs R for task a – simple order strategy.

The following MATLAB code above shows the orderStrategyA represented in source code:

```
function orderStrategyA(this)
   this.currentOrderMAT =
       logical(this.soldMAT).*2;
    this.currentOrderMAT =
      this.currentOrderMAT +
     (sum(this.nextDayOrderMAT,1));
   this.nextDayOrderMAT=zeros(1,12);
   this.soldMAT=zeros(1,12);
end
```

First the current order matrix is calculated. This is done on multiplying the number of sold products with two within the sold matrix. In addition the orders which were not able to be finished the day before are added. Afterwards the matrix is set to a zero matrix.

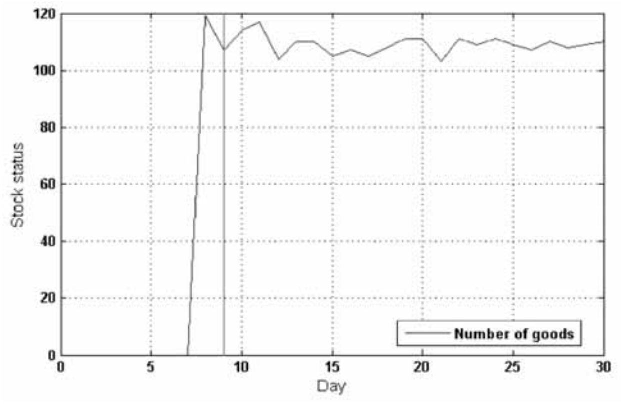

Figure 3. Stock status evolution for distributor 1 - Task b – On Demand Order Strategy.

4 Results Task b – On Demand Order Strategy

In Task b the on demand order strategy is used. The distributor order as many as needed pieces of products every day. Figure 3 shows the stock status over 30 days. Due to the ordering mechanism which is adapted to the actual wholesaler requirements the stock status remains constant.

The MATLAB code below shows the orderStrategyB represented in source code.

```
function orderStrategyB(this)
    this.currentOrderMAT = this.soldMAT;
    this.currentOrderMAT =
           this.currentOrderMAT +
           (sum(this.nextDayOrderMAT,1));
this.nextDayOrderMAT = zeros(1,12);
this.soldMAT=zeros(1,12);
end
```

First the current order matrix is calculated. Contrary to Task a, this is done on summing up the sold matrix with the matrix that contains the unfinished order from the day before.

Table 2 shows that the maximum of costs as well as the minimum of costs decreases in comparison with Task a. The number of sold products remains constant.

	C	N	R
max	11031	257	55.90
min	10582	195	41.80

Table 2. Min and max values of total costs C, delivered products N, and relative costs R for Task b – On Demand Order Strategy.

5 Results Task c – Minimal Supply Time - Strategy

The *Order Delay Strategy* in Task c implies the possibility for the distributors to order at the factory with the minimum of inter-arrival time.

Table 3 shows the minimum and maximum values of C, N, and R. Even the stock status is similar to the one in task b – see Figure 4 – and the distributors have the free choice of factories, the costs increase.

That is because of the charging of the costs of delivery which are independent from the number of delivered goods. Hence, the distributor pays the same amount of money regardless of the number of goods delivered.

In Task a and Task b the factories are fixed. One distributor can only order from to specific factories. In Task c it is possible that the distributor orders from more than two factories in one day. This results in increasing delivery costs and directly influences the total costs.

Below, MATLAB code is shown regarding the order strategy of Task c:

```
function orderStrategyC(this);
  calculateStorageCost(this);
  if max(this.currentOrderMAT(1,:))~=0
   contactFactories(this,tic);
end
```

The difference to strategy Task b is the contact of factories for ordering goods. Factories are contacted by getting the factory with the shortest supply time. Therefore the supply-time matrix is sorted:

```
sort(this.supplyTIME)
```

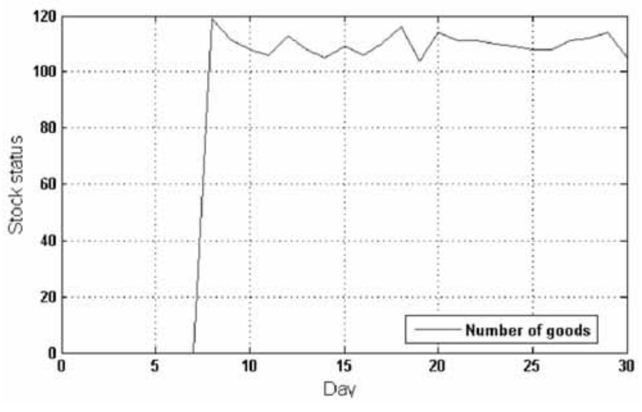

Figure 4. Stock status evolution for distributor 1 - Task c – Minimal Supply Time - Strategy.

	C	N	R
max	11031	257	55.90
min	10582	195	41.80

Table 3. Min and max values of total costs C, delivered products N, and relative costs R for Task c – Minimal Supply Time - Strategy.

6 Summary

The simulation is accomplished in MATLAB. The simulation deals with the advantages and disadvantages of the *Simple Order Strategy*, *On Demand Order Strategy*, and *Minimal Supply Time* **-** *Strategy*. These tasks imply a high number of matrices. Hence, the application MATLAB is of advantage as it shows high performance in matrix calculations.

Table 4 shows a comparison of the mean and deviation values of total cost C, the number of delivered products N, and the relative costs R between the three tasks. It is shown that the *On Demand Order Strategy* is the cheapest one. Contrary to the *Simple Order Strategy*, products are ordered on demand which results in lower stock costs. On the other hand the delivery costs are lower than those achieved with the *Minimal Supply Time - Strategy* as the number of factories are limited. Even the delivery costs are limited in Task c by choosing the factory with the lowest delivery time. Hence the total costs get higher than in Task b.

	Task a Mean/ St.Dev	Task b Mean/ St.Dev	Task c Mean/ St.Dev
C	11357 / 1634.5	10482 / 1949	11983 / 2248
N	220.9 / 33.6	218.6 /43.2	217.9 /44.1
R	51.6 / 2.86	48.1 / 3.16	55.3 / 4.4

Table 4. Comparison between order strategies in the three tasks.