An Approach for Dynamic Supply Chain Modelling

J. Manuel Feliz Teixeira
António E. S. Carvalho Brito

September 2003

GEIN – Section of Management and Industrial Engineering
Faculty of Engineering of the University of Porto, PORTUGAL
Emails: feliz@fe.up.pt; acbrito@fe.up.pt

KEYWORDS: Supply Chain, model, simulation, operational policies, costs, inventory, transports, delivery.

ABSTRACT

This paper presents some concepts related with the development of a "model generator" for simulating supply chain systems. Such concepts arise not only from the object point of view taken over this system, but also from our intention to synthesise the behaviour of such kind of systems in a more general form, in particular concerning the events and activities related with *suppliers*, *factories*, *warehouses*, *retailers*, and even the last *customers*, which in fact trigger the flow of materials and information on the chain.

Unlike the static approaches usually used for strategic purposes, where parameters like the time of delivery or the average rate of material flow are held as inputs to the system, the ideas described here consider a dynamic representation of the supply chain in which those parameters are *results* of some detailed simulation process, what gives this approach the ability of modelling those systems starting from the less abstract point of view to a more abstract representation. The result is a more realistic picture of the dynamics involved.

Due to this fact, this approach could be seen as being more directed to managers than to strategists. Nevertheless, the objective is also that the models can be used to simulate either short or long periods of time, revealing their usefulness also for strategic analysis.

The basis of these ideas is to consider the flow of products, information and money between any elements in the chain as a general form of *customer-supplier* exchange activity, and also by treating each of those elements as inheriting from a single element which includes the basic behaviour (and resources) of a *factory*, *warehouse* and *retailer*. A description of such element will be made, explaining its structure and the associated *fixed* and *variable* costs of its various processes.

As we will see, many parameters considered inputs to other modelling techniques will appear here as outputs, giving the analyst more interesting data with which it can measure the supply chain performance by means of any statistical methods.

1. Introduction

During last decades, most of the simulation approaches used to model the supply chain have been deeply related with strategic intents, that is, were mainly devoted to study the problem of dimensioning and geographically positioning the chain components in a way the overall expected costs could be calculated and, maybe, optimised. The objectives of such approaches were mainly strategic, and so, dealing with average values over long periods of time. In a certain way, the intent of those models was to turn the results independent of *time* variable, and so, what one can retrieve from them is also a static expectation. Although these methods still are very important on the first step of a chain establishment on the ground, they cannot give confident results that could help to manage the supply chain, that is, they hardly would survive to the need of characterizing the chain dynamics, or optimising week-by-week the processes developing in it. To try to resolve this kind of issues, the supply chain started to be also modelled in a dynamic way, and the approach presented here must be considered one more perspective to enrich such trials.

In the point of view of this approach, the supply chain is already established on the ground and the problems to be studied or analysed must be related with the optimisation of the actual processes running in the chain. These processes are mainly the *purchasing*, the *stocking*, the *manufacturing*, the *delivering* and the *managing*; all of them will be measured using not only technical but also economical indexes. This last aspect will be included as a way to measure the economical performance of each element, being also very useful when the chain contains elements belonging to different enterprises, that is, when there is concurrence between different management systems implanted on the same ground. By simply representing the flow of money, one can visualise and easily calculate the *costs* and the *income* for a particular group of elements on the chain, and so, also for those one considers its own part of the system...

As showed in figure 1, in general the supply chain is considered a kind of network connecting *factories (F)*, *warehouses (W)* and *retailers (R)*, with these elements treated as having different structures and different behaviours on the

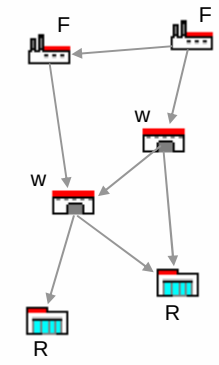

Fig. 1 General concept of a supply chain

processes running in the chain. In the *factory*, for example, there is a transformation process that gives the row materials the aspect of products, while in a *warehouse* there is only storage and handling processes and in the *retailer* only the action of sales. Finally, what flow in the net are products, transported by vehicles.

Here, however, elements like *factories*, *warehouses* and *retailers* are considered inheriting from a single element that will be named *customer-supplier-unit* (CSU), with which will even be possible to model the figures of the *supplier* and *last customer*. Thus, the overall chain is seen as a group of CSUs connected together by a net of transports, where there also can be exchange of information (fig. 2). Now, what will distinguish a *factory* from a *warehouse* or from a *retailer*, or any other element, will only be how that particular element uses the resources of its CSU and the kind of materials it handles as input and output.

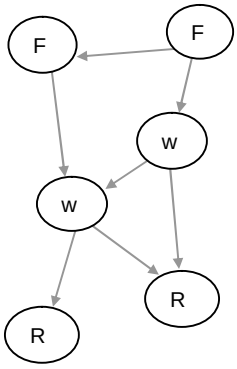

Fig. 2 CSU equivalent of a general supply chain

Each CSU lives from a constant exchange activity with its neighbours, receiving material and information and giving back money, delivering material and information and receiving money. Thus, in order to see how the economy develops between the elements on the chain, it will be included in the basic structure of the CSU a reference to its own *economic account*. One must not forget most of the decisions are taken not only based on the technical performance of the systems but also on how the money spreads along those systems… This account, of course, must include *costs* and *incomes*, in a way that with the results of the simulation one can better predict how each of these elements is contributing to the overall economical performance of the chain.

2. The *customer-supplier-unit* (CSU)

The CSU is then a basic element who owns a group of *customers*, to whom it delivers material and from whom it receives money, and a group of *suppliers*, from whom it receives material and to whom it returns money. Anyhow, to be able to handle this basic functionality, the CSU must also have in its structure some more general resources. Basically, these resources will be a *stock*, an *economic account*, an *input queue* where the *supplier* vehicles will wait to deliver the materials, and an *output queue* where other vehicles will line up to receive the material that must be delivered to the *customers*.

In order to be able to manage these resources, the CSU must also include in its structure a *list of suppliers*, a *list of customers*, a *list of products* and, finally, a *list of vehicles* representing its own *fleet*. In this approach we established as a principle that the *fleet* always belongs to the one who delivers, what seems a reasonable approach for most practical cases, even when the fleet is hired, as we will see later.

Considering this structure, which is depicted in figure 3, the CSU becomes a general element ready to be linked to other elements of its kind, making possible the overall supply chain representation. Of course, different CSUs can have different amounts of stock, different number of customers and (or) suppliers, different products, and so on, depending on the role

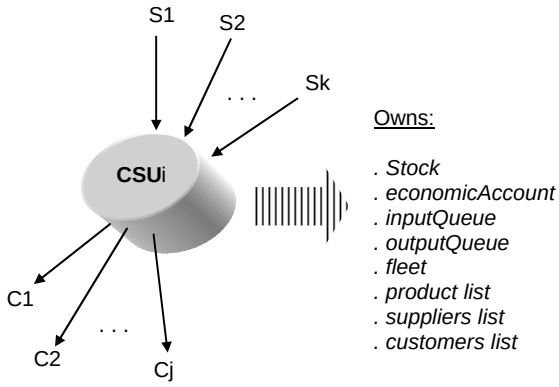

Fig. 3 Generalised CSU concept having *k* suppliers and *j* customers

they play in the chain. For instance, ultimate customers can be thought as having null stocks, null fleets and null customer list, what will reduce them to a source of money for the chain and a demand of products. On the other hand, a supplier source can be seen as a CSU having infinite stock and null supplier list. Note also that raw materials can be treated as products, even if later will be necessary to adapt the unity of transport to the proper case. Thus, this approach opens the representation also to *last customers* and *suppliers*, which in most of the simulators are treated as simple statistical parameters.

3. Connecting the CSUs and the flow of materials

To be able to model the chain dynamics is now necessary to connect each CSU to its effective neighbours. A net of *paths* in which vehicles will move ensures this. These *paths* can include normal roads, highways, railways, boat circuits, and even airlines. However, for didactic purposes, let us assume general-purpose ground vehicles as presented in figure 4.

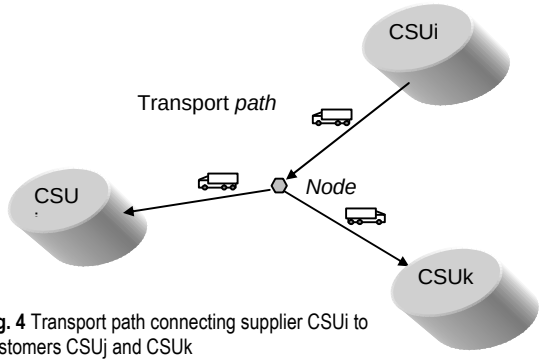

Fig. 4 Transport path connecting supplier CSUi to customers CSUj and CSUk

Note that each *segment* of a *path* can then represent different costs to the delivery process; for instance, vehicles can first run on a highway and then change to normal roads. That, and the probable existence of real road junctions in practice, implies also the usage of the concept of *node* (fig. 4). Therefore, vehicles will travel from *node* to *node* with the objective of executing a certain predefined *job*. This also means each CSU must include the dimension of a *node*. Of course, as the vehicle moves from one *node* to the next *node* the delivery costs are expected to rise, as they are calculated based on the known costs of the vehicle utilization per kilometre and driver cost per day.

Note also that, in principle, time spent by the vehicles in these *paths* will not be predictable, also because different vehicles from different suppliers can reach the customer around the same time, and then it will be necessary to queue before the effective delivery will take place. Situations like this will be, of course, dependent

on the layout of the chain, on the quantity of suppliers serving each customer and on the intensity of delivering flow in the net, as well as on the delivering policy and the size of vehicles.

Each vehicle is treated as an independent entity, and then the movement of a vehicle trough these *paths* will be simply dependent on the vehicle itself, that is, it will result of the execution of a sequence of events that are "property" of the kind of vehicle. Thus, from a high-level point of view, there will be only two commands (events) each CSU can send to its vehicles: *e_startDelivery()* and *e_startReturn()*. The first sent after attributing a certain *job* to the vehicle and the later when a *job* is finished. The vehicle itself must then be capable of delivering the material without any other interference from its CSU, even if a *job* includes more than one delivery point, as in the case of local delivery.

4. Delivering speed

After the material is ordered to a particular CSU, it will start a sequence of actions on that CSU that will lead to the *processing of the order*, the *material packing* and finally the start of the *delivery process*. From the point of view of the customer, the speed of the delivery will also depend on the time spent on such sequence of actions and on the criteria used by the supplier CSU to assign *jobs* to vehicles. Over this, however, it must be added the time needed to carry the material along the *paths* between the CSUs, which is dependent not only on the *vehicle speed*, but also on the *limit speed* of each *segment* on the *path* trough where the vehicle must travel. The figure 5 helps to explain better these ideas, representing a *path* with 3 segments and 4 *nodes* connecting supplier CSUi to customer CSUj.

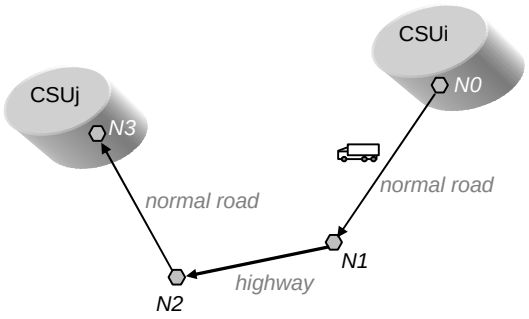

Fig. 5 Path connecting supplier CSUi to customer CSUj including normal roads and highway

If one assigns a *limit speed* to each *path segment* and a certain *vehicle speed* to each *vehicle*, then the effective speed of transport in that particular *segment* can be calculated as:

transportSpeed = Min(segmentSpeed, vehicleSpeed)

Thus, each time the *vehicle* reaches a new *node* leading to a new *segment*, it must recalculate its next speed of transport. As it is easily understandable, the *limit speed* assigned to a *segment* can serve to model not only the legal limit speed of that road but also the usual conditions of traffic flow on it.

In order to model other more specific aspects related with the delivery process, nodes will be of four different types: *normal*, if it is a node where the vehicles just change to a new path segment; *CSU*, when the node refers to a CSU; *transfer*, when the node is used to transfer material from one vehicle to another; and *pause*, when it represents a certain delay in the delivering process. This, of course, can also contribute to the mean speed of delivery observed between CSUs.

5. Costs

Once the supply chain is seen as a dynamic system, the costs associated with it must also result from a dynamic process, that is, they will be calculated and affected to the respective CSUs by means of certain event executions in time. Thus, it is important not only to know the amount of a particular cost but also the time when it must be affected to the CSU. This will also allow a continuous calculation of *costs* and *incomes* during the simulation process, for example, important economic indicators for the supply chain manager.

Based on the idea that each process in a CSU must have its costs, and considering as main processes the *purchasing*, the *stocking*, the *manufacturing*, the *delivering* and also the *management*, as depicted in the figure 6, it was decided to specify the costs following this same kind of taxonomy.

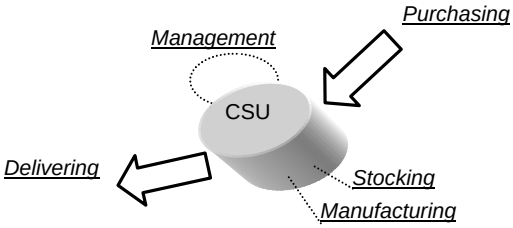

Fig. 6 Main processes for assigning costs to a CSU

We will talk about *purchasing costs*, *stocking costs*, *manufacturing costs*, *delivering costs* and *management costs*. Each of these costs can be thought as the result of other more specific costs, some of them assumed *fixed* and others *variable*, anyhow, it is important to bear in mind the more the costs are specified the more events one need to add to the model. Here we will only be interested on those costs usually considered more relevant, and each of them handled as a *fixed* and a *variable* component. The *fixed* will be considered the price to pay for "existing", while the *variable* will represent the price to pay for "operating" or "processing".

5.1 Purchasing costs

These costs must be affected to the CSU at the end of each *purchasing cycle*, that is, when products (or materials) are received from (and paid to) the supplier. The amount of these costs depends on the *price* previously established for the products by the supplier, as well as on the *administration* involved in the process. This last aspect is modelled as a cost per purchasing cycle, and contains the idea of the "*setup cost*" usually referred in *inventory management* literature. So, the purchasing cost will occur in the end of each *purchase cycle* and its amount will be given by:

$$PurchasingCost(t) = ProductsPrice(t) + administration(t)$$

Where *administration* can refer to invoice prepare, papers, phone-calls, etc., and also to personnel costs. Purchasing costs are considered here only as *variable* costs.

5.2 Stocking costs

The process of stocking includes many aspects, and so, many kinds of costs can be assigned to it. However, the most usually considered costs[1] are *operations*, which include receiving, order picking, replenishment, etc.; *administration*, including order processing, managing accounts, personnel, etc.; *occupancy*, which includes rents, services, insurance, depreciation, maintenance; and *inventory*, or *holding* costs, based on the existence of products in stock, usually 1% above the bank base lending rate. The overall stocking costs can then be expressed as:

$$StockingCost = [operations(t) + administration(t)] + occupancy + holding\ cost(t)$$

To turn the simulation process simpler, we will call *stock processing* cost to the *operations* and *administration* costs together, which are considered per unit of material handled. Then, the *occupancy* cost is based on a fixed cost per unit time (a fixed cost). And finally, the *holding* cost is based on the price of the money invested on the products in stock.

Stock processing costs and *holding* costs will be affected to the CSU at the end of each action of input or output on the stock (variable costs), while o*ccupation* costs will be affected continuously, as the time increases (fixed cost). It is important to notice that the *holding* costs begin when the CSU pays the product to the supplier and will only end when the CSU receives the payment for delivering it to the customer, thus also *in-transit* inventories are considered.

In general, the *holding* cost for the i^{th} product in the stock will exhibit the following dependence on time:

$$HoldingCost(t)i = (Cp)i \times (Np)i \times (HC)i \times (t - to)$$

Where:

$(Cp)i = i^{th}$ product unit cost.
$(Np)i$ = Quantity of products i in the stock.
$(HC)i$ = holding cost rate per unit time (bank lending rate)
to = Instant of time when the product was purchased
t = present time

There are also costs associated with *stock breaking*[2], each time the demand cannot be satisfied from the stock. These costs will be estimated by the quantity of material that have not been served, thus generating a *stockout*, as well as by the time spent to serve the customer compared with the time the customer accepts to wait for the materials to arrive, leading to a measurement of a "*customer satisfaction index*".

5.3 Manufacturing costs

In the present approach the factory is considered to be similar to the warehouse, with the differences that it will have two kinds (*R* and *P*) of products in stock (R for raw materials, P for finish products) and also a *process* named *manufacturing*, which will be responsible for the transformation of *R* products into *P* products. Basically, the manufactory is then seen as a warehouse having a frequent rearrangement of products in its inventory. Such a rearrangement is imposed by the *manufacturing* process.

As the figure 7 suggests, the costs associated with the process of *manufacturing* will mainly be the costs of *transforming* *R* products into *P* products (variable cost), the costs of *labour* (assumed as a fixed cost), and the costs of space *occupancy* (fixed cost) for the production zone. The costs associated with *stocking*

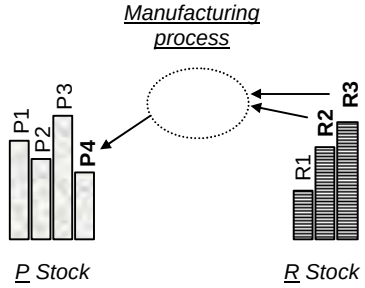

Fig. 7 Equivalent of a manufacturing process, in this case products *R2* and *R3* will give product *P4*.

will be the same referred in the previous section. The *transforming* costs must be specified for each product *P* in the factory, and given per unit of product. Of course, these costs must be affected to the CSU at the end of each *production cycle*.

Then, after a certain period of time *t*, the contribution of the *manufacturing* process for the overall *variable costs* of the CSU will be given by:

$$ManufacturingCost(t) = \sum_i [transforming(t)]_i$$

Where *i* referes to the *ith* product manufactured in that same period of time.

Finally, the *occupancy* costs related with the factory's zone, as well as the *labour* costs, will contribute to the global *fixed costs* of the CSU.

5.4 Delivering costs

Delivering costs are here considered as the result of two main contributions: *occupancy*, related with space reserved for the fleet, garages, or even rents, if the fleet doesn't belong to the CSU; and *transport*, which include the costs associated to drivers, vehicle consume of fuel and oils, paid roads, meals, delays, handling the materials, and so on.

Similarly to what was considered in the *StockingCost*, the *occupancy* cost is seen as a fixed cost, and given per unit time. The *transport* costs, however, will be computed during the simulation process each time a *vehicle* reaches a new *node* in the transport path network, and assigned to the respective CSU when the material reaches its destiny. Thus,

$$DeliveryCost = occupancy + transport(t)$$

The *transport* costs include the following main aspects: *fuel*, *lubricants*, *maintenance* and *roads* as costs depending on the distance travelled (ds); *drivers*, depending on the time travelled (dt); and *meals* and *delays*, as constant costs. As an example, if a vehicle is going from *node A* to *node B* with a speed *v*, the following *transport* costs will be added to its CSU when reached the *node B*:

$$Fuel(ds) = (VehicleFuelConsume/PricePerLitre) * ds$$
$$Roads(ds) = (PriceOfRoadPerKilometre) * ds$$
$$Driver(dt) = (DriverPricePerUnitTime) * (ds/v)$$

Where *lubricants* and *maintenance* can be considered included in the *fuel's* price. Notice that when the node corresponds to a node of resting or meal, also the constant costs of resting or meal must be added. Also notice that hiring the services of an external fleet can be modelled as keeping the cost of occupancy null.

5.5 Management costs

The *Management* costs, related with the costs of managers, analysts, secretaries, strategists, marketing, etc., will be considered here as *fixed* costs, and, as a first approach, they must be included in the fixed costs of the main resource of the CSU, that is, the inventory. These costs are also affected to the CSU as a continuously growing cost.

5.6 Total and global costs calculation

Each of the previous *variable costs* are recorded during the simulation as *instantaneous* values, thus, the calculation of their *totals* will automatically result from the simply sumation of those values along the time domain. For instance, considering the output transport costs the function $g(t_i)$, which would have the form of a data output record from the simulation, the *total variable costs of transport* would simply be given by:

$$total = \sum_i g(t_i)$$

This is a general method used in this modelling approach to record the behaviour of the interesting variables during the simulation run, which simplifies future calculations. For example, it will turn the calculation of *global costs* extremely simple, not only in a particular CSU but even in the entire chain network.

6. Information and the flow of money

Other things flowing in the chain are *information* and *money*, which in the present approach we consider associated with instantaneous operations. At the moment, the *information* will mainly be related with the communications between CSUs during material ordering, but we expect to include more about that in the near future, mainly the transference of the clients demand to the CSU's suppliers, in order to transfer the demand upstream on the chain, a procedure claimed to improve the partner's flexibility.

But let's now concentrate on the *money* flow, related with the payment to the supplier in the moment the material reaches its destiny. We make the following assumption: money always flow in the opposite direction of the material flow, that is, each time one receives materials one has to pay back its price to the supplier. It is based on this process of exchange the chain will maintain its activity along the time. The payment is here considered to be instantaneous.

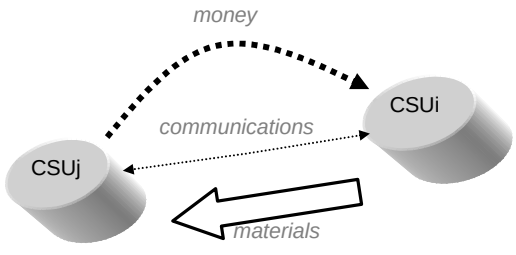

Fig. 8 Exchanging process between supplier CSUi and customer CSUj

The introduction of this economic aspect will give the analyst the possibility of studying the evolution of the chain parameters along the time not only based on technical performances but also taking into account economical aspects, as in fact it happens in reality.

Money flow will be triggered each time a customer receives material from a supplier. As the chain continues to work, such flow will spread to all the elements in the network, then allowing the instantaneous calculation of the *variable costs, incomes, total costs, etc.*, as well as certain important economical index, as the *cash flow*, for instance.

7. Modelling the general CSU activity

As we have seen, in each CSU some processes are running and interfacing with each other to create the complex activity of the unit. Those processes are the *purchasing*, the *stocking*, the *production*, the *delivery* and the *management*. Except the *management*, which is here considered instantaneous and coincident with any events in the CSU, each of the other processes contribute to the overall CSU's state with a certain number of states (we call *live states* those states which duration can previously be known or computed, and *dead states* those states where this cannot be done, as in waiting states and queues, for instance)[3]. The collection of such states, and the possible connections between them, will form the *state diagram* of the CSU, the real basis for the model. Therefore, first we have specified the *state diagram* of the CSU, and then converted it into a sequence of events which finally have been translated to a programming language code, in our case the C++.

Notice that, while building the supply chain network in the simulator, the user will no longer need to think on those internal states of the CSU, as they become endogenous to the CSU object, turning the supply chain representation more likely a game of objects connected between each other by paths of information and material flow. Thus, at the user's point of view, the paradigm used in the simulation is a *real object paradigm* and not any of the usual basic paradigms described in simulation literature like Petri-nets, activities, events, processes, 3phase, etc.

7.1 The purchasing process

This process is the process of ordering material to suppliers, and starts in each of the conditions: (1) The present stock control policy determines it is time to reorder. (2) There is an order from a client that cannot be fulfilled direct from stock and, in the case of accepting *backorders*, new material can be ordered.

After creating and preparing the new order with the material requirements, each of these "procedures" will then trigger the start of the purchasing process, which in fact reduces to two "events": (1) The sending off the new order to the respective supplier, we named *e_order()*. (2) The receiving of the material arrived, we named *e_arrive()*. Between theses events the process will be in a *dead state*, waiting the material to arrive from the supplier.

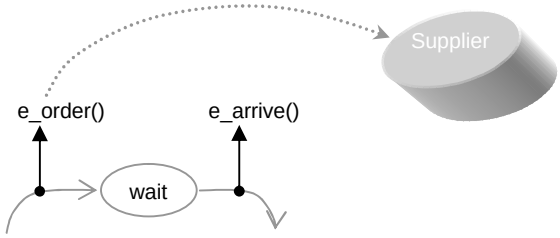

Fig. 9 The basic purchase process.

Notice that *lead times* are initially unknown, as they result from other dynamics internal to the CSU supplier and on the transport speed and availability, for example. Also notice *e_arrive()* can be thought as the beginning of a subsequent *live state* which in fact represents the UNLOAD activity. This will let us later substitute the event *e_arrive()* by the event *e_startUnload()*.

7.2 The stocking process

We considered the *stocking process* the gathering of two distinct processes: the *input process* and the *output process*. As their names state, the former is responsible for inputting material to the stock, and the later for the output of material from the stock. Figure 10 gives an idea of the events and activities involved in the *output process*:

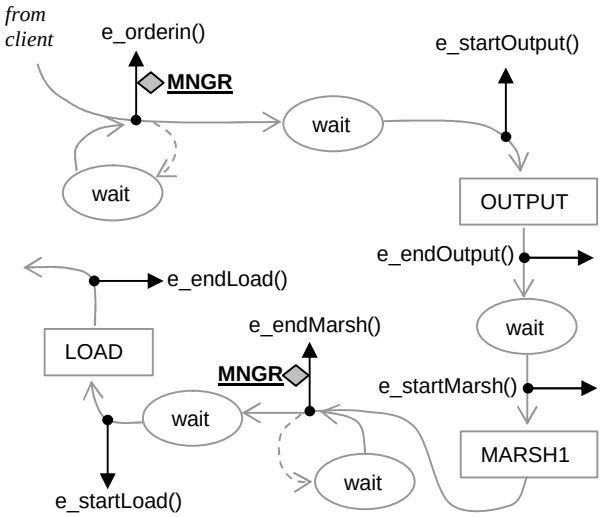

Fig. 10 State diagram of the stocking output process.

Basically, an order-for-output from a client enters the CSU by means of an *e_orderin()* event. This event will then ask permission to an upper-level "*manager*" for the satisfaction of such an order. The "*manager*" will decide if the order will be immediately served, rejected as *stockout*, or made to wait to be satisfied later when new material will arrive from the supplier. As long as the order can be served, the process enters in the phase of outputting the material from the stock. Then the material will be prepared and packed at the MARSH1 activity, and the order becomes ready to be sent to the client. Anyhow, the "*manager*" called by the *e_endMarsh()* event will decide whether or not the order wait for later LOAD and transportation, keeping it in a different waiting place depending on such a decision. The event *e_endLoad()* will trigger the *delivering process*.

The *input process* is basically the inverse of this process. However, in the present approach the *input process* is considered reduced to the activity of UNLOAD, which is due when new material arrives in a vehicle from a supplier. This is the same as considering the material will be available in the stock precisely at the start of the UNLOAD, which great simplifies the modelling. In fact, as not-yet-fulfilled orders will have to wait in the queue before OUTPUT activity, this approach is expected not to lead to significant discrepancies in the results, as it is as if the *input process* would, in the worst case, run in parallel with the *output process* and then, in average the material will already be in stock when it must be taken out. This consideration was mainly used to simplify instantaneous holding costs calculation during the simulation process.

7.3 The delivering process

This process is responsible for carrying the material to the clients. As it was referred earlier in this paper, this can be done by many different ways, using from normal roads to airlines, and thus the efficiency of this process will be great dependent on the kind of transport it assumes. Although the delivery process is conceptually very simple, as it is presented in figure 11, it can turn much more complex in certain cases, for example, when using paths of *transfer* vehicles between two nodes or when the delivery paths are complex. Anyhow, the vehicle entities will be responsible themselves to handle all the material while they move along the network, what will make them practically transparent to the delivery process as it is defined in the present figure.

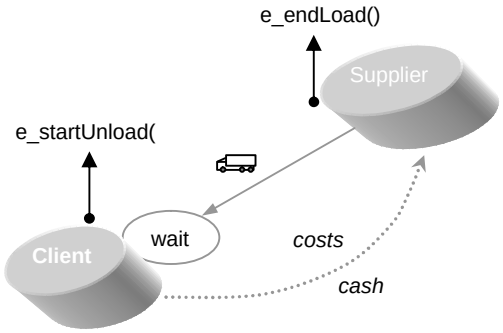

Fig. 11 The basic delivering process.

Thus, one considers that this process begins with the event *e_startDelivery()*, which is equivalent to *e_endLoad()*, and will end in the moment the vehicle will be received by the CSU client at the event *e_startUnload()*. In the case of more than one vehicle on delivering to the same client, probably there will be the need to wait for a free *inputBay*, as figure suggests. With this model only during or after the simulation one will be able to estimate the *lead times* of the system. Unlike certain Supply Chain Simulators, here *lead times* become outputs from the simulation, and not inputs. Nevertheless, the time spent on the return of the vehicle to its home is estimated, considered to be 75% of the time consumed to reach the client. At the end of this process costs and other parameters are computed, and the respective *cash* made to flow to the right suppliers.

7.4 The manufacturing process

The *manufacturing process* is modelled here assuming no product mixing and only one line of production for each product manufactured. This, of course, can be made more complex in future, but at the moment it will be treated in this simplified form, as our main concern is to focus the attention in the overall supply chain behaviour and not particularly in production. In a wide range of real cases, however, this model can be used without affecting significantly the results. The state diagram of such model is presented in the next figure.

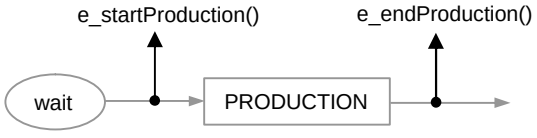

Fig. 12 The basic manufacturing process.

There is a waiting state (queue) for the orders waiting to enter the production, and the events *e_startProduction()* and *e_endProduction()* signalling the start and the end of the PRODUCTION activity. This model also uses the classical concepts of *production rate*, which establishes the maximum frequency the orders can be made to enter the PRODUCTION activity, and the *raw cycle time*, what defines the time spent by

each lot in that same activity.

The start of the *manufacturing process* will be triggered or by the *stock level* of the product to be produced, or using a *cycle* as in the case of continuous production. This will be established by the user. At the proper time, the necessary products and the respective amounts are removed from the stock (using the *output process*) so that the new product can be produced by the *manufacturing process*. At the end, the new product will be added to the stock of the CSU and the related costs computed.

8. About the operational policies in the CSU

The overall activity of the CSU is also dependent on the operating policies adopted, being the most decisive of them the *inventory policy* (how to order the materials), the *production policy* (how to produce), and the *delivery policy* (how to bring the materials to the customers). In this section we briefly present some considerations about the policies implemented in the scope of this modelling approach. To handle such tasks each CSU C++ object includes a *CSUManager()* method to concentrate the code associated with these policies.

The *delivery policy*, for instance, belonging to the FLEET-MANAGER section of the *CSUManager()*, is in this first version very simple: each time the material must be carried to the customer, a vehicle, if any available and free, is loaded with the material and sent to the customer. Otherwise, it will wait. There is not yet *multi-drop* delivery implemented, although different products can integrate the same order.

Concerning the *production policy*, it is given to the user the following three options of managing: ON-DEMAND, if the production is only to start when there is an order from the client which cannot be satisfied directly from the actual stock. BY-LEVEL, if the production is triggered by the level of the stock. BY-CYCLE, when the production is to be continuous and restarted cycle by cycle.

Finally, in the *inventory policies* one can already use a wider range of management options, which go from the simpler (r, Q) and (r, S) models till the notions of *Safety Stock*, KANBAN, Economical Order Quantity (EOQ), and a more general (r(t), Q(t)) model which is expected to try to adjust *r* and *Q* to the demand pattern along the time. In fact, the actual main core of the CSU management is this set of policies, which can be used by the analyst to test different management strategies and other concepts within the model. More about this issue will be presented in a paper to come.

9. RESULTS

The main results of this approach is a software application focused on *Supply Chain Simulation* with which is possible to compare different scenarios in terms of some common Supply Chain Metrics. With such a simulator one can also test operational policies in response to different patterns of demand, in order to have an idea on how fast the chain is able to adapt to demand changes, thus getting an indirect and qualitative measure of its flexibility, as well as of the parameters that can interfere with such flexibility. In fact, this approach also includes a first trial to measure and quantify FLEXIBILITY, or AGILITY, although we only expect to refer to such a subject in a future presentation.

At this stage, we will finish by presenting a simulation of a relatively simple model which is usually used by the *University of Cranfield* (UK) as a didactic supply chain management exercise[4], considering only one product in the chain and all the delivery times set to be one cycle.

Figure 13 shows the network structure used on this exercise, named "*Cranfield Blocks Game*" (Richard Saw 2002). Usually, this game is played with seven groups of students, each group being responsible for the management of a single facility. The customers demand (*Cj*) is random generated, and the supplier (up on the figure) is considered an infinite source of materials. Thus, the groups are only endorsed to manage from the *Depot1* to the *Factory*. No backorders are considered, and each facility has to

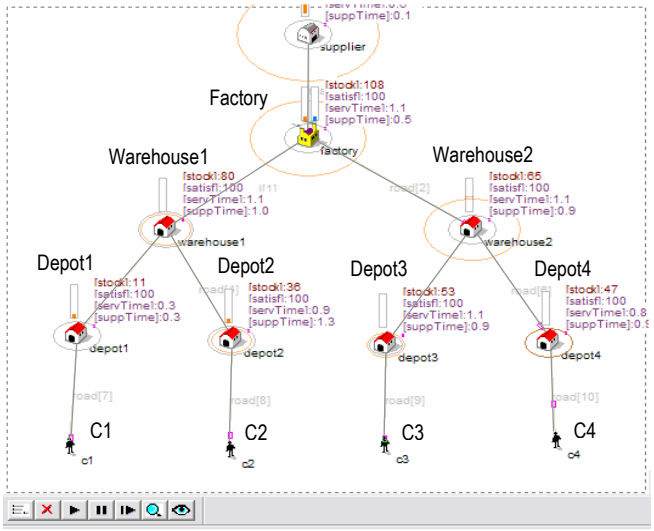

Fig. 13 Cranfield Blocks Game, being simulated.

adjust its reorder quantity to a multiple of a certain *Base Quantity* (BQ) which increases as the facility approaches the *Factory*. All the activity in the chain begin with the customers demand (*Cj*), and the goal of each facility manager is to fulfil demand keeping the *stock* as low as possible without incur in *stockouts*. The average level of costumers demand increases from left to right in the chain, meaning *C1* < *C2* < *C3* < *C4*.

As merely didactic results we present now some output data achieved with the simulation of this case, using in each facility the (r, Q) inventory control model.

The first chart (Fig. 14) shows in different colours the DEPOT3 *local* and *in-motion* stock levels along the time, as well as the demand, the quantity of material arrived from the supplier and also the quantities of the stockouts.

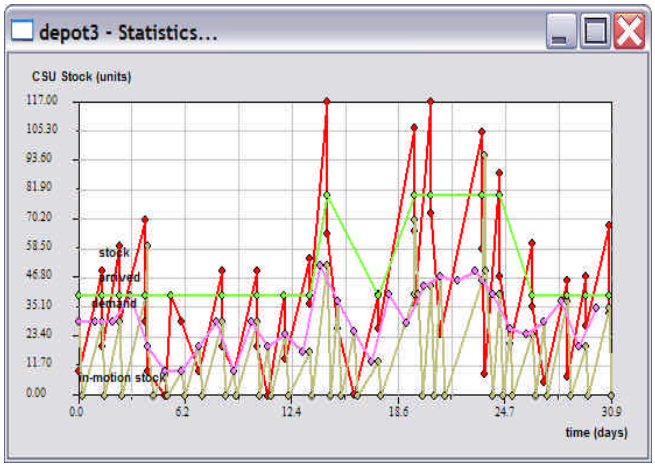

Fig. 14 Depot3 stock levels versus time.

In the next figure (Fig. 15) is plotted how the WAREHOUSE2 *variable costs* were running with the time. It shoes that was

observed a certain stability in the purchasing costs, in the delivery costs and in the holding costs, comparing with higher amplitude variations observed in the stocking costs due to the input and output of materials from the stock. However, these results have been obtained considering default values in the stocking process, what gives them no special significance.

Accumulated costs are also computed during the simulation run, letting the user have an idea of the total amounts involved in the CSU activity during the simulated period of time, and can be observed in another sort of chart, not presented here.

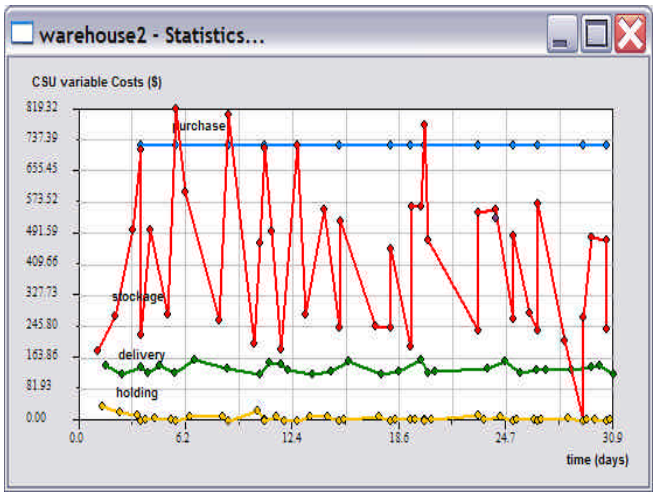

Fig. 15 Warehouse2 variable costs versus time.

The next figure (Fig. 16) relates with WAREHOUSE1 and shows the variations of the supplier's lead time compared with the average time to serve clients. It also shows the delivery time, here considered to be equivalent to the transport time. Based on this information the analyst can better estimate how much the behaviour of such times can affect the performance of the CSU, as well as to project more reliable inventory policies that can better adapt to its suppliers response times. Based on this data the analyst also can understand, for instance, why a certain customer is not being satisfied, whether it is due to its delivery policy or due to its suppliers lead time pattern.

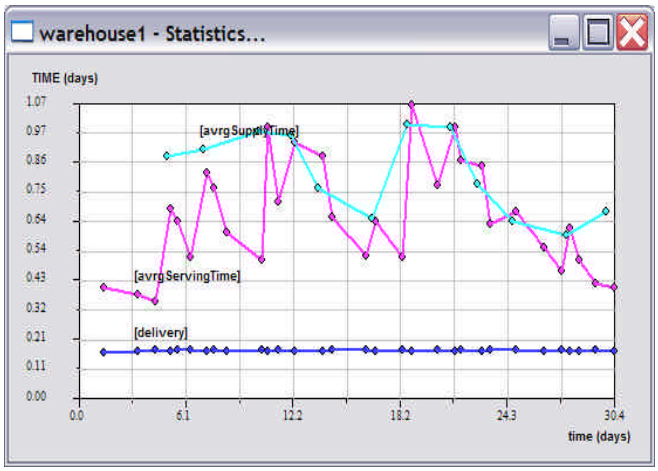

Fig. 16 Some Warehouse1 times versus time.

Finally, figure 17 shows some measures used as Supply Chain Metrics, in this case associated with the WAREHOUSE1 performance computed after 5 simulation runs. On this char is represented the estimated average values and also the variability of measures like TotalCosts, Incomes, Turnover, ServiceLevel, StockoutRatio, etc., thus giving the user the final ability to evaluate different policies implemented in the CSUs, as well as the capability to compare the performances of the different partners in the chain.

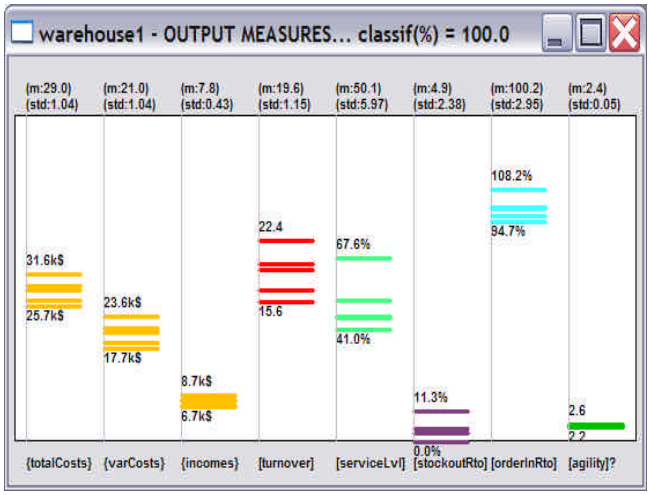

Fig. 17 Warehouse1 output measures after 5 runs.

References:

[1] John E. Sussams, "*Logistics Modelling*", Pitman Publishing, 1992, p. 55
[2] José Fernando Gonçalves, "*Gestão de Aprovisionamentos*", Edição revista, Publindústria, 1999
[3] António E. S. Carvalho Brito, J. Manuel Feliz Teixeira, "*Simulação por Computador*", Editora Publindústria, Junho 2001
[4] Richard Saw, "*Cranfield Blocks Game*", private communication, Centre for Logistics and Supply Chain Management (CSCM), University of Cranfield, 2002