Spreadsheet vs. multiagent-based simulations in the study of decision making in supply chains

T. Moyaux*
DAMAS, FOR@C & CIRRELT,
Département d'Informatique et de Génie Logiciel, Université Laval
Quebec City G1K 7P4 (Quebec, Canada)
E-mail: Thierry.Moyaux@insa-lyon.fr
*Corresponding author

B. Chaib-draa
DAMAS & FOR@C,
Département d'Informatique et de Génie Logiciel, Université Laval
Quebec City G1K 7P4 (Quebec, Canada)
E-mail: Brahim.Chaib-draa@ift.ulaval.ca

S. D'Amours
FOR@C & CIRRELT ,
Département de Génie Mécanique, Université Laval
Quebec City G1K 7P4 (Quebec, Canada)
E-mail: Sophie.DAmours@forac.ulaval.ca

Abstract: A game called the Quebec Wood Supply Game (QWSG) is a role-playing simulation based on the Beer Game for teaching supply chain dynamics, and, in particular, the bullwhip effect. In this context, this paper describes and compares two simulators based on the QWSG which may be used to study decision making and its impact on supply chain dynamics. We first focus on the direct implementation of the QWSG in a spreadsheet program. This spreadsheet model is the base on which we next build a more complex MultiAgent Based Simulation (MABS) in which JACK™ agents represent companies. Finally, we compare the respective advantages of each simulator. We identify the features of a supply chain model making a spreadsheet simulation impossible, and those for which a spreadsheet simulation is better, as good as, or worse than MABS.

Keywords: supply chain management; comparison of simulation types; spreadsheet simulation; multiagent based simulation; Beer Game; Quebec Wood Supply Game.

Reference to this paper should be made as follows: XXXX, XXXX and XXXX (2008) 'Spreadsheet vs. multiagent-based simulations in the study of decision making in supply chains', Int. J. Simulation & Process Modelling, Vol. 1, No. 2, pp.1–2.

Biographical notes: TODO

1 Introduction

Several types of simulations are used in Supply Chain Management (SCM) in order to study the dynamics induced by the decisions made in such networks of companies. Kleijnen (2003) proposed the following four classes of types of Supply Chain (SC) simulations: Spreadsheet, System Dynamics (SD), Discrete-Event Dynamic Systems (DEDS), and Business Games. Next, this author claimed that "which of the four simulation types is applied in SCM depends on the problem to be solved," which is illustrated as follows (Kleijnen, 2003, p.85):

- "SD aims at qualitative insights (not exact forecasts)" such as demonstrating the bullwhip effect – Lee *et al.* (1997a,b) define this effect as the amplification of order variability in supply chains,
- "DEDS can quantify fill rates, which are random variables," and
- "Games can educate and train users."

Next, MultiAgent Based Simulation (MABS) is a form of DEDS, that is, the agents in a MABS are autonomous entities driven by discrete events, and, therefore, the specificity of MABS in comparison with the rest of DEDS is to focus on the global effect (macro level) resulting from the interactions of the local behaviours of entities (micro level). In this paper, we focus on MABS rather than DEDS in general. The goal of this paper is to detail Kleijnen (2003)'s claim in the context of choosing among two of the four simulation types. More precisely, we detail in what ways the choice of "which of the spreadsheet simulation or MABS is applied in SCM depends on the problem to be solved" in order to make explicit the relative strengths and drawbacks of these two simulation types in SCM. For that purpose, we provide examples demonstrating that some features of a SC model:

- make a type of simulation not appropriate. For instance, modeling interactions among companies is hardly possible with spreadsheet programs.
- make an approach more suited than another. For example, implementing decision making in a multi-product scenario is difficult with spreadsheet programs.
- are not discriminating. For instance, implementing the causes of the bullwhip effect proposed by Lee *et al.* (1997a,b) is as long with spreadsheet programs as with MABS.

To our knowledge, (Kleijnen and Smits, 2003) and (Kleijnen, 2003) are the only two papers comparing different approaches of simulation in SCM. However, these two papers also point to comparisons within the four simulation types, i.e. comparisons of several models of MABS, but never of a model of MABS with a model of a different type.

Complementing Kleijnen (2003)'s claim is important because the choice of what simulation type to implement is often a first stage which has next a great impact on what can and cannot be studied with the simulation. For example, we have just presented the example provided by Kleijnen (2003) about the educational purpose of games. Since MABS allows for more realistic simulations, it is also possible to model games with MABS, but simulation designers tend to always use the power of MABS to propose more complex models which are too complex to educate people. Another example about the importance of the choice of the simulation approach deals with simulation speed. In fact, a choice may be necessary between the high simulation speed obtained with spreadsheet programs, and the poor realism of such models in comparison with MABS. More specifically, a spreadsheet simulator may be enough if many simulations are necessary in order to explore a large space of parameters, while MABS is required if we prefer a closer reproduction of reality.

Finally, the contributions of this paper are threefold. First, we detail the code of a spreadsheet simulation of the Quebec Wood Supply Game (QWSG), which is a business game related to Sterman (1989)'s Beer Game. This code is complete, which allows its direct use in spreadsheet programs. The second contribution is the presentation of an evolution of this first simulation implemented as a MABS. The third and main contribution of this paper concerns the insights gained when implementing these two simulations based on the same model. Since this unique base model is the QWSG, the insights gained only apply to the context of this model, that is, to decision making in SCs, but not to other contexts, in particular to contexts outside of SCM.

The outline of this paper is as follows. Section 2 describes the QWSG. Section 3 presents the complete code of an implementation of the QWSG in a spreadsheet program. Section 4 describes in less detail the MABS model of an evolution of the QWSG. Finally, Section 5 compares these two simulations in order to underline the relative strengths and drawbacks of spreadsheet and MABS approaches.

2 The Québec Wood Supply Game (QWSG)

2.1 QWSG lineage

First of all, let us introduce the background of our two simulators, namely, the Quebec Wood Supply Game (QWSG), and its "relatives." The parents of the QWSG are two board-games based on the structure and dynamics of Sterman (1989)'s "Beer Game." They are called "Wood Supply Games" and were developped by Fjeld (2001) and Haartveit and Fjeld (2002). These games are role-playing simulations that simulate the material and information flows in a production-distribution system, and were designed to make human players aware of the bullwhip effect (Lee *et al.*, 1997a,b). They illustrate how complex supply chain dynamics can be, even when each company has a simple model. Compared to the classic Beer Game, the Wood Supply Games introduce divergent product flows to increase their relevance to the North European forest sector. According to Fjeld (2001), such a bifurcation represents the broad variety of products (paper, books, paperboard boxes, furniture, buildings...) manufactured from a few types of raw materials (wood). FOR@C (2007) has kept this bifurcation and adapted the structure of the supply chain to the forest sector in Quebec (a Canadian province). This version is called the Quebec Wood Supply Game (QWSG), and we use it as the basis of our two simulators. The structure of the supply chain simulated in the QWSG is displayed in Figure 1. The main difference between the original Wood Supply Games and our QWSG

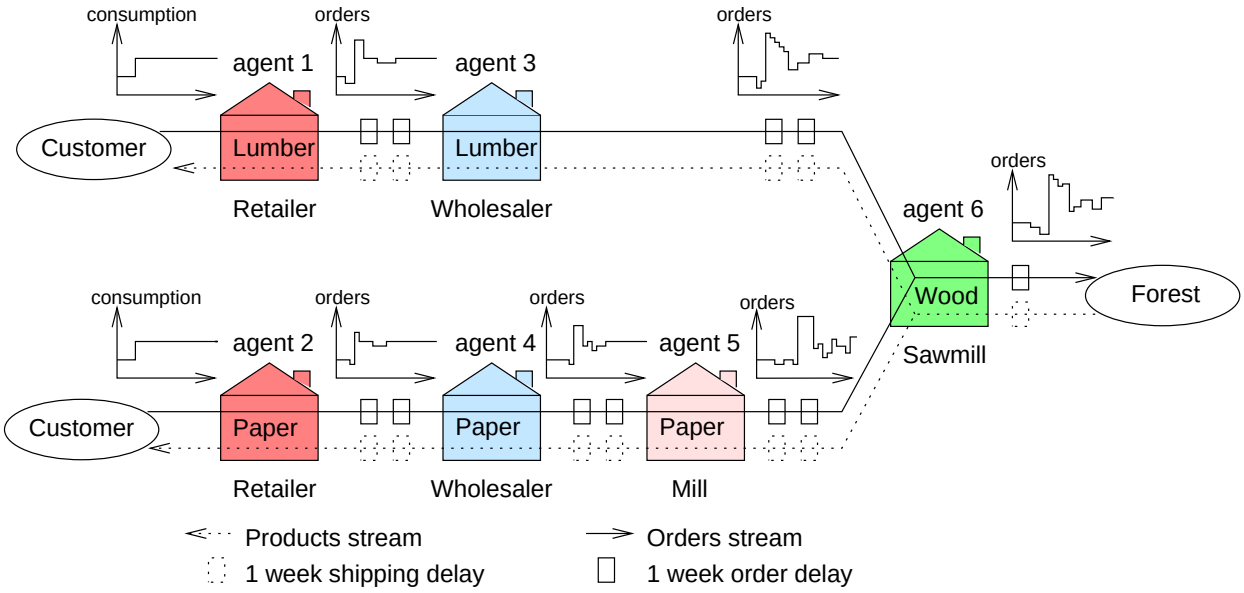

Figure 1: The amplification of order variability, also known as bullwhip effect (Lee *et al.*, 1997a,b), in the QWSG.

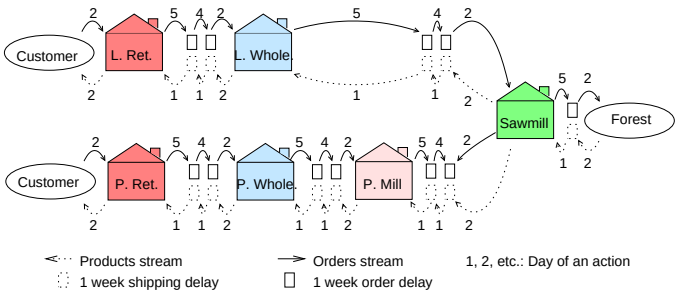

Figure 2: The run of the QWSG.

is in the length of the lumber and paper chains, which are either different or the same, that is, Fjeld's games have an additional company between the LumberWholesaler and the Sawmill. This difference corresponds to the specificities between Quebec and Northern European wood industries. The goal of this paper is not to support the choices made about the design of the QWSG, that is, the QWSG is only described in order to understand how the spreadsheet simulation operates and how it differs from the MABS simulation.

2.2 Players behaviour in the QWSG

Figure 1 illustrates the role taken by each of the six (human or artificial) players of the QWSG. The game is played by turns: each turn represents a week, and is split in five days played in parallel by every player. Figure 2 details the action carried in parallel on each day by every player, as now presented. On the first day, players receive products and put them in their inventory (these products were sent two weeks earlier by their supplier, because there is a two-week shipping delay), and move the products between the two squares called "shipping delay" between themselves and their customer (see the arrows in Figure 2). Then, on the second day, players look at their incoming orders, and try to fill them by moving items from their inventory into their outgoing "shipping delay" square. If they have backorders they try to fill those as well. Backorders correspond to products requested by clients, but that cannot currently by shipped. Their role is to represent products that should have been shipped in the past or in the current week, and that have to be shipped as soon as possible. In other words, there are no lost sales in this model because clients are assumed to wait infinitely for product availability. Backorders are recorded as negative inventory levels. If they do not have enough products in inventory, they ship as much as they can, and add the rest to their backorders. On the third day, players record their inventory or backorders. On the fourth day, players advance the order slips between every two successive "ordering delay" squares. On the last day, players place an order to their supplier(s) by writing a number on an order slip and by putting this slip in their outgoing "order delay" square; they also record this order. To decide the amount to order, players compare their incoming orders with their inventory/backorder level so that their inventory is well managed. Since the bullwhip effect deals with order variation, this effect depends on the orders placed here by every company. Finally, a new week begins with a new day 1, and so on.

Each position is played in the same way, except the Sawmill because this position receives two orders (one from the LumberWholesaler, another from the PaperMill) that have to be aggregated when placing an order to the Forest. The Sawmill can evaluate its order by basing it on the lumber demand and/or on the paper demand. In other

words, the bifurcation in the supply chain makes the decision making harder for the Sawmill. Moreover, the Sawmill receives one type of product, and each unit of this product generates two units, namely, a lumber and a paper unit. That is, each incoming unit is split in two: one piece goes to the Sawmill's lumber inventory, the other goes to the Sawmill's paper inventory. Notice that these two simulated pieces represent different quantities of real materials that are measured in different units.

2.3 Game purpose and simulations

The QWSG demonstrates the bullwhip effect, which is defined by Lee *et al.* (1997a,b) as the amplification of the order variability in the supply chain. Both retailers in Figure 1 place orders more variable than the demand of the two end-customers, next, both wholesalers also adds variability in the demand signal, ... and, eventually, the Forest receives a demand very variable, and also unpredictable, while this demand was constant except a single increase at the level of the two customers. In order to reduce the bullwhip effect, our two simulations allow using information sharing, as suggested by Lee *et al.* (1997a) and developped for the QWSG by Moyaux *et al.* (2007). Precisely, any order is represented by a vector (O, Θ) and is the sum of its two elements, instead of a unique number $X = O + \Theta$ where O and Θ are hidden. Os follow a lot-for-lot scheme (see Equation 13) and transmit thus the market consumption from every company to its direct supplier(s). Notice that a consequence of using a lot-for-lot scheme is that no bullwhip effect occurs in O. Next, Θs are adjustment orders that manage the complement of company requirements, because ordering only the quantities O is not enough to manage inventories. In fact, the ordering and shipping delays make so that more or less products are needed in comparison with the Os returned by the lot-for-lot scheme, and these quantities may be ordered in Θ. In other words, Θs directly depend on delays in our model (see the example in Equation 14).

We now present our spreadsheet simulation model, which closely implements the QWSG and allows using (O, Θ) orders.

3 The Spreadsheet Simulator

In order to present our spreadsheet simulation, Subsection 3.1 first introduces the notations used. Then, we give the equations that implement one company. All companies are represented with the equations in Subsection 3.2, except the Sawmill, which has the differences outlined in Subsection 3.3. Finally, we illustrate in Subsection 3.4 the behaviour of companies with the two equations of an ordering strategy implementing the placement of (O, Θ) orders as suggested by Moyaux *et al.* (2007).

3.1 Notations

Since orders may have up to two dimensions, called O and Θ, we note Op_w^i the component O placed in Week w by company i, and Θp_w^i the corresponding adjustment order Θ. The way to calculate Op_w^i and Θp_w^i depends on the company-agent's behaviour, that is, on its ordering strategy, which will be illustrated with an instance of ordering strategy in Subsection 3.4. The other variables used to represent Company i in Week w in our spreadsheet simulator are:

To_w^i = company i's outgoing transport in Week w.

Too_w^i = company i's outgoing transport in Week w corresponding to the current O.

Tob_w^i = outgoing Transport corresponding to backordered Os.

$To\Theta_w^i$ = outgoing transport corresponding to current and backordered Θs.

Ti_w^i = incoming transport.

I_w^i = on-hand inventory.

Op_w^i = placed order O.

Oo_w^i = outgoing order O.

Oi_w^i = incoming order O.

Ob_w^i = backordered Os.

Θp_w^i = Θ sent (placed).

Θo_w^i = outgoing Θ.

Θi_w^i = incoming Θ.

Θb_w^i = backordered Θs.

$D_w^{\text{lumber}} = Oi_w^1$ = lumber market consumption (demand).

$D_w^{\text{paper}} = Oi_w^2$ = paper market consumption (demand).

Except the inventory I and the two market consumptions D, the first letter in the notation of these variables indicates the modelled flow (T for transportation stream, or O and Θ for the two-dimensional ordering stream), and the second letter indicates if it is an incoming, outgoing, placed or backordered value of this stream for the considered company i. For the sake of implementation simplicity, the outgoing transportation To (quantity of products to ship) is split into three parts: Too for products shipped to fulfill O orders, Tob for backordered Os, and $To\Theta$ for current and backordered Θs. Again for the sake of program simplicity, we can note that, as in the QWSG, it is easier not to represent backorders as negative inventory levels $I < 0$, but as the separate variable Ob, because (i) the definitions of Too, Tob and $To\Theta$ are shorter since they do not depend on the sign of I, and (ii) introducting Ob is similar to the introduction of Θb (there were no (O, Θ) orders in the QWSG). Consequently, $I_w^i \geq 0$ in our spreadsheet simulator, while I_w^i may be negative in the QWSG to represent backorders.

$$Oi^i_w = Oo^{i-1}_{w-1} \tag{1}$$
$$Ti^i_w = To^{i+1}_{w-1} \tag{2}$$

$$Too^i_w = \begin{cases} Oi^i_w & \text{if } I^i_{w-1} \geq 0 \text{ and } I^i_{w-1} + Ti^i_w \geq Oi^i_w \\ I^i_{w-1} + Ti^i_w & \text{if } I^i_{w-1} \geq 0 \text{ and } I^i_{w-1} + Ti^i_w < Oi^i_w \\ Oi^i_w & \text{if } I^i_{w-1} < 0 \text{ and } Ti^i_w \geq Oi^i_w \\ Ti^i_w & \text{if } I^i_{w-1} < 0 \text{ and } Ti^i_w < Oi^i_w \end{cases} \tag{3}$$
$$Ob^i_w = Ob^i_{w-1} + Oi^i_w - Too^i_w - Tob^i_w \tag{4}$$

$$\Theta i^i_w = \Theta o^{i-1}_{w-1} \tag{5}$$
$$Tob^i_w = \begin{cases} Ob^i_{w-1} & \text{if } I^i_{w-1} \geq 0 \text{ and } I^i_{w-1} + Ti^i_w - Too^i_w \geq Ob^i_{w-1} \\ I^i_{w-1} + Ti^i_w - Too^i_w & \text{if } I^i_{w-1} \geq 0 \text{ and } I^i_{w-1} + Ti^i_w - Too^i_w < Ob^i_{w-1} \\ -I^i_{w-1} & \text{if } I^i_{w-1} < 0 \text{ and } Ti^i_w - Too^i_w \geq -I^i_{w-1} \\ Ti^i_w - Too^i_w & \text{if } I^i_{w-1} < 0 \text{ and } Ti^i_w - Too^i_w < -I^i_{w-1} \end{cases} \tag{6}$$

$$To\Theta^i_w = \begin{cases} -Too^i_w - Tob^i_w & \text{if } I^i_{w-1} \geq 0 \text{ and } I^i_{w-1} + Ti^i_w - Too^i_w - Tob^i_w \geq \Theta b^i_{w-1} + \Theta i^i_w \\ & \text{and } Too^i_w + Tob^i_w + \Theta b^i_{w-1} + \Theta i^i_w < 0 \\ \Theta b^i_{w-1} + \Theta i^i_w & \text{if } I^i_{w-1} \geq 0 \text{ and } I^i_{w-1} + Ti^i_w - Too^i_w - Tob^i_w \geq \Theta b^i_{w-1} + \Theta i^i_w \\ & \text{and } Too^i_w + Tob^i_w + \Theta b^i_{w-1} + \Theta i^i_w \geq 0 \\ I^i_{w-1} + Ti^i_w - Too^i_w - Tob^i_w & \text{if } I^i_{w-1} \geq 0 \text{ and } I^i_{w-1} + Ti^i_w - Too^i_w - Tob^i_w < \Theta b^i_{w-1} + \Theta i^i_w \\ -Too^i_w - Tob^i_w & \text{if } I^i_{w-1} < 0 \text{ and } Ti^i_w - Too^i_w - Tob^i_w \geq \Theta b^i_{w-1} + \Theta i^i_w \\ & \text{and } Too^i_w + Tob^i_w + \Theta b^i_{w-1} + \Theta i^i_w < 0 \\ \Theta b^i_{w-1} + \Theta i^i_w & \text{if } I^i_{w-1} < 0 \text{ and } Ti^i_w - Too^i_w - Tob^i_w \geq \Theta b^i_{w-1} + \Theta i^i_w \\ & \text{and } Too^i_w + Tob^i_w + \Theta b^i_{w-1} + \Theta i^i_w \geq 0 \\ Ti^i_w - Too^i_w - Tob^i_w & \text{if } I^i_{w-1} < 0 \text{ and } Ti^i_w - Too^i_w - Tob^i_w < \Theta b^i_{w-1} + \Theta i^i_w \end{cases} \tag{7}$$

$$To^i_w = Too^i_w + Tob^i_w + To\Theta^i_w \tag{8}$$

$$I^i_w = I^i_{w-1} + Ti^i_w - To^i_w \tag{9}$$

$$\Theta b^i_w = \begin{cases} \Theta b^i_{w-1} + \Theta i^i_w - To\Theta^i_w + Too^i_w + Tob^i_w + To\Theta^i_w & \text{if } Too^i_w + Tob^i_w + To\Theta^i_w < 0 \\ \Theta b^i_{w-1} + \Theta i^i_w - To\Theta^i_w & \text{if } Too^i_w + Tob^i_w + To\Theta^i_w \geq 0 \end{cases} \tag{10}$$

$$Oo^i_w = Op^i_{w-1} \tag{11}$$
$$\Theta o^i_w = \Theta p^i_{w-1} \tag{12}$$

Figure 4: QWSG in a spreadsheet program when information sharing with (O, Θ) orders are allowed.

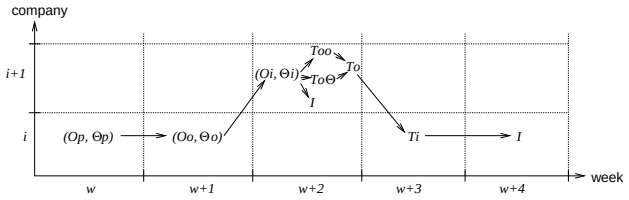

Figure 3: The trip of an order and the shipping of the corresponding products.

3.2 Company Model

Figure 3 presents the relations between variables which represent the transmission of an order, then the shipping of the corresponding items. We now detail these relations, as well as the others implementing our model. The "mechanical" part of the QWSG is modelled by the twelve equations in Figure 4. (The decision process needs two more equations to choose Op and Θp, as outlined in Subsection 3.4.) The equations in Figure 4 are in the order in which they are automatically fired by a spreadsheet program: (i) Equations 1 and 2 (in any order) first, then (ii) Equations 3 and 4 (because they rely on the result of Equations 1 and 2), (iii) Equations 5 and 6 (because they need the result of Equations 1, 2, 3 and 4), (iv) Equation 7, (v) 8, (vi) 9, and, finally, (vii) Equation 10. (These groups of equations are separated by vertical spaces in Figure 4.) Equations 11 and 12 may be fired anytime.

As illustrated in Figure 3, an order $(Op, \Theta p)$ is placed in Week w by a client i. This order is put in Week $w+1$ in $(Oo, \Theta o)$ to represent a first week of ordering delay. To represent the second week of delay, this order is received by supplier $i+1$ in Week $w+2$ in $(Oi, \Theta i)$. This order reception decreases supplier's inventory I in the same week, because products are shipped in Too and $To\Theta$, thus in To. Next, these shipped products are put in Week $w+3$ in client's Ti to represent a first week of shipping delay. To model the second week of shipping delay, these products are put in client's inventory I in Week

$w+4$. Therefore, in order to implement a two-week delay in information transmission and in transportation, order and transport variables are doubled, which explains the existence of pairs $\{To^i_w, Ti^i_w\}$ for transportation, and $\{(Oo^i_w, \Theta o^i_w), (Oi^i_w, \Theta i^i_w)\}$ for order placement. The simulation begins in Week $w = 1$ and ends in $w = 50$ to have a simulation over a year. In this section, for simplifying equations, $i = 1$ represents a company (except the Sawmill on which we shall come back), and $i+1$ is i's supplier, even if this is compatible with Figure 1 only for the PaperMill, and should read $i + 2$ instead for both retailers and both wholesalers. When $i = 1$ denotes the PaperRetailer, Oi^1_w corresponds to this retailer's demand, that is, the market consumption ($Oi^1_w = D^{\text{paper}}_w$).

To^i_w represents products sent to its client by Company i in Week w. To make the calculation of this quantity easy, it is divided into three parts (see $To^i_w = Too^i_w + Tob^i_w + To\Theta^i_w$ in Equation 8 in Figure 4). Too^i_w represents products that are first sent to fulfill the current order (or the quantity of products the company is able to ship when inventory and incoming transports are not enough), as reflected by Equation 3. Then, company i ships the quantity Tob^i_w of products in Equation 6, in order to reduce its backorder Ob. Finally, when orders are fulfilled and there are no backordered Ob left, $To\Theta^i_w$ products are sent in Equation 7 to reduce the incoming Θ order Θ^i_w, and Θb^i_w (i.e. the backordered Θ).

Like orders, backorders are represented by two variables: Ob^i_w in Equation 4 represents backorders created by unfulfilled O, and Θb^i_w in Equation 10 backorders created by unfulfilled Θ.

Then, incoming transport is supplier $i + 1$'s last week outgoing transport (see Equation 2 in Figure 4).

Equation 9 represents the fact that inventory level is previous inventory level plus inputs minus outputs.

Figure 3 shows how orders are delayed between a client i and its supplier $i+1$. Each order is placed in $(Op^i_w, \Theta p^i_w)$, goes next in $(Oo^i_{w+1}, \Theta o^i_{w+1})$ to simulate the first week of delay, and is finally put in supplier's $(Oi^{i+1}_{w+2}, \Theta i^{i+1}_{w+2})$ to simulate the second week of delay. This explains why incoming order (O, Θ) is the last week client $i-1$'s outgoing transport (Equations 1 and 5). Figure 3 also explains how Oo^i_w and Θo^i_w are setup in Equations 11 and 12.

Finally, Figure 5 gives an overview of the implementation of the 38 first weeks of the lumber market demand and of the LumberRetailer in our spreadsheet simulator. In this figure, the market demand is for 11 products in the four first weeks, then for 17 products until the end of the simulation in Week $w = 50$, and the column "Inventory," for example, contains I^1_w for all weeks w.

3.3 Specificities of the Sawmill model

Every company is setup in the same way with the twelve previous equations, except the SawMill which has to process two types of products, namely, lumber and paper. As a consequence, some parts of this company are doubled to manage this particularity. Specifically, we use the pairs of variables $\{Oi^{\text{6-lumber}}_w, Oi^{\text{6-paper}}_w\}$, $\{\Theta i^{\text{6-lumber}}_w, \Theta i^{\text{6-paper}}_w\}$, $\{Op^{\text{6-lumber}}_w, Op^{\text{6-paper}}_w\}$, $\{\Theta p^{\text{6-lumber}}_w, \Theta p^{\text{6-paper}}_w\}$, $\{I^{\text{6-lumber}}_w, I^{\text{6-paper}}_w\}$, and $\{To^{\text{6-lumber}}_w, To^{\text{6-paper}}_w\}$. Note that $Oo^6_w, \Theta o^6_w$ and Ti^6_w are unique. Indeed, we operate as if there were a lumber SawMill and a paper SawMill having the same product input Ti^6_w: each subcompany in the Sawmill receives what is in the incoming transport Ti^6_w, because one unit of wood coming from the Forest provides one unit of lumber and one unit of paper. In the same way, each subcompany wishes to place a specific (O, Θ) order: the lumber Sawmill would like to place orders $(Op^{\text{6-lumber}}_w, \Theta p^{\text{6-lumber}}_w)$, while the paper Sawmill prefers $(Op^{\text{6-paper}}_w, \Theta p^{\text{6-paper}}_w)$. The design of the ordering strategies managing the variables $(Op^{\text{6-lumber}}_w, \Theta p^{\text{6-lumber}}_w)$ and $(Op^{\text{6-paper}}_w, \Theta p^{\text{6-paper}}_w)$ are outined in the next subsection, since they are the same as for the five other companies. However, the Sawmill has to aggregate $(Op^{\text{6-lumber}}_w, \Theta p^{\text{6-lumber}}_w)$ and $(Op^{\text{6-paper}}_w, \Theta p^{\text{6-paper}}_w)$ into a common $(Op^6_w, \Theta p^6_w)$ sent to the Forest. In general, we use $Op^6_w = (Op^{\text{6-paper}}_w + Op^{\text{6-lumber}}_w)/2$ and $\Theta p^6_w = (\Theta p^{\text{6-paper}}_w + \Theta p^{\text{6-lumber}}_w)/2$, but the definition of the most efficient function is an open question addressed by Moyaux *et al.* (2004a). These two functions implement the aggregation of paper and lumber requirements when the Sawmill places an order.

3.4 An Example of Ordering Strategy

Equations 1 through 12 describe the shipping and ordering *flows* in a Company i, but not how decisions are made by this company. Since the purpose of this paper is not the study of decision making, but the simulation of supply chains, we only indicate how to implement one of the ordering strategies presented in (Moyaux *et al.*, 2007). This scheme was designed to reduce the bullwhip effect thanks to information sharing with (O, Θ) orders. To this end, incoming Os, which are the market consumption when all clients use this scheme, are transmitted from clients to suppliers according to the lot-for-lot policy (i.e. the company orders what is demanded by the client), as shown in Equation 13.

$$Op^i_w = Oi^i_w \tag{13}$$

Next, company requirements are added to incoming Θs, and this sum is sent as Θ to the supplier (see Equation 14). In these conditions, $\lambda * (Oi^i_{w-1} - Oi^i_w)$ is an estimation of the inventory reduction caused by the variation of Oi, which reflects the variation of the market consumption. Note that we use $\lambda = 4$ for all companies, which corresponds to the sum of ordering and shipping delays in the QWSG. To see this, let us consider the following example. The demand O increases by seven units between Weeks -1 and 0 ($Oi^i_{-1} - Oi^i_0 = -7$), then remains at this new level ($Oi^i_{w-1} - Oi^i_w = 0$ for $w > 1$). The company will keep receiving the old quantity for the $\lambda = 4$ weeks 0, 1, 2 and 3 because of O orders placed in weeks $w-4$, $w-3$,

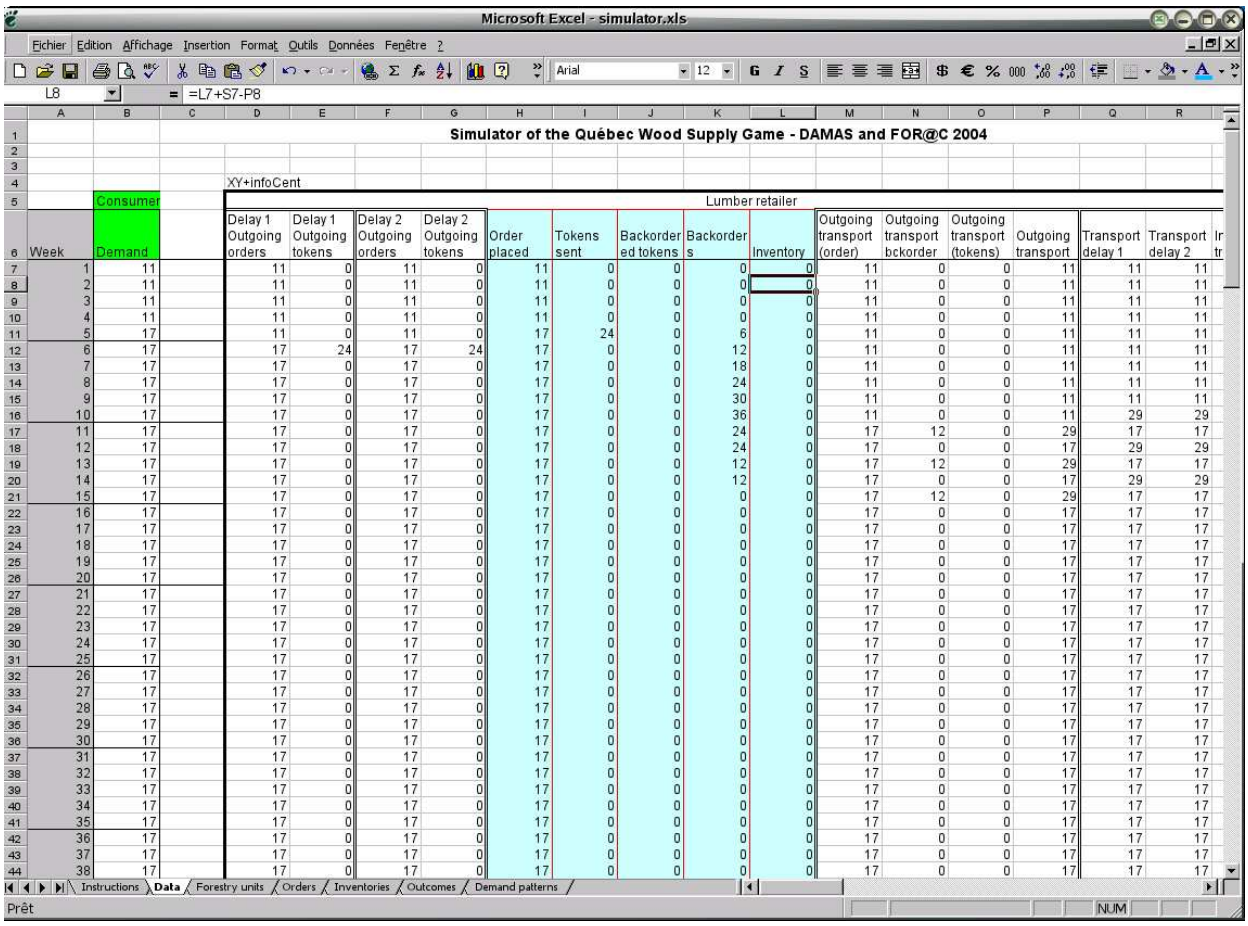

Figure 5: Implementation of the Québec Wood Supply Game (QWSG) in a spreadsheet program.

$w-2$ and $w-1$. Since the company ships the new quantity from Week 0 on, this company ships more products than it receives in the 4 weeks 0, 1, 2 and 3, which causes its inventory decrease by 7 units in each of these 4 weeks. This is the reason for which this company needs to order 4*7 units in Θ, in addition to what is ordered in O.

$$\theta p_w^i = \theta i_w^i - \lambda * (Oi_{w-1}^i - Oi_w^i) \tag{14}$$

4 The MABS Simulator

The MultiAgent Based Simulation (MABS) is based on the spreadsheet simulator. To this end, the model in both simulators is the QWSG, except that the MABS uses the agent programming language JACK™ in order to model every company with increased realism.

4.1 Generic model and implementation

Of course, such a realism implies a greater complexity of the simulator, which explains why companies are implemented with the agent-oriented language JACK™ from the Agent Oriented Software Group (2003). This language helps us implement our simulator thanks to its characteristics, such as the high-level representation of company behaviours, implementation flexibility, and suitability for distributed applications (Agent Oriented Software Group, 2003). Using the agent paradigm also allows taking company autonomy more into account. Indeed, JACK™ provides tools to implement systems of intelligent agents, that are both reflex and goal-directed. The first of these two kinds of decision making families is the most simple since reflex means that agents only react to events, even though this reaction may look quite "intelligent" by depending on the past (i.e. on some internal state) of the agent. The second kind deals with goal-directed behaviours in which agents are even more "intelligent" in the sense that they take the initiative in order to fulfill their goals. For instance, the reflex behaviour implemented by Equations 13 and 14 may be read as "an order has arrived $\Rightarrow$ place an order," which may also be implemented by a MABS. An example of goal-directed behaviour to replace it would be "I want to keep my inventory at level $X \Rightarrow$ place an order whenever its level is below X." In the current version of our simulator, we only use the reflex behaviour of agents because we only implemented the ordering and shipping streams, but JACK™ easily allows making this behaviour

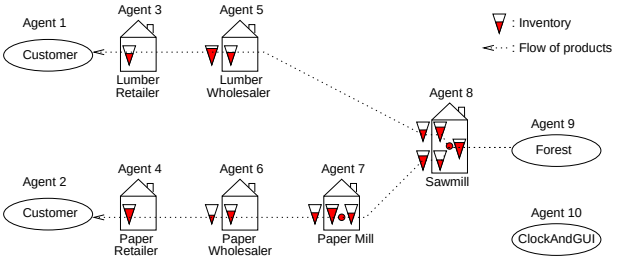

Figure 6: The forest supply chain modelled with JACK™ agents.

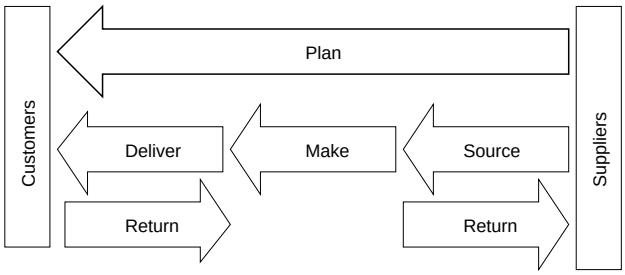

Figure 8: The Supply-Chain Operations Reference (SCOR) model of the Supply Chain Council (2007).

evolve toward a more complex one in order to implement sophisticated goal-directed decision making. The Agent-Oriented Programming Language JACK™ uses events to fire plans. Specifically, an event models the occurrence of an intra-agent event (i.e. a communication between parts of the agent) or of an inter-agent event (i.e. a message to another agent) that agents must address. A plan is the procedural description of actions that an agent achieves when it receives a specific (internal or external) event. Figure 6 represents the flows of products in inventories and in shippings in the simulated supply chain. Each company has one or several inventories figured as triangles; the total height of a triangle represents inventory capacity, while its filling represents its current level. In PaperMill and Sawmill, circles represent the transformation of a quantity of material. (Production is an addition to the QWSG, in which companies were only seen as warehouses.) This work-in-process inventory cannot be partially filled, that is, it is exclusively either full or empty. Both wholesalers are like in the QWSG: they do not have production activity and they have a truck to ship products to their retailer. The other companies differ from the QWSG: both mills manage raw material inventories due to their production activity, and none of the retailers ship products because customers come to buy them. Like in the QWSG, time is discrete in order to make it easier to control that the simulation works as we intend it to do, which avoids issues due to parallelism. For example, depending on the traveling of messages on the network between agents (when agents are spread on two or more computers), two instances of exactly the same simulation without any randomisation may incur different outputs. Also as in the QWSG, each company has a product inventory ready to be shipped, called _finishedProductInventory in Figure 7, and there are delays between companies modelled as the queue _productsInTruck; note here that we choose to begin the names of variables with a "_", and the name of JACK™ plans with "Pl".

The PaperMill is the only agent that has the basic structure shown in Figures 7 and 10, while the other agents are adaptations on this basis. Figure 7 shows the changes between PaperMill in the QWSG and the PaperMill in our MABS: there are information and product streams travelling the four functions Transport, Deliver, Make and Source of the company. Besides that, every shipping delay is represented as an inventory at the visual level in Figures 6 and 9, and as a queue called _productsInTruck at the logical level in Figure 7. Product batches go through this queue and are given to the client's _rawMaterialInventory as soon as the shipping delay has elapsed. We add capacity to this queue, whereas shipping delays in the QWSG (that can be seen as trucks) can ship as many products as needed. We add capacity to inventories too. The inventory in the QWSG, like I_w^i in our spreadsheet simulator, corresponds to _finishedProductInventory in Figure 7. We can note that the four functions Transport, Deliver, Make and Source are based on the first level of the Supply-Chain Operations Reference (SCOR) model from the Supply Chain Council (2007) illustrated in Figure 8. Indeed, in comparison with SCOR, we put a function Transport in addition to Deliver. Transport is a truck which takes the place of the two shipping delays in the QWSG. Next, if we do not consider Returns, SCOR represents a company as three activities: Deliver (shipping to clients), Make (basic activity of the company), Source (purchase from suppliers). The additional levels of SCOR details these three activities, but we do not consider that in the current version of our simulator. We can also note that, in our spreadsheet simulator, all companies were modelled with the same equations, except the Sawmill, while in our MABS, only the PaperMill has the basic code outlined in Figures 7 and 10, while all other companies derive from this: the Sawmill has two Transport and two Deliver, the LumberWholesaler and the PaperWholesaler have no Make, as well as the LumberRetailer and the PaperRetailer. Indeed, the QWSG and our spreadsheet simulator do not make any distinction between companies in the distribution network (the retailers and the wholesalers) and production companies (the PaperMill and the Sawmill). On the contrary, we made our MABS model more realistic by adding the Make function to the PaperMill, which forces us to add the Source function. Both of these functions contain a limited inventory, called _workInProcessInventory in Make (cf. Figure 7), and _rawMaterialInventory in Source. The _workInProcessInventory in the Make function represents the product batch being processed, that is, it is a small inventory which is either full or empty. We now detail how the simulation works.

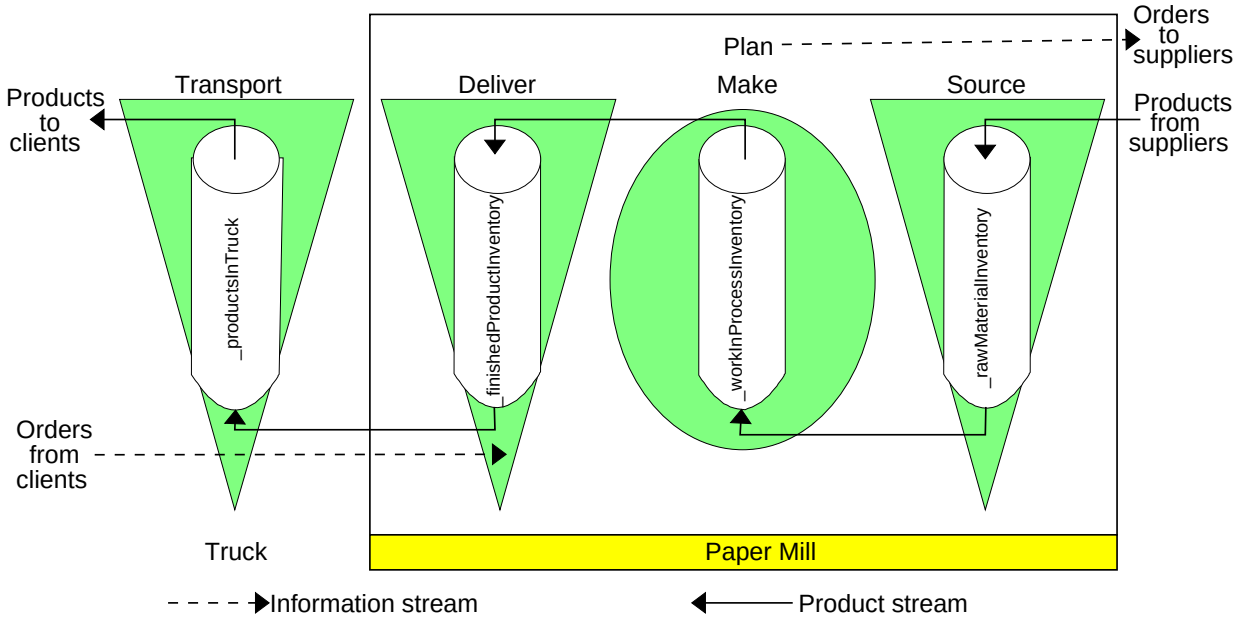

Figure 7: The PaperMill model.

4.2 Details of the different types of agents

We first present the five different types of agents (i.e. the company-agents, and an additional agent called ClockAndGUI) in the simulation, and then the implementation of the company-agents. Agents can be divided into five categories:

1. Customers are modelled as two agents buying products from their retailer. Customer agents place orders according to a distribution law representing different kinds of consumption patterns.

2. Retailers and Wholesalers, i.e. the distribution network, are companies having neither production activity nor raw material inventory to manage: products coming from the supplier do not wait in the _rawMaterialInventory because these agents directly move products from _rawMaterialInventory to _productsInTruck. Moreover, these four agents have to place orders to their suppliers.

3. PaperMill is the basic company-agent in our model in Figure 10, because it has the Transport, Deliver, Make and Source functions, and the Sawmill is an extension on this. Make and Source share a common code for the PaperMill and the Sawmill, but are created differently, e.g. _workInProcessInventory and _finishedProductInventory are arrays containing an integer in the case of the PaperMill, and two integers in the case of the Sawmill, and _productsInTruck are arrays containing one array of integers in the case of the PaperMill, and two arrays of integers in the case of the Sawmill.

4. Forest is the most upstream company-agent, and thus has no supplier and does not place any orders. It is assumed to be an infinite source of wood in the QWSG but we assume in our model that its capacity is given by a cutting plan representing companies' procurement contract with the provincial government. It should be noticed here that FOR@C (2007) organizes games of the board version of the QWSG in which the production of the Forest is constrained.

As previously noted, we assume all companies work at the same time, and that this time is discrete, i.e. every company waits for the other company to complete the actions in the current day, before beginning the next day (and maybe, the next week). This is the purpose of the ClockAndGUI agent, which both broadcast the time to all other agents in order to centralize the control of the simulation, and display the graphical user interface (GUI). Figure 9 gives an overview of the information displayed by the ClockAndGui interface.[1].

5. The ClockAndGui is an agent multi-casting an event representing the time in the whole supply chain. Each tick of the clock makes company-agents perform actions by triggering their JACK™ plans. ClockAndGui can also display information on the GUI, and/or write raw simulation outcomes on the shell, and/or put simulation outcomes in an Excel (Microsoft Corp., 2004) file with the Java Excel API 2.3.12 from Andy Khan (2004).

We should finally note that Transport could have been modelled as truck agents instead of as a function in each company. It would have allowed us to focus on the impacts

[1]We would like to thank Eve Levesque who adapted the GUI from the Nereus project (Paquet, 2001; Plamondon *et al.*, 2003; Soucy, 2004) in DAMAS laboratory at Université Laval (Québec City, Québec, Canada).

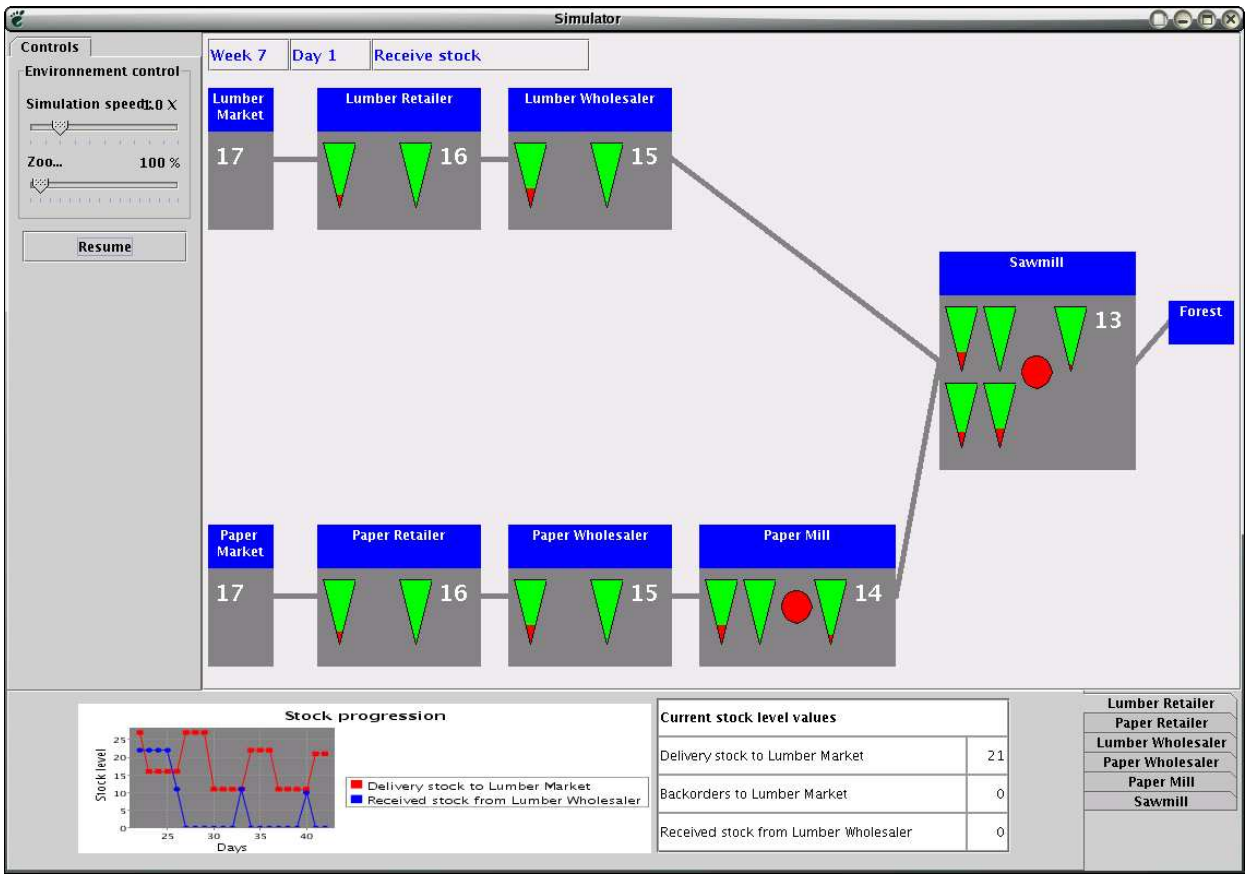

Figure 9: Graphical user interface of the MultiAgent Based Simulation (MABS).

of unanticipated transportation events on the bullwhip effect. We currently assume that transportation can simply be viewed as a queue in which there is no problem and managed by each company.

4.3 Implementation of the components of the agents

Since we use the JACK™ framework, this description of the implementation is a summary of the JACK™ plans used to model companies. We have seen that Transport, Deliver, Make and Source functions travel with the information and the product streams. Figure 10 gives more details about the implementation of these four functions by indicating the name of the JACK™ plans involved in each of these four functions. We now detail the code of these plans. These plans simulate and improve the five days per week of the QWSG. Since JACK™ agents are event-driven, every plan is triggered by a particular event. For this reason, we first present the events, then the plans in our agents. There are five types of events in our simulation. The first two types directly come from the QWSG and have a physical meaning, while the other three types are necessary for the simulation operation:

1. EvOrder: These are messages (external events) containing three information, namely, the (O, Θ) order

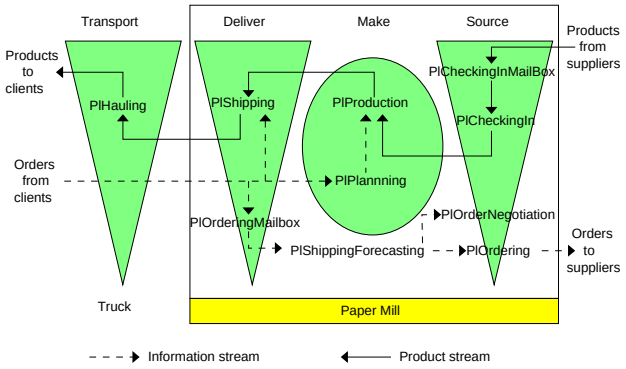

Figure 10: JACK™ plans in the PaperMill.

and the character string of the type of the ordered item (paper or lumber).

2. EvShipping: Each of such external events may be seen as a delivery truck. These messages have a field representing the quantity of products of the shipping and another field for the name of the product hauled (paper or lumber).

3. EvTime: These events are messages representing the time multicast by ClockAndGui to drive all company-

agents' plans.

4. EvAskForUpdateGUI: This message event is sent by company-agents to the ClockAndGui agent in order to display on the user interface the state of the company-agent.

5. EvInitGUI: This message is sent by company-agents to ClockAndGui at the beginning of the simulation to initialize the display, and is similar to EvAskForUpdateGUI, except that it also contains capacity parameters.

We now present the plans triggered by these events. See Figures 7 and 10 for the signification of notations. Note that all company-agents' plans are driven by events EvTime, except plans whose name finishes by "MailBox" that are driven be a message event EvOrder or EvShipping which may be received anytime. We first outline the latter type of plans.

- PlCheckingInMailBox: Each time an EvShipping truck arrives, the quantity of products it indicates is added to company's variable _rawMaterialInventory.

- PlOrderingMailBox: Each time an EvOrder message arrives, the (O, Θ) order it contains is memorized. The purpose of this memory is to transmit (O, Θ) to PlOrdering and PlShipping.

Next, the other plans are fired by the day in week indicated by the EvTime multicast by the ClockAndGUI agent. The number of the item in the following list indicates what day of the week fires the corresponding plans:

1. PlHauling sends to the client an EvShipping truck indicating the last quantity shipped in _productsInTruck.

2. PlOrderNegociation will negociate (when implemented) the price, quantity, shipping date, and so on, with the supplier whenever an EvTime (sent by the ClockAndGUI agent) indicates the 2^{nd} day in week.

3. PlOrdering sends an EvOrder message to the supplier indicating the quantity (O, Θ) calculated by the ordering policy of the agent. The ordering policy is included in PlOrdering and, in the current state of our simulator, consists of the same ordering strategies as in the spreadsheet simulator. However, some strategic behaviour of the agent may be added here in the future.

4. PlProduction performs the activity of the agent: If production time has elapsed, i.e. if _week in EvTime is superior to company's (_beginningProductionWeek + _productionDuration), then (i) move items from _workInProcessInventory into _finishedProductInventory, and (ii) if a new batch of products can be launched, process it, else set _beginningProductionWeek such as this plan triggers the following week.

5. PlShipping tries to fulfill the client's demand memorised by PlOrderingMailBox. In detail, PlShipping (i) tries to fulfill current O and backordered O, (ii) tries to fulfill current Θ and backordered Θ, and (iii) puts these four quantities into company's _productsInTruck.

6. PlPlanning will plan (when implemented) which product to produce if the company processes different kinds of products.

7. PlShippingForecasting will anticipate the future demand of the customer (not yet implemented)

All these plans belong to the PaperMill. Since neither Customers have either PlHauling or PlShipping, they have a special version of the PlOrdering plan:

3. PlOrdering send to the supplier (here, a Customer) an EvOrder message indicating the market consumption in O and zero in Θ.

Finally, the four companies in the distribution network and the two Customers do not produce anything, which explains why the following plan replaces PlProduce:

4. PlNoProduction move items from _rawMaterialInventory into _finishedProductInventory if _finishedProductInventory has enough room.

All the elements we have just described replicate a more realistic supply chain than the QWSG. Similar to our spreadsheet simulator, the only cause of the bullwhip effect that appears in this agent simulator is the consequence of ordering and shipping delays.

5 Comparison of our two Simulators

We now compare the two approaches to simulate the decision making in SCs adopted in this paper. We do not aim to propose a general comparison of MABS with spreadsheet simulators, but only to precise the differences we encountered with them. Basically, using a spreadsheet program is the quickest way to implement a simulator, as long as this simulator is not too complex. In particular, it is an interesting way to identify the salient points of the model. On the other hand, the advantage of a MABS is its realism, which may also be of huge importance depending on the reasons for which a simulation is needed. In fact, if the goal of a simulator is to:

- *replicate* the real-world as nearly as possible in order to predict what will happen in reality, then the MABS approach should be chosen.

- *understand* the underlying mechanisms of a phenomenon, then a spreadsheet program forces the designers of a simulator to simplify their model as much as possible so that basic mechanisms in the supply chain are emphasized, which allows understanding them. On the contrary, realistic MABSs mix everything and make understanding what occurs more challenging.

Issue	Spreadsheet	Vs.	MABS
1. Perturbations	Impossible	>	Short
2. Interactions (e.g. negotiation)	Impossible	>	Long
3. Multi-product scenarios	Impossible	>	Long
4. Implementation of the causes of the bullwhip effect			
4.1 Updating of demand forecasting	Short	=	Short
4.2 Order batching	Short	=	Short
4.3 Price fluctuation	Short	=	Short
4.4 Rationing and shortage gaming	Short	=	Short
5. Capacities			
5.1 Transportation capacity	Short	=	Short
5.2 Inventory capacities	Short	=	Short
6. Implementation time			
6.1 Time to learn the tool	Short	<	Long
6.2 Coding time	Short	<	Long
6.3 Validation of the code	Short	<	Long
7. Time to run simulations	Short	<	Long
8. Publication of the simulation model	Short	<	Long

Table 1: Time needed to implement different aspects of supply chain management with the two simulation approaches (Impossible > Long > Short).

Next, Table 1 summarizes the times needed to perform some tasks with these two approaches. The following subsections explain Table 1. Basically, some aspects are not possible to model with a spreadsheet simulation (cf. Issues 1), or so long to implement that they can be seen as impossible (cf. Issues 2 and 3). Next, all what can be represented with the spreadsheet approach can also be with MABS, and the implementation with MABS is at least as easy as with the spreadsheet simulator (cf. Issues 4 and 5). However, the higher realism allowed by the MABS approach has a price (cf. Issues 6, 7 and 8).

5.1 Perturbations

Perturbations are events (e.g. transportation delays, or issues dealing with the reliability of processes) which make so that plans cannot be followed. The most intuitive way to model perturbations is to represent them as events occuring anytime. By definition, any DEDS, such as a MABS, is therefore well suited to manage perturbations, that is, managing events is the purpose of this simulation type. For instance, let us assume that a machine breaks down at time $T2$ between the two time steps $T1$ and $T3$. The MABS is naturally able to take account of this event from $T2$, without waiting for the next tick $T3$ of the clock. As a consequence, Table 1 describes as "Short" the simulation of perturbations with MABS.

In contrast, the spreadsheet simulator will know about the breakdown only at period $T3$. This example illustrates something similar to what we shall see in the next subsection, that is, nothing can happen between two time steps in spreadsheet simulations, which explains the "Impossible" in Table 1.

5.2 Interactions (e.g. negotiation)

Interactions are also difficult to implement in a spreadsheet simulator because it is hard to fit an interaction between two time steps. In fact, when companies interact in this type of simulators, a whole sheet of the spreadsheet program is devoted to that, rather than to any other question of SCM. More precisely, the lines in the spreadsheet represents the different rounds in an interaction, rather than the time separating two interactions (the latter possibility corresponds to our spreadsheet simulation). As a consequence, studying interactions with a spreadsheet simulator is described as *almost* "Impossible" in Table 1.

In order to illustrate this, let us consider a simple bilateral negotiation in which a requester makes offers to a provider until either the provider accepts an offer or the requester gives up. This protocol may obviously end after a single round when the provider accepts the first offer, or run forever when the requester rejects all the offers while the requester is insisting. Studying the operation of such an interaction protocol in a spreadsheet simulator would thus take between one and an infinite number of lines, each line representing a round in the protocol. These lines would have to be in a sheet different from the main one, that is, the main sheet would simulate all the time steps and, sometimes, use the other sheet in order to run the interaction protocol. Of course, it is not completely "Impossible" to implement this in a spreadsheet program, in particular thanks to macros, but the difficulty is such that we described it as harder than "Long."

On the contrary, the concept of agents, on which MABS relies, focusses on interactions among agents. Implementing our example of bilateral negotiation is straightforward with MABS. However, tracking the interactions is often difficult, especially when more than the two agents in our

example of a bilateral negotiation are involved, because each agent impacts on each other. This difficulty is even greater when the agents run on different computers, as this is allowed by MABS but not by spreadsheet programs. As a consequence, Table 1 describes MABS as "Long."

5.3 Multi-product scenarios

It seems easy to enhance our spreadsheet simulator in order to consider companies buying and selling different kinds of products, because this only requires adding an indice p indicating the type of the product for all the variables, e.g. $To^i_{w,p}$, $Ti^i_{w,p}$, $Op^i_{w,p}$, etc. Technically, every product p would flow in a different supply chain represented in its own sheet of the spreadsheet. Unfortunately, multi-product scenarios are more complicated than just representing different flows of products, because decision making becomes much more complicated. In particular, production planning may be needed (no production conflicts may occur with a single type of products), and there might be some interactions between production planning and the ordering strategies calculating Op and Θp. All these modifications are very "Long" to implement in a MABS simulator, and are thus *almost* "Impossible" to program in a spreadsheet simulator (see Table 1).

5.4 Implementation of the four causes of the bullwhip effect proposed by Lee *et al.* (1997a,b)

Since our spreadsheet simulator implements a model related to the Beer Game, it only addresses one cause of the bullwhip effect, namely, the misperceptions of feedback identified by Sterman (1989). But the bullwhip effect is a more complex phenomenon, and other causes have been proposed to explain its occurence. Then, we may wonder how to extend our two simulations so that more known causes of the bullwhip effect are taken into account, either separately, or at the same time in order to study their interactions and the interactions of the solutions to these causes. We present this extension by explaining the four causes of the bullwhip effect identified by Lee *et al.* (1997a,b), then how these causes may be added to both simulations. We focus on the four causes in (Lee *et al.*, 1997a,b) because they are the most studied ones, even if Moyaux *et al.* (2007) found a few more proposed causes in their literature review.

5.4.1 Updating of demand forecasting

Every company places orders based on a forecast of its future demand, and the history of incoming orders is used in this forecast. Unfortunately, the signal of demand information is deformed by forecasting. That is, retailers make a quite accurate forecast because they are in contact with the market, while their suppliers make worse forecasts because they only have their incoming orders to base forecasts on, thus the amplification of demand variability.

This cause of the bullwhip effect does not play in our spreadsheet simulation because companies do not forecast future demand. It is possible to add this by introducing a variable Of representing the forecasted Oi, e.g. $Of^i_w = .5 * Oi^i_w + .5 * Of^i_{w-1}$, and to make the ordering strategy depend on Of when calculating Op and Θp. We may even assume retailers share their forecasts Of by making the Op and Θp of Company i depend on a retailer's Of.

This same modification may be added to MABS.

5.4.2 Order batching

Order batching, and lot sizing in a more general way, deals with the fact that ordered quantities are made discrete in order to profit from economies of scales.

Since the spreadsheet simulator only considers the distribution network of a supply chain, this cause can be implemented by making companies ship only with full truckloads. Since Too in Equation 3, Tob in Equation 6 and $To\Theta$ in Equation 7 are very complicated, we suggest to modify To (at the moment, $To = Too + Tob + To\Theta$ in Equation 8) so that it may only take some specific values.

The MABS may not only implement that, but also its equivalent inside companies by considering production by batches. (Production is not modelled in the spreadsheet model.)

5.4.3 Price fluctuation

When a company proposes a promotion, end-customers and industrial clients buy more products than requested at the moment in order to fill their inventory, which increases demand. Later on, demand decreases when the products in excess are taken from inventory instead of being ordered.

This behaviour may be implemented with both spreadsheet and MABS simulators by introducing a price of the products between every pair of companies,[2] and letting sellers sometimes reduce their price. This also supposes that the ordering strategy takes price into account when calculating Op and Θp in Equations 13 and 14. Since this is "Short" to implement with the spreadsheet simulator, this is also "Short" with MABS (cf. Table 1).

5.4.4 Rationing and shortage gaming

Rationing and shortage gaming deal with the strategic behaviour of companies when demand exceeds supply, e.g. because a machine breakdown reduces the quantity of available products. In this condition, some clients might order more than their actual needs, because they try to have a bigger proportion of available products by "gambling," in order to receive a quantity closer to their actual needs.

In order to implement this cause of the bullwhip effect, every agent may overorder if it does not receive what it

[2] We do not mean prices in general, i.e. with some form of negotiation between buyers and sellers to agree on that price (Subsection 5.2 has just shown why spreadsheet simulation is not appropriate for that), but only in the specific case of promotions causing the bullwhip effect.

ordered four weeks before (because products ordered in Week w are received in Week $w + 4$).

5.5 Capacities

Besides the bullwhip effect, we may also wish to study how capacities constrain decisions.

5.5.1 Transportation capacity

Currently, both simulators may have an infinite quantity of products in inventory or in trucks. It is possible to modify Equation 8 so that an outgoing transport To is not allowed to contain more than a fixed quantity of products, e.g. $To = Too + Tob + To\Theta$ when $Too + Tob + To\Theta < TransportCapacity$, otherwise $To = TransportCapacity$. In other words, transportation capacity can be implemented in a way very similar to transportation by batches in Subsection 5.4.2. Since this modification is "Short" for the spreadsheet approach, it is also "Short" for the MABS.

5.5.2 Inventory holding capacity

It is less easy to implement a capacity for inventories than for transportations, because units arriving to the warehouse have to be stored somewhere. As a consequence, planning has to avoid such an event. In other words, inventory holding capacity is taken into account by the decision making of companies. For example, decisions should be made so that the inventory-order position is never greater than the inventory holding capacity, where the inventory-order position in Week w of company i is the sum of (i) its upcoming orders on Week w $\sum_{v<w}(Op_v^i + \Theta p_v^i - Ti_v^i)$ plus (ii) its on-hand inventory I_{w-1}^i minus (iii) its customer backorders $Ob_w^i + \Theta b_w^i$.

5.6 Implementation time

We split the implementation time in three parts: (a) How difficult is it to learn the tools needed to program the simulator? (b) When we know these tools, how long does it take to code the simulator? (c) How difficult is it to check that the simulation implements its model?

5.6.1 Time to learn the tool

Spreadsheet programs are user-friendly because they are intended to be used by everyone. Moreover, spreadsheet programs have been designed for many years, and many tutorials are available. As a consequence, learning how to use a spreadsheet is relatively short. On the contrary, JACK™ is a language designed as a powerful general-purpose tool for computer scientists and engineers. This power allows building much more complex applications than a spreadsheet, but the price of this power is the difficulty in learning the langugage. Indeed, agent-oriented concepts have to be learnt to use JACK™, and object-oriented concepts are also useful because JACK™ is built upon JAVA. Finally, agent-oriented languages such as JACK™ are young in comparison with spreadsheet programs, and, therefore, tools provided for such agent-based languages are not as user-friendly as they could be, and only a few tutorials are available.

5.6.2 Coding time

Coding time is also an advantage for spreadsheets, because many generic tools are provided with these programs, e.g. to analyze simulation outcomes or to automatize some tasks. In particular, optimization tools, such as the Solver in Excel, are very useful tools to find the best value of some parameters without having to program an algorithm looking for such values through an exhaustive search, a genetic algorithm, a simulated annealing, etc. Concerning again optimization tools as the Solver, we notice that optimization is only possible for a single company in a MABS, and very difficult to carry out for a whole supply chain, while it is easy to optimize for both a single company and a whole supply chain in a spreadsheet simulator due to its non-distributed aspect.

Of course, we can object here that programming a spreadsheet simulator is more difficult than programming a MABS for two reasons. The first reason is that it is not always obvious to find which variables are needed in the spreadsheet. For example, Moyaux *et al.* (2003) have published the equations of the spreadsheet simulation of the QWSG in which backordered orders O were measured as negative inventories. A bug in this simulation was then found, and its correction obliged introducing Ob_w^i so that inventory I_w^i cannot be negative. The second reason is also practical. If you take a large quantity of variables, displaying your simulator is less pretty, and its use is heavier. But if you do not take enough variables, writing the equations defining each variable is an intractable problem. As an illustration, Equation 3 shows several of such what-if rules that have to be gathered in one single entry of a spreadsheet, where each what-if rule is not obvious to define.

On the other hand, we should note that a MABS is longer to implement, because it is basically much more realistic. However, multi-agent frameworks are provided with tools helping agent implementation. In particular, JACK™ allows tracing all messages exchanged between agents and provides a graphical tool to fully design and partially program every agent's behaviour. In spite of that, programming a MABS is longer, and this is the price to pay for obtaining a more realistic simulation.

5.6.3 Validation of the code

Since the progammer of a spreadsheet simulator has a global view on the simulated supply chain, it is easy to understand what occurs in the simulation, and, in particular, to be sure that streams are well implemented (i.e. does material appears or disappears for no reasons?). On the other hand, the distributed nature of a MABS (i.e. agents may be spread over several computers) makes it difficult to construct a global view of the simulation in order to debug

the operation of its agents.

We should note that the outcomes of our spreadsheet simulator are helpful to debug our MABS, because we can simplify our MABS (e.g. by setting the production rate of the Paper Mill and of the Sawmill to infinite) to check if it behaves similarly to our spreadsheet simulator. But when the realism of the simulator increases, the complexity of the spreadsheet program rapidly becomes intractable, and this method becomes impossible. In this case, unit tests, for example with JUnit (`http://www.junit.org/`), are helpful when the MABS runs in a single process.

5.7 Time to run simulations

Since spreadsheet simulators run entirely in the same process, inter-company communications are really fast, no overhead is required by the operating system to manage the multithreading required by several agents when they run on the same CPU, and no communication overhead is spent by the network because all companies run on a single PC. As an insight into the difference of speed, a fifty week simulation takes less than a second on our spreadsheet simulator, while our MABS takes around 20 seconds on our 2 GHz PC (in this second measure, the six company-agents and the ClockAndGui agents all run on the same PC). This difference may be crucial depending on the context. For example, when random functions are used, simulations need to be run a large number of times in order to apply some statisical tools (means, standard deviation...) to analyze simulation outputs.

Moyaux *et al.* (2004b) present another example in which many simulation runs are needed in order to build games analysed with Empirical Game Theory. For that purpose, they run the $3^6 = 729$ combinations of three strategies among the six companies, that is, all companies use the first strategy in the first run of the simulation, next all companies used the first strategy except the LumberRetailer that used the second strategy in the second simulation run, ... and, finally, all companies use the third strategy in the 729^{th} run. The idea of these three strategies was that companies do not collaborate to reduce the bullwhip effect with the first strategy, they collaborate a little with the second strategy (this is the strategy presented in Subsection 3.4), and they collaborate a lot with the third strategy. After each of these 729 simulations over fifty weeks, the individual cost for each company is noted in a $3 \times 3 \times 3 \times 3 \times 3 \times 3$ matrix. This matrix is a game in the normal form in which two things are looked for, namely, (i) all the Nash equilibria with the free software Gambit by McKelvey *et al.* (2007), and (ii) the minimum of the overall supply chain cost. As we can see, the simulation time is important here, because the spreadsheet simulator takes around 6 minutes to run the 729 simulations with a specific market consumption pattern, while $729 * 20 = 14,580$ seconds (4 hours) would have been required by the MABS.

5.8 Publication of the simulation model

It is easy to communicate the code of a spreadsheet simulator with other people, because we only have to replace the name in the spreadsheet by the name of the corresponding variable. For example, in the screenshot in Figure 5, the column L represents the LumberRetailer's inventory in our spreadsheet simulator, so that, L7 contains the value of the initial inventory level, i.e. $I_1^1 = 0$ in this figure, and L8 contains the equation L7+S7-P8 (the content of this entry is displayed above the label of the column C), which is noted $I_w^i = I_{w-1}^i + Ti_w^i - To_w^i$ in Equation 9.

Conversely, the description of our MABS is more difficult, because there is currently no standard to describe the behaviour of agents, even if some tools are widely used, such as SWARM (SwarmWiki, 2006). This is the reason why we first presented a part of the JACK™ framework in Subsection 4.1. But the difficulty in communicating an agent model is also the price to pay for having a more realistic simulation, that is, making a model more realistic also implies making it more complex. However, this price could be lower if a standard existed to describe an MABS, as is the case to describe DEDS with Zeigler (1976)'s DEVS formalism (Discrete Event System Specification).

6 Conclusion

The Quebec Wood Supply Game (QWSG) is a board game similar to the Beer Game designed to teach supply chain dynamics. This game provides an agent model of each company in a supply chain, and a supply chain structure. This paper has presented how the model in the QWSG can be used to implement two simulators in order to study decision making in supply chains. The first simulator strictly implements the model of companies in the QWSG. Since this simulator runs in a spreadsheet, we provide all the equations of this model in this paper. The second simulator is a MultiAgent Based Simulation (MABS) which enhances the first one by detailing the operations in every company. Since this MABS is more realistic, thus more complex, we cannot both detail entirely its implementation and use a spreadsheet program to implement it.

Finally, we have described the pros and cons of each of our two simulators. We started with the drawbacks of spreadsheet simulations by stating that the representation of perturbations is impossible with this simulation type. In addition, modeling interactions (e.g. negotiation) and multi-product scenarios are so long that they are almost impossible with spreadsheet programs. Next, some features are as long to add to the spreadsheet simulation as to the MABS, such as some causes of the bullwhip effect, and modeling transportation and inventory capacities. Finally, we presented some drawbacks of MABS. The first one is the longer time required to (i) learn such a tool, (ii) implement the simulation, (iii) validate the implementation, and (iv) run the simulations. The second drawback of MABS is the difficulty to publish a model longer than

a spreadsheet model would permit due to the increased realism of the MABS approach.

ACKNOWLEDGMENT

We would like to thank the industrial partners of FOR@C, the Research Consortium in e-business in the forest products industry (Université Laval, Quebec City, QC, Canada), and the Natural Sciences and Engineering Research Council of Canada, for supporting this research. We also thank the referees for their valuable suggestions, as well as Eve Levesque for her help on the second simulator.

REFERENCES

Agent Oriented Software Group (2003). JACK Intelligent Agent (TM) User Guide. Technical Report Release 4.1. http://www.agent-software.com/.

Fjeld, D. E. (2001). The wood supply game as an educational application for simulating dynamics in the forest sector. In K. Sjostrom and L.-O. Rask, editors, *Supply Chain Management for Paper and Timber Industries*, pages 241–251, Växjö (Sweden).

FOR@C (2007). Web site of the research consortium in e-business in the forest products industry, Université Laval, Quebec City (PQ, Canada). `http://www.forac.ulaval.ca` (accessed 16 April 2007).

Haartveit, E. Y. and Fjeld, D. E. (2002). Experimenting with industrial dynamics in the forest sector - A Beer Game application. In *Symposium on Systems and Models in Forestry*, Punta de Tralca (Chile).

Khan, A. (2004). Java Excel API 2.3.12. A Java API to read, write and modify Excel spreadsheets `http://www.andykhan.com/` (accessed 2 March 2003).

Kleijnen, J. P. C. (2003). Supply chain simulation tools and techniques: a survey. *International Journal of Simulation & Process Modelling*, **1**(1/2), 507–514.

Kleijnen, J. P. C. and Smits, M. T. (2003). Performance metrics in supply chain management. *Journal of the Operational Research Society*, **55**(5), 507–514.

Lee, H. L., Padmanabhan, V., and Whang, S. (1997a). The bullwhip effect in supply chain. *Sloan Management Review*, **38**(3), 93–102.

Lee, H. L., Padmanabhan, V., and Whang, S. (1997b). Information distortion in a supply chain: The bullwhip effect. *Management Science*, **43**(4), 546–558.

McKelvey, R. D., McLennan, A. M., and Turocy, T. L. (2007). Gambit: Software tools for game theory, version 0.2007.01.30. `http://econweb.tamu.edu/gambit` (accessed 25 April 2007).

Microsoft Corp. (2004). Pack MS-Office 2000. `http://www.microsoft.com/office/` (accessed 29 January 2004).

Moyaux, T., Chaib-draa, B., and D'Amours, S. (2003). Coordination à base de jetons pour réduire l'amplification de la variabilité de la demande dans une chaîne logistique. In *Proc. 5th Int. Industrial Eng. Conf.*, Quebec City (PQ, Canada).

Moyaux, T., Chaib-draa, B., and D'Amours, S. (2004a). An agent simulation model for the québec forest supply chain. In *Proc. 8th Int. Workshop on Cooperative Information Agents*, volume 3191 of *LNCS*, pages 226–241, Erfurt (Germany).

Moyaux, T., Chaib-draa, B., and D'Amours, S. (2004b). Multi-agent simulation of collaborative strategies in a supply chain. In *Proc. 3rd int. joint conf. on Autonomous Agents and MultiAgent Systems (AAMAS)*, New York (New York, USA).

Moyaux, T., Chaib-draa, B., and D'Amours, S. (2007). Information sharing as a coordination mechanism for reducing the bullwhip effect in a supply chain. *IEEE Trans. on Systems, Man, and Cybernetics*, **37**(3), 396–409.

Paquet, S. (2001). *Coordination de plans d'agents: Application à la gestion des ressources d'une frégate*. Master's thesis, Université Laval, Quebec City (Quebec, Canada).

Plamondon, P., Chaib-draa, B., Beaumont, P., and Blodgett, D. (2003). A frigate movement survival agent-based approach. In *Vol. 2774 of Lecture Notes in Artificial Intelligence*, 7th International Conference on Knowledge-Based Intelligent Information & Engineering Systems (KES'2003), pages 683–691, University of Oxford (UK). Springer.

Soucy, M. (2004). *Gestion des ressources en temps réel : Cas des frégates canadiennes*. Master's thesis, Université Laval, Quebec City (Quebec, Canada).

Sterman, J. D. (1989). Modeling managerial behavior: Misperceptions of feedback in a dynamic decision making experiment. *Management Science*, **35**(3), 321–339.

Supply Chain Council (2007). SCOR 8.0 overview booklet. Available from http://www.supply-chain.org/ (accessed March 2007).

SwarmWiki (2006). Web site. `http://wiki.swarm.org/wiki/Main_Page` (accessed December 20, 2006).

Zeigler, B. P. (1976). *Theory of Modelling and Simulation*. John Wiley & Sons, Inc.