A Preventive Multi-agent based Policy for Distributed Hardware Resource Balancing

Abdelhafid Benaouda
Departement of Computer Science
Ferhat Abbas University Setif I
Setif, Algeria
ahbenaouda@univ-setif.dz

Nacera Benaouda
Departement of Computer Science
Ferhat Abbas University Setif I
Setif, Algeria
nacbenaouda@univ-setif.dz

***Abstract*—The success of a business, especially a multi-site extended enterprise, depends on the good management of all its distributed resources. It is difficult for a company to be successful if it does not have a reliable and optimal management of resources by avoiding overstocking of certain resources on a site $Site_m \in E$, and at the same time, the sub-storing of the same resources on another site $Site_p \in E$. In both cases, there is a lack of profit. In this paper, we will try to resolve this situation, by the proposal of an architecture based on the cooperative multi-agent systems paradigm combined with the Contract-Net protocol. We bring in an intelligent agent whose role is to warn in advance and for each item $item_i \in Site_m$, the coming of breakdowns and stock excesses by balancing the level of inter-site availability by a flow of resources of the same $item_i$ by calling on the other E sites whose levels are in over-storage or under-storage.**

Keywords-resource-balancing; multi-agent; flux-resource management

I. INTRODUCTION

In the management of distributed material resources in a multi-site enterprise, always non-optimization issues arise, which generate losses and waste of resources. In the past, this situation was tolerable because the use of the networks was not so vast and communication and interaction means were not in the level they are today. Nowadays, the use of solutions on network platforms, that may optimize the management of these distributed resources, is more than necessary. In this sense the provision of an academic solution is proposed in this paper, based on the multi-agent systems (MASs) paradigm. As an application, a mapping of this proposal on a concrete example of distributed hardware resources was made. This example can be applied to any type of resources [1]. In other words, this paper offers a software solution that can ensure the balancing of various resources of a multi-site enterprise E whose sites are linked by a computer network on which a solution based on middleware such as Java-RMI, Dcom, ORB/Corba or SOAP/WebServices was deployed through Web. Thus, the optimal solution that can bring added value [1, 2] can be found. The sites share their resources according to a cooperation model, therefore, they must interact with each other, and consequently, the implementation of our solution must use one of the previously mentioned middleware.

In the remainder of this article, the generic term *item* instead of resource is used.

II. THE PROBLEM

Let's suppose a company composed of several websites $Site_m$, m=1, nbS. These sites are interconnected by a computer network that may be a LAN, Internet or a VPN. The software means providing the middleware can be Corba, RMI, DCOM or WebServices if corporate sites are connected through the Internet. Let's suppose further, that each site $Site_m$ has a set of N items locally managed. We consider the class C items (see III-A) because they are very numerous. We must find a MAS paradigm based solution which can ensure the balancing of items between the different company sites, in other words, a solution that avoids having items in overstocking or in sub-storage in order to prevent and avoid abnormal storage situations. Figure 1 illustrates this situation.

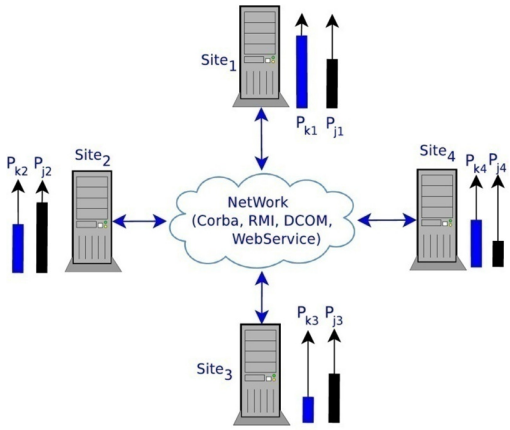

Fig. 1. The problem

On the site $Site_1$, quantitatively P_{k1} is over-stocking, while on the $Site_3$, P_{k3} is understaffed storage for the same item k. For another item j, there is an approximate balance between the two quantities of items P_{j1} and P_{j3} on the sites $Site_1$ and $Site_3$. We make the same observation between the situations of the two sites $Site_2$ and $Site_4$ concerning items j and k. By extending this

Corresponding author: Abdelhafid Benaouda

observation to the sites ($Site_k$, k=1, nbS) with nbI on each site, the proposed solution consists of watching dynamically at every moment T_c the stock levels of all items on all the system's sites, to alert policymakers and managers of an overstocking or sub stocking situation, and provide a solution that helps correcting this imbalance.

III. RESOURCE MANAGEMENT IN MULTI-SITE ENVIRONMENT

In this paper, we discuss items that are storable in a multi-site environment. These items exist within the company, and are intended to be sold or reserved for internal use, such as, for example, spare parts, pharmaceuticals, consumables (paper, cartridges, etc.). The modeling and automation of such items on a local site is currently a very easy task. Many dedicated solutions, marketed and even free, have been currently proposed. However, solutions for multi-site companies are stock-keeping applications only, and not stock management applications, because item inventory management for a company whose sites are interconnected via a local network or Internet, requires the study, modelling, and validation of other criteria (communication, interaction, cooperation, negotiation, etc.). The "zero inventory" is a wrong approach because stock is necessary but it should not exceed a limit, because it negatively affects the business by including unnecessary costs (insurance, local, security, aging, etc.). On the other hand, sub-storage too can lead to out of stock situations. Stock shortage can, in some cases, cause the production chain alteration by stopping the production machine. Good management means managing the stock of these items, in sufficient quantity, but above all in an optimal way, to meet the needs for the proper functioning of the company. This is equivalent to finding an optimal stock S^* for the entire multi-site enterprise, in the sense of cost and availability over a period T, so as not to expose it to an abnormal situation. In order to correct this way of storing, locally and statically, the current trend is to make "an item flow management" between different sites and to ensure an intra-site and inter-site balancing of these items in order to avoid over or under-storages.

A. Item Classification

An item is specific to the class to which it belongs, and this strongly influences the adopted replenishment policy. On the basis of this classification we can answer to questions such as:

- What to resupply?
- When resupply?
- How much to resupply?
- And above all, what partner site to resupply, in the case of deficit. And, which site, surplus items must be passed on?

We will adapt this classification to item management:

- Class A: is the class of slow-rotating or slow-moving items.
- Class B: is the class of medium-rotating items.
- Class C: is the class of fast-rotating or fast-moving items.

B. The Extended Enterprise Case

Cited replenishment policies show the strategy with regard to each item, after having previously studied the demand, the cost, the necessary time, etc., which they can be applied especially to an autonomous site. A multi-site enterprise's replenishment policy depends on the needs of all the interconnected sites. The class C items, for instance, can be found in all sites, but items supposed to belong to B and A may be available in specific sites deemed important by the extended enterprise managers. The management principle in this case is to maintain a balancing: first, in terms of availability, and second, in terms of cost. This balancing must be between all sites and for all items that are needed by each of these sites. To achieve this, we must design and implement appropriate solutions, to prevent and eventually avoid an availability imbalance of these items in a distributed multi-site environment with the constraint that each site supplies and demands different items. We know that, each item follows different criteria of cost, importance in the case of failure, etc.

C. Distribution of Items

In this paper, focus is given on class C items, because even if they are not expensive, their lack can cause system shutdown. The item classification is dynamic, a $Class_i$ item may end up in another class $Class_j$, lower or higher. This will depend on this item consumption rate, and its importance.

IV. RELATED WORKS

There are not many works dealing with this issue as a "balancing of material resources" in a distributed environment, but they exist, as "load balancing". A load is the set of processes (Unix process, threads), running on a processor, on a machine or on a server at a time T. Many algorithms have been proposed and implemented on load balancing. But the projection of this work on the distributed management of material resources is limited. The problem came up with big-data, high availability in cloud-computing [1, 4-8] and in distributed systems in general because disk quotas distributed in clusters hosting data can be considered as well and optimally managed hardware resources[9, 10]. The problem of material resource balancing was initiated in [2, 3], within the framework of the European project PROTEUS, with application to the distributed management of the spare part in a multi-site environment, which can avoid over-storage in a S_m site, to the detriment of an under-storage at another S_k site. In our opinion, the management of material resources in a multi-site platform is at the intersection of the following three issues :

- The projection of algorithms developed in the load balancing field.
- Inventory management across an extended enterprise.
- Distributed management of information flows in a multi-site cooperative platform.

V. PREVENTION POLICY

A. Mathematical Formulation

The consumption law of an $item_k$ on a $\Delta t = T_C - T_0$ interval with $T_0 = 0$, on a $Site_m$, is defined by :

$$P_i = f(T_i),\ i = 1,\ n \quad (1)$$

where P_i is the consumption of the $item_k$ at time T_C.

The law (1) obeys many criteria, item types (A, B or C), etc. For our part, we recall that we focus on the class C items with very high consumption. This class of items obeys a consumption law, approximated by a least squares line. But if this consumption law changes, the conceptual proposition that we present in this paper will remain valid. We wish to approximate the real consumption, by a consumption law at the moment T_c, in order to be able later to detect stock shortages, at the moment T_{os}, before the next replenish at time T_r.

$$P_i = a_k T_i + b_k,\ i = 1,n\ and\ k = 1, nbS \quad (2)$$

This should be done automatically for all common and existing items on each site.

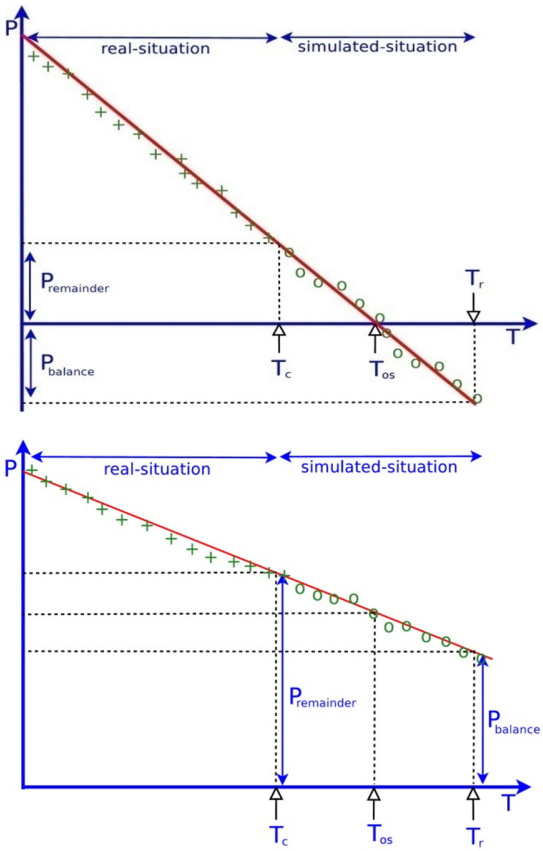

Fig. 2. Storage Balance between two sites for $item_k$: (a) under storage, (b) over storage

To do this, we use least squares to find the coefficients a_k and b_k. It is sufficient then to solve the system:

$$\begin{cases} \frac{\delta \sum_{i=1}^{i=n}(a_k T_i + b_k - P_i)^2}{\delta a} = 0 \\ \frac{\delta \sum_{i=1}^{i=n}(a_k T_i + b_k - P_i)^2}{\delta b} = 0 \end{cases} \quad (3)$$

For a single $item_k$, there are nbS systems of linear equations, with unknowns a_k and b_k to be solved. By considering:

$$z_{1k} = \sum_{i=1}^{i=n} P_i \quad , \quad z_{2k} = \sum_{i=1}^{i=n} P_i T_i \quad ,$$
$$z_{3k} = \sum_{i=1}^{i=n} T_i \quad , \quad z_{4k} = \sum_{i=1}^{i=n} {T_i}^2$$

which give:

$$\begin{cases} z_{1k} = z_{3k} a_k + n b_k \\ z_{2k} = z_{4k} a_k + z_{3k} b_k \end{cases} \quad (4)$$

This is equivalent to solving $nbA * nbS$ systems, similar to the system in (4). Note that the number of units n available by $item_k$ can vary on the same $Site_m$ or on different sites. The solution for $item_k$ on a $Site_m$ is therefore immediate:

$$\begin{cases} a_k = \frac{z_{1k} z_{3k} - n z_{2k}}{{z_{3k}}^2 - n z_{4k}} \\ b_k = \frac{z_{3k} z_{4k} - z_{4k} z_{1k}}{{z_{3k}}^2 - n z_{4k}} \end{cases} \quad (5)$$

Knowing a_k and b_k, the stock shift for the item k at the moment T_{os} on a $Site_m$ will occur at :

$$T_k = -b_k / a_k$$

Applying the *Sm* and *Sp* sites, stock-outs will occur at:

$$\begin{cases} T_{os_{km}} = \frac{-b_{km}}{a_{km}} \quad for\ Site_m \\ T_{os_{kp}} = \frac{-b_{kp}}{a_{kp}} \quad for\ Site_p \end{cases} \quad (6)$$

and

$$\begin{cases} P_{km} = b_{km} \quad for\ Site_m \\ P_{kp} = b_{kp} \quad for\ Site_p \end{cases} \quad (7)$$

if the condition (8) is satisfied:

$$\begin{cases} if \quad T_{os_{km}} \ll T_{os_{kp}} \ at \ \ T_{os} \\ \qquad\qquad and \\ if \quad P_{kp} \gg P_{km} \ at \ \ T_c \end{cases} \quad (8)$$

A balancing for $item_k$ between sites S_m and S_p is necessary, because there will be a stock shortage in the $Site_m$ and a useless stock in the $Site_p$ (see Figure 1), which is a shortfall for the company.

B. Transfer Mechanism

The transfer mechanism (Figure 3) that we are going to use is essentially based on the role of a software agent called prevention agent AGP_k (see *VII1*), associated with each site $site_k$. AGP_k allows controlling the items by managing their state before, concretely, making their transfer (Figure 3).

VI. MAS MODELING

A. Agent, MAS and Cooperation

We consider that the agent paradigm and *MAS* is known now. However, it should be remembered that there is no unified definition of the term "agent" in *MAS* community. However, everyone agrees that the autonomy concept is central to the issue of agents. For our part, we consider the consistent

academic definition of this notion given in [11, 12]. For our case study, we define an agent as a software entity, located in an environment, with an autonomous behavior enabling it to reach, by interacting with this cooperative environment, the objectives assigned to it during its creation. Cooperation is the form of coordination between non-antagonistic agents. Conversely, negotiation corresponds to coordination in competitive universes or between "selfish" or competitive agents. Typically, to successfully cooperate, each agent must maintain a model of other agents, and also develop a model of future interactions (planning). The fact that a $Site_m$ agrees to troubleshoot another $Site_k$ is a form of cooperation. This can be done, of course, after a negotiation phase between the sites while maintaining their respective interests. Moreover, a MAS is a set of agents that respond to the architecture of a system and its dynamics, based on an organizational model and on a model of interaction between the agents of this system. These agents are activated for the emergence of the desired solution.

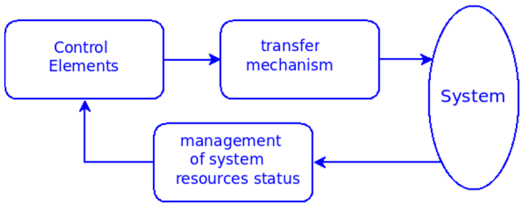

Fig. 3. Transfer mechanism

In this paper, we take note of the keywords: organization, autonomy, interaction, cooperation, and objective which, in our view, constitute the principal characteristics that the agent-MAS pair must have, and which constitute the reason for the dynamics of the system in order to fulfil the desired objective. Unlike the top-down approach, known as static conventional approach, modeling the functioning of a complex problem using MAS is starting from simple distributed tasks with local objectives in order to bring out the overall solution of the posed problem. In our case, the local objective is the optimal availability of items in quantity (neither under-storage nor over-storage) and cost. The goal of the entire system is the balancing of these items across all the enterprise's sites. Using MAS in modeling is also to look for an adequate organization model that must be associated with an interaction model in order to respond as much as possible to the proper functioning and dynamics of the system. We consider that there is not yet a generic model that can satisfy all the criteria of organization and functioning of any problem.

B. Agent-oriented Architecture Model

The agent architecture is the product of the transition from the objective component to the projective component. Indeed, in the case of oriented object programming, the object is essentially a passive component offering services and using other objects to realize its functionalities. The object architecture is therefore an extension of the architecture in calls and returns. The agent, on the other hand, intelligently uses other agents to achieve his objectives. It performs this by establishing dialogues, negotiating and exchanging information with them. We involve the following five agents (Figure 4):

- AGI, Information AGent.
- *AGP*, Prevention AGent.
- *AGS*, AGent Site Supervisor.
- *CAG*, Coordinator of the Group Acquaintances.
- *CIG*, Inter-Group Coordinator.

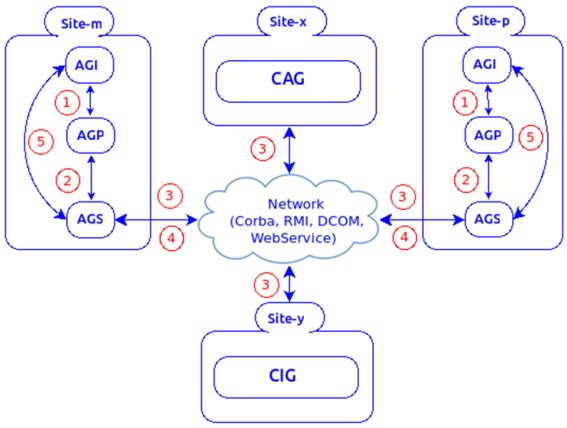

Fig. 4. Agent-oriented architecture

VII. DEPLOYMENT SCENARIO

Before describing the roles and behaviors of the software agents deployed to make the balancing in quantity of our system different items, we first briefly describe the process of interaction between these agents (Figure 4):

- On each site ($Site_m$,m=1, nbS), there is an agent AGP_m associated with it, which watches periodically. After evaluation of the state of each $item_i$, i=1, nbI it alerts AGS_m in case of under-storage or over-storage.
- AGS_m informs the *CAG* for items that are in under or over storage.
- The *CAG* launches a call for tenders to sites k which are in under storage for an item j which is overstocked at site m ($m \neq k$). If the answer is negative, an alert is sent to the *CIG*.
- If there is a positive answer from the AGS_p, a negotiation starts between the AGS_m and the AGS_p on quantities, deadlines, etc.
- If there is confirmation of the incoming or outgoing, the AGS_m, AGS_p orders respectively the AGI_m, AGI_p to update the respective items information systems.

VIII. ROLES OF SYSTEM AGENTS

The roles of the agents involved in our architecture are:

A. Prevention Agent (AGP)

AGP is an intelligent agent that has an intelligence capacity that allows it to take decisions based on its Knowledge Base (BC). It is based on algorithms that calculate previous consumption and other availability criteria. This agent has the ability to predict cases of sub or over-stock of item quantities, based on the T_{prev} prevention times calculated for each item. It

sends emergency messages to the supervising agent indicating, for a type of item, whether there will be an over or under-storage. This agent therefore consults the database in a continuous and periodic manner and looks for cases that may cause abnormal situations, in order to prevent and send alert messages to the supervisor. Our study gives more importance to the prevention aspect, and thus to the script of the agent dedicated to this role, because it ensures an optimal management of the items in a cooperative framework between all the sites of the extended enterprise. Our study gives more importance to the prevention aspect and consequently to the script of the agent dedicated to this role, because it ensures an optimal management of the items in a cooperative framework between all the sites of the extended enterprise. The script of this agent is presented in Figure 5.

```
1:  BeginScript
2:  // I am the agent AGP_m
3:  // -1 for input, 1 for output
4:  // we are at Current Time T_c
5:  // the quantity, in units, is denoted by P
6:  while (true)
7:     foreach period_dT do
8:        foreach piece_k do
9:           //determine T_r, P_r ; before re-supply
10:          //determine T_os, P_os ; at out-stock time
11:          //determine P_c, available quantity of item_k at T_c
12:          //determine Pr, remaining quantity of item_k at T_r
13:          //determine Pb_k, balance quantity of item_k at T_os
14:          if Pbk < 0 then
15:             // item_k is in state of underStorage
16:             // I want to avoid the "out-of-stock" before T_os
17:             send(AGS_m, item_k −1, P_k T_os)
18:          else
19:             // determine Ps_k, strategic quantity of item_k
20:             // determine Po_k, overStorage quantity of item_k
21:             // Ps_k should be greater than Po_k
22:             if Pb_k >> Ps_k then
23:                // item_k is in state of over-storage
24:                // I want to offer at T_c
25:                send(AGS_m, item_k 1, P_k T_c)
26:             endif
27:          endif
28:          // there is a item_k-balancing
29:       enddo
30:    enddo
31: enddo
32: endScript
```

Fig. 5. Script of AGP_m

B. Site Supervisor Agent (AGS)

AGS is the supervising agent. It is the master agent associated with a site, its role is to manage and control the system operation. It describes the overall objectives of the system and observes its state. The supervising agent communicates the incoming and outgoing quantities of items. It also launches requests for item acquisition or offering. It receives/sends messages from/to other sites. With this approach, it acts as an interface relative to other sites. Its script is can be seen in Figure 6.

C. Information Agent (AGI)

AGI is a reactive agent that has no intelligence, its role is stock management, in the traditional sense, with actions such as creating, consulting, and updating item database. This agent receives the order from the *AGS* before any operation.

```
1:  beginScript
2:  // I am the agent AGS_m of Site_m
3:  // I am the interlocutor of Site_m
4:  // -1 for input, 1 for output
5:  // Q = P of itemk at T_c
6:  while (true) do
7:     if receive(AGS_p, item_k −1, Q)
8:     then
9:        negotiate (AGS_p, item_k 1, Q)
10:    endif
11:    if receive (AGS_p, item_k 1, Q)
12:    then
13:       negotiate (AGS_p, item_k −1, Q)
14:    endif
15:    if receive(AGP_m, item_l −1, Q)
16:    then
17:       send(CAG_m, item_l −1, Q)
18:    endif
19:    if receive(AGP_m, item_l 1, Q)
20:    then
21:       send(CAGm, iteml, −1, Q)
22:    endif
23:  enddo
24: endScript
```

Fig. 6. Script of AGS_m

D. Coordinator of the Group Acquaintances Agent (CAG)

CAG's role is associated with a group of sites $Site_k$, k=1, *Gs and Gs subset E* on the basis of a criterion, e.g. the geographical one. The *CAG* ensures the coordination and reception of the needs, in quantity of items, of the different sites of the group. It classifies the needs of the sites concerned by items that are under or over storaged, optimizes these costs, according to the auction algorithm and communicates these needs to the appropriate AGS_k, k=1, *nbS*. Its behavior is described by the script in Figure 7.

```
1:  beginScript
2:  // I am the agent CAG_x of group_x
3:  //-1 for input, 1 for output
4:  // Q = P of item_k at T_c
5:    while ( true ) do
6:       if receive (AGS_m, item_k −1, Q) OR
7:          receive (AGSm, itemk, 1, Q)
8:       then
9:          //Send to all accointances of my group_x
10:            if // receive a proposition
11:               // from AGS_i of group_x
12:            then
13:               // evaluate the best proposition AGS_i and
14:               // link (AGS_k, AGS_i item_k Q)
15:         endif
16:         else
17:            // send this proposition to CIG
18:            // ...
19:         endif
20:   enddo
21: endScript
```

Fig. 7. Script of CAG_x

E. Inter-Group Coordinator Agent (CIG)

CIG has the same prerogatives as the *CAG*, but at the top level. It intervenes only in very exceptional cases, e.g. in the

case where an $item_i$ belonging to $Site_m$ is in global state of under or over-storage. In this case, it is necessary to negotiate with a $Site_k$ outside the group. As a result, the behavior of the *CIG* is identical to that of the *CAG* but in intergroup coordination.

We notice that the agent-oriented architecture we propose has cognitive agents and reactive agents. The prevention agent is a cognitive agent, having the ability to reason on the basis of its past, regarding past consumption and all other information in its knowledge base. Agents belonging to the same agent group recognize each other through their intercommunicating supervisors.

IX. ORGANIZATION MODEL

The problem as it stands does not obey an already existing organizational model. We have developed our own model, based on our needs, called *AGIRAT* (Figure 4, Figure 8) which is based on the following:

1) The agent is the basic conceptual entity of our organization. Each moment a site is represented, in its environment, by the supervising agent. And the site itself can have a MAS or another internal structuring. As a result, an agent is :

- Autonomous.
- In intra-site interaction with the prevention and inventory management agents and in extra-site interaction with the other supervisory agents of the system. Knowing that the agents are cooperative, they must communicate in asynchronous connected mode with answer times Δ_t which must be determined by this same supervisor.
- Distributed.
- Has a goal.
- Our prevention agent is a cognitive agent because it has a form of intelligence, which allows it to predict its future and act in the present on the basis of its past.

2) Group: represents the organization of agents in a group according to a specific criterion. As a result, an agent can belong to several groups (Figure 8).

3) Interaction: The whole system is based on the interaction of one entity with the others. Without interaction, the system is static.

4) Role: Each agent has a role in relation to its existence, but performs one or more tasks at the time $T_{current}$.

5) Authority (or privilege): authority is the rules governing the organization and the interest of the group. On this basis, a holistic strategy can be applied.

6) The task which is the subdivision of the role into task sequences (sequentially or in parallel) is the atomic element in relation to the conceptual MASs. At implementation, a task can represent a script running on one or more processes in the IT sense (Thread or Unix Process)

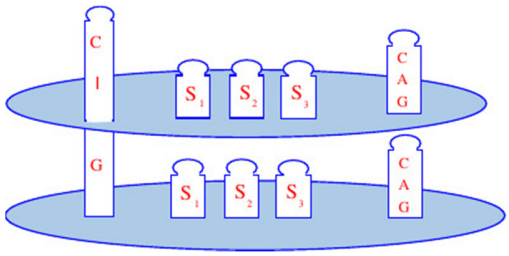

Fig. 8. Organization model

X. EXPERIMENTATION AND RESULTS

Let $P_i(T_i)$ be the consumption of nbU_m units of a $item_{km}$ in the site S_m and nbU_p units of a $item_{kp}$ in the site S_p to consume before a replenishment time T_r. Let's suppose the following initial conditions:

- Items are in class C
- $T_r = 80$ days
- Initial number of items $nbU_m = nbU_p = 5000$ *units*
- Current time = $T_c = 70$
- Delivery times are very small

The results of $item_{km}$ and $item_{kp}$ consumption, respectively on the sites S_m and S_p, yielded the following results:

1) $item_{km}$ consumption:

$$P_{km} = -65.30833T_{km} + 4869.7373 \quad (9)$$

with:

$$\begin{cases} a_{km} = -65.30833 \\ b_{km} = 4869.7373 \end{cases} \quad (10)$$

with Asymptotic Standard Error:

$$\begin{cases} \Delta(a_{km}) = \pm 0.3437\ (0.5263\%) \\ \Delta(b_{km}) = \pm 13.74\ (0.2822\%) \end{cases} \quad (11)$$

We deduce for $item_{km}$:

$$\begin{cases} Tos = 74.56534 \approx 75 \\ P_{remainder} = 298.1543 \approx 298 \\ T_{remainder} = \approx 5\ days \end{cases} \quad (12)$$

2) $item_{kp}$ consumption:

$$P_{kp} = -47.335766 * T_{kp} + 4904.9697 \quad (13)$$

with

$$\begin{cases} a_{kp} = -47.335766 \\ b_{kp} = 4904.9697 \end{cases} \quad (14)$$

with Asymptotic Standard Error:

$$\begin{cases} \Delta(a_{kp}) = \pm 0.2306\ (0.4872\%) \\ \Delta(b_{kp}) = \pm 9.22\ (0.188\%) \end{cases} \quad (15)$$

We deduce for $item_{kp}$:

$$\begin{cases} Tos = 103.6208 \approx 104 \\ P_{remainder} = 1591.46608 \approx 1591 \\ T_{remainder} = \approx 34\ days \end{cases}$$

3) Balancing between $item_{jm}$ and $item_{kp}$:

By comparing the results of (12) and (16), we notice that the $T_{remainder}$ relative to $item_{km}$ is very close, while To_{kp} on $item_{kp}$ is very far (Figure 9). In addition, $item_{km}$ is in under-storage and $item_{kp}$ is in over-storage, i.e. $P_{kp} >> P_{km}$. These quantities of under and over-storage are easily evaluable before the next replenish at $T_r = 80$:

$$P_{km}(80) = -354.9291$$

$$P_{kp}(80) = 1118.10842$$

Therefore, the $Site_p$ in over-storage, can offer the $Site_m$ which is in under-storage situation, at least the deficit that is in this case easily quantifiable. This will be done in the same way for $item_j$ on $Site_m$ and $Site_p$. The $Site_m$ can offer the $Site_p$ the quantity it needs (Figure 10).

XI. GENERALIZATION TO N SITES

By generalizing this process to other items and other sites, and using scripts in Figures 5-7, we are able to perform decision-making followed by the transfer via the network platform, (Figure 4) using a software middleware, a global balancing of the system emerges and is maintained over time by determining the successive differences, i.e. the scales at each time period *balP at* T_{os} and *balP at* T_r, between the same $item_i$ between two different sites (Figure 11). Interactions between the system agents are done by the quoted middleware (Figure 4). Messages sent between these agents and generating these interactions are characterized by the requests send() and receive() such as in the scripts in Figures 5 and 7. The subscript negociate() (Figure 6) is in fact a succession of asynchronous bi-directional messages between each two interlocutors AGS_m and AGS_p, but these messages are synchronous with other system interlocutors. This must be done in order to agree on how to make this contract work. But the last word comes from human decision-makers that are helped by these two facilitating agents installed on sites $Site_m$ and $Site_p$. The system is balanced because it is generalized for all the system item s($item_k$, k=1, nbI_m) and on each site ($Site_m$,M=1, nbS).

XII. COMPARISON AND DISCUSSION

As noted above, resource management in a multi-site environment depends on many research axes (load-balancing, material resource management with all the supply chain issues, management of information flows in an e-environment, etc.). In this work, a material resource can be a spare part to be acquired at the right time for preventive maintenance purposes, or a quota on disk in a multi-user system or even a virtual disk in cloud-computing on a data-center, and particularly in IaaS (Infrastructure as a Service).

Compared to the cited works, our contribution defines this notion of distributed material resources and provides a "quantifiable and preventive" solution to this situation in a distributed multi-site environment. This point was not taken into account in the related work.

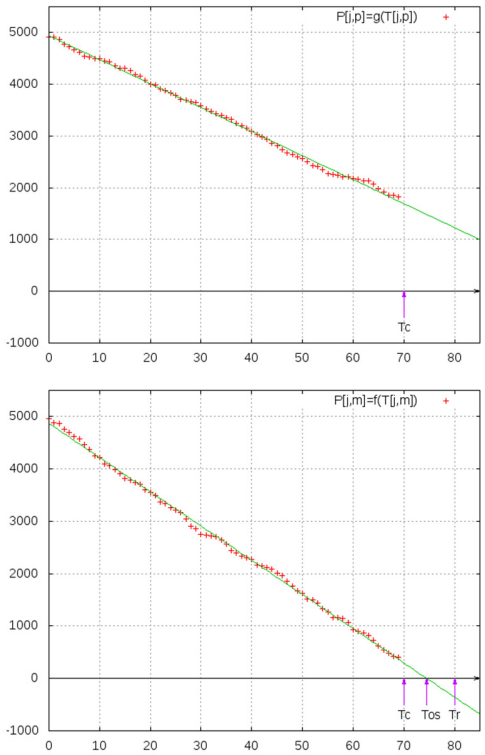

Fig. 9. Unbalancing of $item_j$ on $Site_m$ and $Site_p$: (a) under-storage of $item_j$ on $Site_m$, *(b)* over-*storage of* $item_j$ *on* $Site_p$

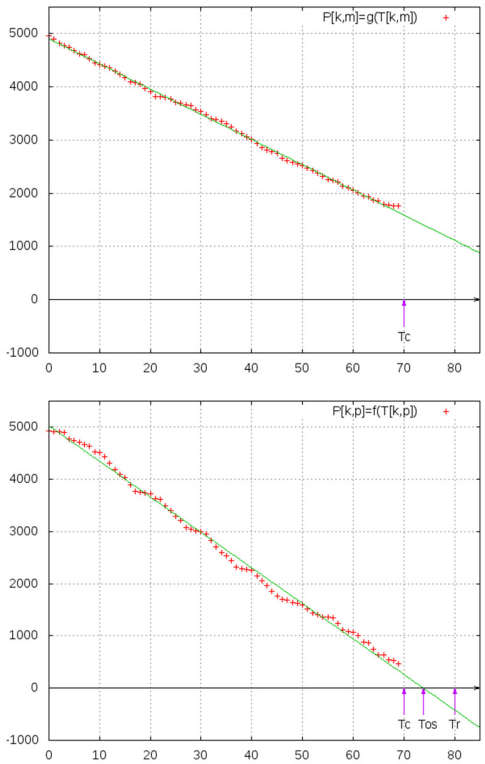

Fig. 10. Unbalancing of $item_k$ on $Site_m$ and $Site_p$: under-storage of $item_k$ on $Site_m$, (b) over-*storage of* $item_k$ *on* $Site_p$

(a)

(b)

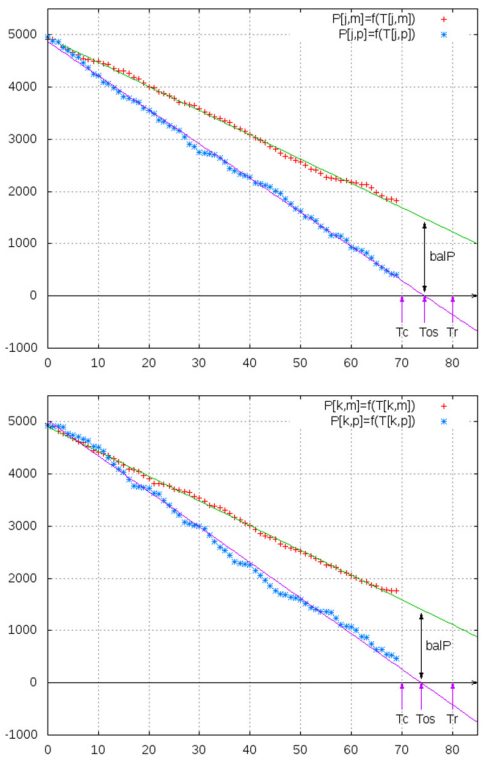

Fig. 11. $P_{balance}$ between two different items on two different sites: (a) $P_{balance}$ between $Site_m$ and $Site_p$ for $item_j$, (b) $P_{balance}$ between $Site_m$ and $Site_p$ for $item_k$

XIII. CONCLUSION

In this paper, an interaction model, dedicated Multi-Agent System (MAS) was proposed, which responds to an organizational model based on the *AGIRAT* (Agent, Group, Interaction, Authority and Task) principles, and was applied to the management of material resources in a cooperative framework. A *MAS* architecture was established, which is internal to the site, and which provides, among other things, a dedicated cognitive agent, whose role is to guard, and even prevent, possible cases of under or over-storage of items, based on past consumption. The global *MAS* architecture of all the sites composing the system interacts with the internal *MAS* of a site via a facilitating agent, in this same cooperative framework, ensuring the emergence of a "resource balancing", and allowing quantification the of the availability of items. This solution prevents sites from exceeding the storage threshold. The implementation of the proposed architecture, requires the use of a dedicated middleware software. Our approach has been validated in a distributed environment and has yielded satisfactory results.

REFERENCES

[1] K. V. Kavitha, V. V. Suthan, "Dynamic load balancing in cloud based multimedia system with genetic algorithm", 2016 International Conference on Inventive Computation Technologies, Coimbatore, India, August 26-27, 2016

[2] A. Benaouda, N. Zerhouni, C. Varnier, "Spare part management for e-maintenance platform", in: IEEE :Mechatronics and Robotics, Vol. 3, pp 1152-1157, IEEE, 2004

[3] A. Benaouda, N. Zerhouni, C. Varnier, "Une approche multi-agents coopératifs pour la gestion des ressources matérielles dans un contexte multi-sites de e-manufacturing", 6e Conference Francophone de MOdelisation et SIMulation, Rabat, Morocco, April 3-5, 2006 (in French)

[4] X. Nan, Y. He, L. Guan, "Optimal resource allocation for multimedia cloud based on queuing model", 2011 IEEE 13th International Workshop on Multimedia Signal Processing, Hangzhou, China, October 17-19, 2011

[5] N. Benmoussa, M. Fakhouri Amr, S. Ahriz, K. Mansouri, E. Illoussamen, "Outlining a model of an intelligent decision support system based on multiagents", Engineering, Technology & Applied Science Research, Vol. 8, No. 3, pp. 2937–2942, 2018

[6] M. A. Patel, R. Mehta, "A comparative study of heuristicload balancing in cloud environment", International Journal of Advance Engineering and Research Development,Vol. 2, No. 1, pp. 2348-4470, 2015

[7] P. Naik, S. Agrawal, S. Murthy, "A survey on various task scheduling algorithms toward load balancing in public cloud", American Journal of Applied Mathematics, Vol. 3, No. 1-2, pp. 14-17, 2014

[8] S. F. Issawi, A. Al Halees, M. Radi, "An efficient adaptive load balancing algorithm for cloud computing underbursty workloads", Engineering, Technology & Applied ScienceResearch, Vol. 5, No. 3, pp. 795–800, 2015

[9] L. T. Yang, M. Guo, High-performance computing: Paradigm and infrastructure, John Wiley and Sons, 2006

[10] W. Zhu, C. Luo, J. Wang, S. Li, "Multimedia cloud computing : An emerging technology for providing multimedia services and applications", IEEE Signal Processing Magazine, Vol. 28, No. 3, pp. 59–69, 2011

[11] J. Ferber, Les Systemes Multi-agents: Vers une intelligence collective, InterEditions, 1995 (in French)

[12] M. Wooldridge, Intelligent agents, a modern approach to distributed artficial intelligence, MIT Press, 2001