A Formal Approach for a Self Organizing Protocol Inspired by Bacteria Colonies: Production System Application

Hakima Mellah[1], Salima Hassas[2], Habiba Drias[3], A. Raiah[4], and A. Tiguemoumine[4]

[1] Research Center in Scientific and Technical Information, Cerist, Algeria
hmellah@mail.cerist.dz
[2] Liesp, Lyon1 university, France
Hassas@bat710.univ-lyon1.fr
[3] USTHB, Algeria
hdrias@usthb.dz
[4] Blida Univeristy, ALgeria

Abstract. Any dysfunction in production system (PS) is likely to be very expensive; so modelling by Multi Agent Systems (MAS) makes the production system (PS) possible to have aspects of robustness, reactivity and flexibility, which allow the PS control to be powerful and to react to all the risks being able to occur. In order to have a fault-tolerant PS, we propose when and how to recourse to a self organizing protocol making the MAS capable of changing its communication structure or organization, and thus reorganizing itself without any external intervention.

Keywords: Production system, self organization, MAS protocol, failure detection.

1 Introduction

Research of productivity remains the major objective of the industrial world. This objective is in permanent evolution and it requires studies and brings increasingly complex solutions in real time piloting of production systems (PS). The critical and important problem to solve is fault tolerance. The control system must provide very powerful mechanisms for fault tolerance in order to ensure the continuous operations of PS that are detection, prevention and correction, etc. A production system is a set of resources realizing a productive activity. The production is the transformation of resources (machines and materials) leading to the creation of goods or services [1]. The transformation is done by a succession of operations (tasks) that use resources (machines and operators), and modify the raw materials or components that enter into the PS in order to create outgoing finished products of this system and assigned to be consumed by customers. Changes may relate to the product form, its structure, its appearance, and so on. The transformation undergone by products, brings them an added value. The resources belonging to the PS mobilized for achieving the production activity can be machines, operators, energy, information, tools, etc. The most important characteristics of a PS are flexibility, reactivity, robustness [2]. The systems theory [3] suggests decomposition of production systems into two subsystems:

(i) Information and decision subsystems that include a control portion which represents the intelligent part of the system, (ii)physical production subsystem consisting of a flows part transforming or assembling materials or entities , and a physical part representing all means necessary to carry out operations. The case study we considered is a production system (PS). We propose a MAS approach where interactions are based on a self organizing protocol that has the following features:

– It assures a decentralised control so each agent can take decisions regarding the interactions with the neighbourhood,
– It allows new communication ways when dysfunctions appear within the MAS or within the informational network. This feature gives robustness to the MAS and allows the PS to be fault tolerant.

2 Multi Agents Approaches for PS

MAS offers a new approach for modelling production systems. Instead of modelling distributed systems with programs exchanging data and commands, agent technology allows the creation of autonomous agents that communicate among themselves, negotiate sub objectives and coordinate intentions in order to achieve the objectives appropriate to the system[6]. In this context several approaches have been cited for modelling production system. In [7], a MAS platform is built for driving workshops. Different types of agents are proposed (resources agents, cell agent and product agent), they represent physical features, virtual islands or operation's sequences. A supervisor agent's role is to monitor the production process, it is assisted by a meta object agent to include new agents in the system. The principal goal of [10] does not deal with self organization. In [8] a MAS architecture in which each agent supervises a production resource is proposed. A supervisor agent is responsible for controlling the entire PS by communicating with agents supervising a production resource. The latter makes decisions about the production rules to be applied to the resources they supervise. However, the agent supervisor can intervene and tell each agent what rule to apply to achieve the overall objectives of the PS. Among the limitations of both approaches, we can quote:

– Possible saturation of the supervisor agent (too many messages from other agents of the system)can suddenly happen
– A failure within the supervisor agent causing stopping of system functioning.
– Communication cost can be very high (a message can take an important time to reach its destination, knowing that this message will go through the MAS leader agent).

In [9] a self organizing approach for manufacturing control is proposed, with our respect to the work of Bussman elaborated in this area, the dynamic routing is proposed to avoid possible congestion and jams on a machine. Agent' communication is assured by invitation and there is no indication on the system when the invitation is not successful or when an agent fails.

3 Bacteria Colony Self Organization

Traditionally the bacteria colonies push with a high level of nutritive elements [14]. A pattern in a bacteria colony is in fact an organizational structure making possible for the bacterium within its colony to communicate(bacterium-bacterium interactions), in order to fight against the adverse conditions of its environment [12]. A set of biological primitives characterizing the bacteria life has been translated to a communication model [13] based on a set of software processes considered as a MAS interaction protocol. The model takes into account only the first five primitives which seem for us to be the most basic and important within the colony, those are: the positioning, the checking, election, routing, and grouping. Each primitive will be described below, using mathematical symbols.

4 The Proposed Multi Agent system

Three types of agents are proposed: Resource Agent (RA), User Agent (UA), Interface Agent (IA):

– Resource agents for piloting a set of heterogeneous machines. They communicate with each other to achieve the production plan consisting in a set of tasks.
– User Agent or the operator. Its role is to develop a production plan of the entire production system. Its knowledge concerns resources, tasks;
– Interface Agent, acts as an interface of PS (intermediary between the RA and the operator); its knowledge are about tasks and RA.

4.1 System Functioning

The user agent develops the production plan (scheduling) which will be determined by the allocation of resources to tasks in order to achieve a product, the plan provides us with information on the task (identifier, priority, duration and precedence link). The user agent sends the plan to the interface agent that has the role of intermediary between the user agent and the various RA agents of the system. The interface agent sends to resource agents tasks to be done, each RA assures local management of its resource by integrating the piloting functions [4] in real time, namely: scheduling, execution and monitoring, each agent can perform these functions autonomously.

- **Scheduling**. RA provides the resource execution plan by respecting tasks priorities, precedence links as the execution time of tasks.
- **Execution.** RA executes the tasks affected to its resource by referring to the execution plan.
- **Monitoring.** RA assures the monitoring, detection, diagnosis, treatment and recovery in a failure case. The latter can be done by transferring tasks to other agents when a problem is detected within the system. In our approach the recovery is done by the recourse to the self organizing protocol that we propose above.

At the beginning of the system, the different RA and IA execute the main important processes of the self organizing protocol those are: positioning process, and Checking process. They allow the detection of a dysfunction within MAS. The rest of processes will be executed by RA for the treatment of failures. Figure1 shows the collaboration process between the system's agents while executing the protocol processes.

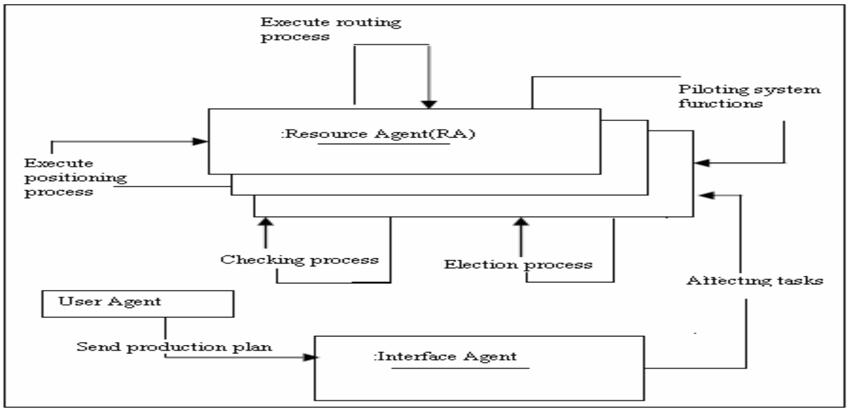

Fig. 1. Collaboration diagram of the system functioning

Fault detection and diagnosis. The fault tolerance is an important aspect characterizing a PS functioning. It is the reason for which the proposed MAS must detect and treat failures that may occur, in order to avoid the operator intervention (apart from extreme cases). The failures can be classed into two categories:

1. *Soft failure* The breakdown can be either localised when: -Communication link between two agents is destroyed. -failure is within the agent itself.
2. *Hard failure* This type of failure can be considered when malfunction of a machine happens; we call this kind of dysfunction a resource failure. The

latter is detected and diagnosed by the agent responsible of the resource below.

Failures can be detected and generally recovered through the MAS self-organising protocol without any external intervention. Detection and diagnosis of the soft failures are provided through the checking process. The checking process is carried out by RA at the system starting up.

Failures treatment. Based on the self organizing protocol, failures are:

1. *Resource Failure.* The agent responsible for this resource launches the routing process to prevent its neighbours that the agent responsible for the resource down, can not perform all its assigned tasks. So the concerned agent has to send its own address rather than sending its neighbours address. After the execution process, two cases can be found:
   -At least one RA can execute the tasks of the agent responsible for the breakdown resource (tasks are carried by the neighbourhood)
   -No agent can perform the tasks of the breakdown agent and in this case the operator has to intervene.
   If the tasks of the agent responsible for the resource down are all taken by the neighbourhood the problem is solved, otherwise, the system continues to operate without the breakdown machine pending the operator's intervention.

2. *Agent resource failure.* Two situations are quoted briefly:
   (i)RA(breakdown) is the alone agent of another RA.
   (ii)RA has several neighbours.
3. *Communication bond destroyed* When a communication link is lost, three situations can be considered:
   (i)RA isolated or the system is broken down into two subsystems.
   (ii)Isolated RA is a neighbour of an agent RA belonging to a sub system.
   (iii)The two RA belong to the same subsystem.

5 The Protocol

To make the paper self contained we present the most important processes characterizing the self organizing protocol [5] that are inspired from life within the bacteria colony. Particularly in this paper, processes are formalized in a formal manner. Each process uses a set of primitives or methods like: Leader(), groupe(), replace(), Ask(), Explore(), Life(), Rep(),.., for selecting a leader, grouping the MAS , asking for some information, exploring an agent, checking . The organization consists in N agents $A = a_1, a_2, .., a_n$, where each agent is considered as a unique node in a social network. The organization is modelled by an adjacent matrix E, where each element of E is like : $e_{ij} = 1$ if there is a communication bond between a_i and a_j , $e_{ij} = 0$ if not.

5.1 Positioning Process

Any agent in the MAS must position itself by carrying out the process of positioning. As soon as an agent integrates or leaves the group, the process of positioning is started. When an agent receives a position message: $position(a_i, role, pos_i)$, $i \in N, a_i \in A$, all agents positions and their identification are summarized in a table. $\forall j \in N, a_i \in A$, $e_{ij} = 1 \Rightarrow a_j$ is a direct neighbour of a_i and can receive a_i messages.
$\forall k \in IN, a_k \in A\ /e_{ik} = 1 \Rightarrow a_k$ receives position($a_i, role, pos_i$), If a_k receives position $(a_i, role, pos_i)$ for the first time then $Pos_{src} \leftarrow pos_i$;$pos_k \leftarrow pos_{src} + 1$;

5.2 Checking Process (Life Signal)

Local checking is an essential process. Each agent regularly diffuses a life signal to its neighbourhood. $\forall i \in N, a_i \in A, a_i$ sends life(a_i) to point out that it is Kept-alive to $a_j/e_{ij} = 1$;
If $\exists a_j$, $a_i \in$ A / $e_{ij} = 1$ and a_j did not receive $life(a_i)$ then a_j sends $Explore(a_i, a_j)$ to point to all the neighbourhood that it is seeking for $a_k/e_{ik} = 1$ and a_k already received $life(a_j)$. At a receipt of $Explore(a_i, a_j)$, if $a_k/e_{ik} = 1$ and a_k already received $life(a_j) \Rightarrow a_k sendsRep(response, pos_j)$ with $response = Yes/No$.
pos_j: position of the agent receiving *Explore()* message by a_j which is searched. If a_k / $e_{kj} = 1 \Rightarrow pos_j = 1$ else $pos_j > 1$. endif.

Three situations can appear:

1. If $\exists a_k$, $a_k \in$ A / $e_{ik} = 1$ and $response = Yes \Rightarrow$
   $state(a_j)$ not-in-failure
   $state(e_{ij})$ in-Failure
2. If $\exists a_k$, $a_k \in$ A / $e_{ik} = 1$ and $reponse = No \Rightarrow$
   a_k is / e_{jk} =1 and a_k did not receive a keep- alive signal from a_j;
   $state(a_k)$ in-failure;

3. if the received responses are:
   ($\forall k$,k$\neq$ i and k$\neq$ j , $pos_k - pos_i \neq 1$) $\Rightarrow a_j$ is isolated **or**
   a_j is / $\forall k \in N$, $j \neq k$ / $e_{ki} = 1$ (ie, $a_k \in$ agents group different from a_i one);
   $state(e_j)$= {in-failure or not in-failure}

Communication bond dysfunction. *(i)An isolated agent or system dissociated into two subsystems.* $State(e_{ij})$=in-failure; a_j changes its port address and sends $life(a_j)$; a_j answers by $life(a_j)$ that it agrees for the new address; in case of no possibility to change the port address, the dysfunction is certainly within the material and an alert is set off to change the link.
(ii)An isolated agent neighbour of an agent pertaining to a subsystem(Figure2).
Let set a_i the isolated agent; a_i changes its address port and sends life(a_i) message to a_j, a_j replies by life(a_j) to agree the new address. Only the isolated agent can change its address.

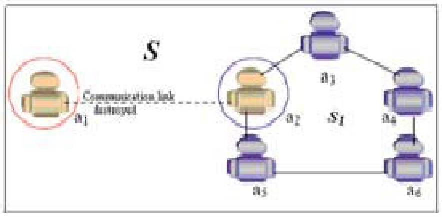

Fig. 2. An agent isolated by the destruction of its communication link

(iii)both agents pertain to the same system. If $\exists a_i$, a_j / a_i, $a_j \in$ A,(a_i and a_j pertain to the same system) $\Rightarrow \exists$ k $\in$ N / $a_k \in$ A, e_{ik} = 1 and e_{kj} = 1, which means that it is possible to find another way to assure the communication between a_i and a_j. The latter launches the *positionning* process and the *routing* process.

Agent failure. *(i)Failed agent is the unique neighbour of another agent.* a_i is the failed agent, a_k is the unique neighbour, a_s searched agent
(a)a_k is able to execute a_i tasks, a_k launches election process. The leader selected removes a_i . a_k launches *positioning* process.
(b) a_k is unable to execute a_i tasks set, the *positioning* process is launched.
If a_k finds a_s
a_s launches *election* process
a_s launches *positioning* process
else
a_k launches *election* process. The leader removes a_i, creates a new agent, affecting it a_i position and tasks.
endif
(ii)Failed agent is a neighbour of several agents
(a) Each agent can execute a subset of tasks
$\forall$ i, k $\in$ N, a_i , $a_k \in$ A, e_{ik}=1, state(e_{ik})=in-failure, $\exists$j $\in$ N / Tasks[a_j]= Tasks[a_j]+ Tasks[a_i](Each tasks subset is affected to a_k)
(b)No agent can execute a subset of tasks
$\forall$ i, k $\in$ N / a_i , $a_k \in$ A, e_{ik}=1, state(e_{ik})=in-failure
$\forall$ j $\in$ N / e_{kj} =1, then a_j launches *routing process.*

5.3 Election Process

Election process is charged to select a leader agent for any decision as adding or removing an agent. It can be launched by more than one agent. As a result an agent leader is selected based on its fitness value. Once the dysfunction is located, a_i sends Leader(chefglo, fitness) to a_j, where chefglo is agent identity and fitness is the fitness value initialized to zero and incremented by 1 after a task accomplishment by an agent.

$\forall$ i $\in$ N, $a_i \in A$, Leader(chefglo, fitness) message is received and propagated in the neighbourhood until getting the highest fitness value; the leader is the agent corresponding to this value.
If $\exists i, k \in IN/\ a_i,\ a_k \in A$ and fitness(a_i)= fitness(a_k), the leader is the one having the long identifier. In this case *Election process* is launched for removing or adding agents in the group and for any decision to take when necessary.

5.4 Grouping Process

In order to avoid that the election process spends too much time selecting the leader, especially when the agent number is high, the Grouping process groups agents by role, without inhibiting all the group; are inhibited only those that have the same role.
$\forall$ i $\in$ N, $a_i \in$ A, a_i sends Groupe(chefglo, fitness) message to a_i / Role(a_i)= Role(a_i).

5.5 Routing Process

The routing process is useful in two cases:
(i) When an agent (not in failure) attempts to replace the agent in failure by other/others in the neighbourhood.
The routing process allows to an agent a_i to select a_j in A where $j = 1, n \wedge j \neq i \wedge$ a_j replaces a_i;
Replace ($a_{failure}$) will be sent to all a_i / i $\in$ N, $a_i \in A$ $/i = 1, n$ and i $\neq$ failure
Task [$a_{failure}$]: tasks table of the failed agent
A variable is used to indicate if task[$a_{failure}$] is empty or not. It is initialised to false when an agent detectes its neighbour in failure. While receiving Replace($a_{failure}$) with task[$a_{failure}$] updated, each neighbour verifies the content of task[$a_{failure}$]. If tasks are all under its capacities, it adds them to its tasks table, otherwise it takes those that are under its capacities and sends replace ($a_{failure}$) to the neighbours.
(ii) When an agent is lost by a destroyed communication bond, its neighbour tries to find another way to reach it, and by searching in the neighbourhood those that can serve as intermediary. *ask(a_{src}, a_{emet}, vois-info)* message is sent to its direct neighbours ie), *ask(a_{src}, a_{emet}, vois-info)* is sent to a_k / k=1, n and k $\neq$ src and k $\neq$ emet. While receiving the message for the second time by a_k, which can't be the intermediary between the sender and the source agent then a_k agent acquits the sender agent by sending acquit(nb, pos_{src}). While receiving the message for the first time and it is capable to be the intermediary, between the sender agent and the source agent, pos takes the value of its position relatively to the source that we analyze as follow:
-$pos_{src} > 1$ means that the receiver of the message is not a direct neighbour of source agent. The receiver of the message computes the set of agent that have not received the message Ask(), the non informed agent set will receive the message ask(a_{src}, a_{emet} ,vois-info).

6 Implementation Aspects

We have modelled via AUML (Agent Unified Modelling Language) the whole production system processes. The latter was tested using the NetLogo simulation tool [11]. It is a modelling programmable environment to simulate natural and social phenomena. In this paper we can not represent all the cases. Is represented below the case where an agent is in dysfunction (figure3). In the first case, by affecting the breakdown agent tasks in the neighbourhood, and without creating another one, in its localization, the production system is still working correctly until the product is achieved.

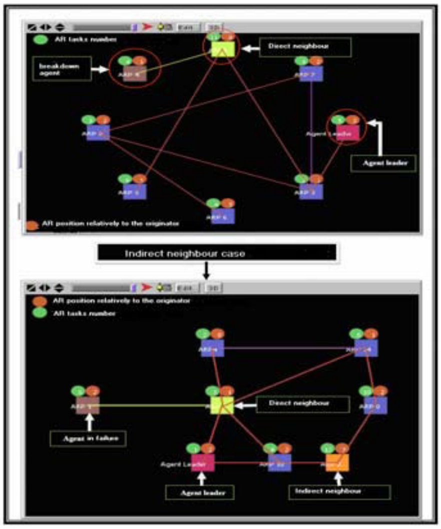

Fig. 3. An agent failure simulation

7 Conclusion

In this work a multi-agents system has been proposed for monitoring the functioning of a production system. The proposed MAS is based on a self organizing protocol[5] we have formalized using mathematical symbols. The protocol has the feature of assuring control decentralization so that each agent can take decisions to interact with its neighbours when necessary, and adapt when environment presents dysfunction. The MAS maintains its connectivity without any external intervention. Agents can perform several functions during their life as being leaders, which is not specific to a single agent. By the mean of their checking and decision they allow the emergence of new organizational structures, through agents' interactions, to cope with not desirable changes, and this is what increases the system fault tolerance and robustness.

References

1. GIARD V. Gestion de production. 2nd edn. Paris: Ed Economica (1988)
2. Draghici, G., Brinzei, N., Filpas, I.: La modélisation et la simulation en vue de la conduite des systèmes de production. Les cahiers des enseignements francophones en Roumanie (1998)
3. Habchi, G., Huget, M., Pralus, M.: D'une approche composant vers une approche agent pour un pilotage optimisé des systèmes de production. 6eConférence Francophone de MOdélisation et SIMulation, MOSIM 2006 Rabat, Maroc (2006)
4. Suarez, O.A., Foronda, J.L.A., bren, M.: Standard based framework for development of manufacturing control system. International Journal of Computer Integrated Manufacturing 11(5) (1998)
5. Mellah, H., et al.: Massop: A self organizing protocol for Multi agent systems inspired by bacteria colonies. In: CODS 2007, China, sisn.2007.07.090, July 2007, vol. 1(3), pp. 310–314 (2007)
6. Bussmann, S., Jennings, N.R., Wooldridge, M.: Multiagent Systems for Manufacturing Control: A Design Methodology. Springer Series on Agent Technology. Springer, Heidelberg (2004)
7. Roy, D.: Une architecture hiérarchisée multi-agents pour le pilotage réactif d'ateliers de production. Thèse de Doctorat: Université de Metz (1998)
8. Kouiss, K., Pierreval, H., Mebarki, N.: Using muliagent architecture in FMS for dynamic scheduling. Journal of Intelligent Manufacturing 8, 41–47 (1997)
9. Bussman, S., Schild, K., et al.: Sel-organizing Manufacturing Control: AN industrial Application of Agents technology. In: 4th Int. Conf. on MAS (2000)
10. Davidsson, P., et al.: A MAS architecture for coordination of just-in-time production and distribution. In: SAC 2002, Spain (2002)
11. Wilensky, U.: NetLogo User Manual version 3.1.1 (2006)
12. Radhika, N.: Programmable Self Assembly: constructing Global Shape using Biologically inspired local intercations and Origami mathematics. PhD Thesis, MIT (2001)
13. Mellah, H., et al.: A communication model of distributed information sources bacteria colonies inspired. In: 10th IEEE International Conference on Intelligent Engineering Systems, INES 2006, London,UK (2006)
14. Jacob, E.B., et al.: Modelling branching and chiral colonial Patterning of Lubricating Bacteria. In: Proceedings of IMA workshop: Pattern formation and Morphogenesis (1998)