E. Tranvouez, A. Ferrarini, B. Espinasse (2006), « Cooperative Disruption Management in Industrial Systems: a Multiagent Approach », INCOM 2006, Special Track on Holonic and Multi-agent Technologies for Industrial Systems, Saint Etienne, 17-19 May 2006.

COOPERATIVE DISRUPTION MANAGEMENT IN INDUSTRIAL SYSTEMS: A MULTIAGENT APPROACH

Erwan TRANVOUEZ, Alain FERRARINI and Bernard ESPINASSE

LSIS UMR CNRS 6168, Université Paul Cézanne, Domaine Scientifique de Saint Jérôme, Marseille, 13397 Cedex 20, France. {forename.name@lsis.org}

Abstract: This paper presents a cooperative repair method for disruption management in industrial systems. Our method aims to enable industrial systems, and in particular workshops, to minimize the consequences of disruptions in their forecasted schedule. In order to define rescheduling measures proportional to a disruption importance, a set of distributed repair strategies, ranging from isolated to cooperative solving process, are defined. Agent based modelling is used to specify and operate these cooperative processes. Agent-based simulation illustrates the application of this method on a workshop where autonomous machines cooperate to manage disruptions. *Copyright © 2006 IFAC*

Keywords: Disruption Management; Repair Scheduling; Distributed Artificial Intelligence; Multi-agent Systems, Cooperative Problem Solving.

1. INTRODUCTION

Distributed Artificial Intelligence (DAI), and more particularly MultiAgent Systems (MAS), proposes innovative solutions to many industrial problems (Müller and Parunak, 1998). A distributed approach can improve the modelling and performances of complex systems such as managing changes in a schedule in a disrupted environment (Pendharkar, 1999). This paper presents a cooperative repair method for industrial plans with a Cooperative Distributed Problem Solving (CDPS) approach supported by a MAS framework. This method proposes to minimize the consequences of disrupting events on their production scheduling. We aim at testing and validating the systems agility through agent-based simulation.

The second section of the paper introduces the disruption management problematic. The third section presents a cooperative repair method for disruption management based on distributed repair operations on schedule organized in strategies. In section four, a multi-agent modelling approach enabling the cooperative repair method is introduced. The section five develops the agent-based simulation of these cooperative methods with some details on the implementation. Simulation results are also presented on an illustration case to validate the first steps of the method. Finally, we conclude on the interests of these methods and its perspective of application to the supply chains management.

2. DISRUPTION MANAGEMENT PROBLEMATIC

In nowadays industrial context the intrinsic uncertainty of the environment generates events that disrupt pre-established plan controlling processes. The problem of disruption management is present in most of industrial areas and becomes a high interest topic (Gu and Qi, 2004). The main problem consists in dealing with these unanticipated events that may disrupt the system and get the plan deviate from its intended course and even get it unfeasible. The problem becomes “how to reach the plan goals while minimizing the effects "(negative impact") of the disruption(s)”. A disruption is defined as a situation during the operation's execution in which the deviation from plan is sufficiently large that the plan has to be changed substantially.

An usual solution, in industry, consists in pre-producing a recovery plan that can be applied on the day of the disruption and that integrates recovery solution(s). But this approach often lacks reactivity and decision-makers often stop after having generated a single feasible option for recovery; (computation time simply does not allow for several structurally different alternatives generation) (Clausen et al., 2001).

Various works has relied upon distributed scheduling to generate more efficiently these plans. We distinguish four main classes of approaches among these works (see (Tranvouez, 2001) for a detailed discussion and more complete review): Resource allocation, Decision support, Coordination and Cooperation. We have considered a strict definition of cooperation that excludes interaction between solving entities (or agents) belonging to two different hierarchy levels. These classes are described below.

Decision Support: the solving process focuses on helping a human decision-maker as in (Grabot *et al.* 1999). Such works intend to integrate one or several users in the solving process, taking into account their individual objectives or preferences. Computation is distributed between a scheduling software (distributed or not) and one or more human decision makers.

Resource Allocation: the problem is to find the most efficient resource (see (Tharumarajah 2001) for a detailed review). This process is repeated for all the tasks to be scheduled and is usually based on a call for proposal mechanism. The allocation can be controlled by a manager agent (named once for all or regularly changed) as in (Rabelo and Camarhina-Matos, 1998), decentralised in (Ferrarini and Bertrand, 1997) or mixed.

Coordination (centralised or decentralised), the scheduling problem is decomposed in several sub-problems: how to maintain the coherence of the local solutions of each sub-problem while dealing with their dependencies. Then how to ensure an efficient global solution is reached by interacting agents. Strong hierarchical societies usually rely upon coordinators to control and enforce behaviour from subordinate agents. Other works propose decentralised societies with mutual adjustments mechanisms as in (Liu and Sycara, 1998).

Cooperation, when autonomous problem-solving entities jointly participate to the solving process as in (Miyashita, 1998; Espinasse and Tranvouez, 1998). Such works consider that pure hierarchical organisation of production systems act as a brake on their reactivity. On the opposite, they propose a decentralised approach where autonomous decision centre cooperate in order to solve resource conflicts.

This literature review reveals that few research works use full cooperative interactions as a solving process. Moreover, online generation of plans for dealing with environment uncertainty does not usually take into account the degree of impact of the new plans on the organisation of the industrial system. Our aim is to promote cooperation down to the lowest level of the industrial system hierarchy, in order to tackle the disruption the soonest possible as well as providing means to react adequately to a disruption importance. In other words, we propose a distributed problem solving method allowing the lower levels' decisional centers to autonomously seek to minimize the changes in the pre-disruption plans (and so avoid frequent complete computation and organisation modifications when subsequent adjustment could be made). Thus earlier time expensive computations are capitalized and the solving process includes the human factor as all these changes have strong consequences on how human operators plan and work.

3. A COOPERATIVE REPAIR METHOD FOR RE-SCHEDULING

This method is based on an organisational modelling of an industrial system granting sufficient decisional autonomy to the considered resources. As such, application domains can range from production workshops to supply chains (Dalmau, 2004). In this paper, the method is tailored to a production workshop; where the considered autonomous resources are the production machines. This autonomy enables the machines to cooperate in order to manage the effects of a disruption (such as a machine or tool breakdown, delays on production …) the soonest possible (Ounar *et al.*, 2002). To reach this objective, we have defined a set of rescheduling strategies consisting in sequences of repair scheduling operations which are detailed below.

3.1 Distributed Repair Operations on schedule

A repair scheduling operation can be defined as the limited and local modification of a previously computed scheduling (Zweben *et al.,* 1993). These modifications consist in task shift or tasks permutation in one or several production plans. Repair scheduling is particularly adapted to work with incomplete scheduling (some tasks are not endowed) or partially deficient (time constraints violation) resulting for example from workshop disruptions.

Moreover, this approach has already been successfully applied to distributed scheduling (Rabelo and Camarinha-Matos, 1998). Repair solutions allow confining rescheduling to a single production machine while promoting negotiation or cooperation between machines. As an example, Miyashita's CAMPS distributed scheduling system (Miyashita, 1998) is based on a multiagent architecture composed of three agent types: (i) a *manager agent* representing costumers ; (ii) *planner agent* associated to a product and (iii) *scheduler agents* associated to a resource. Conflicts in the local scheduling computed by the agents are sorted out through negotiations.

Such operations, when performed on a particular machine scheduling, must comply with other machines production plans. This constraint derives from the distributed nature of the solving approach and thus requires machines to know part of the other machines plan. The temporal margin d_min and d_max are defined in that purpose. d_min (resp d_max) constitutes the earliest (resp. latest) beginning time of a task execution. Thus all the constraints of a task are summarized in these two variables enabling a machine to know to what extend it can modify its plan without jeopardizing other machines plans. Cooperation and negotiation take place when such margins appear to be insufficient to manage a disruption effect.

3.2 Distributed Repair Strategies

The disruption management process is organised in 6 steps, each step implementing a particular repair strategy. When it cannot be handled at the step n, the disruption is passed on the step n+1 denoting its increasing threat (i.e. whether it involves one, two or several machines,) to the scheduling stability. Below a typical sequence of steps is illustrated in figure 1:

This sequence begins with a disrupted machine, (i.e. the machine targeted by the disruption), trying to handle alone the disruption by modifying locally its scheduling (**Step 1 -** arrow 1 on figure 1). This is done with respect to other machine constraints. In case of failure **Step 2** requires the disrupted machine to engage unilateral cooperation (e.g. task transfer from one machine workload to another) (arrow 2a) or bilateral cooperation (tasks exchanges between two machines - arrow 2b). Confronted to a strong disruption, **Step 3** consists in extending the cooperation process to all machines of the workshop (arrow 3). If nothing can be done at this point, the disruption is said to pass from level 3 (the machine level) to level 2 (the workshop level) as indicated by arrow 4a. This emphasizes that the disruption impact now concerns the whole workshop.

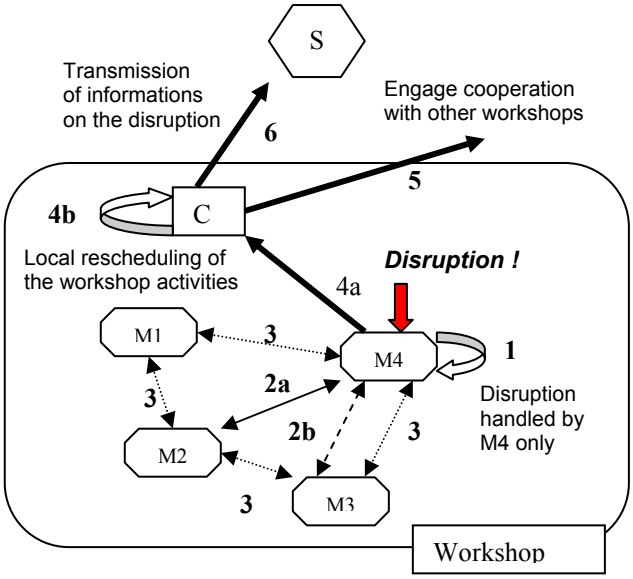

Figure 1. Rescheduling Strategies

Step **4** to **6** are used when the disruption is too important and requires global rescheduling, first tried by the workshop coordinator on the machines under his responsibility (arrow 4b). If this fails, cooperation with other workshops is engaged with mechanisms similar to those of step 2 and 3 (**step 5**). Finally, the workshop coordinator transmits information on the disruption, to the production manager (level 3) in order to relax some of the constraints (**step 6**).

These 6 steps intend to tackle the disruption effects with actions gradually increasing in complexity. First one machine then two are concerned to finally extend the disruption impact to the whole workshop or production facility.

3.3 Solution Evaluation

The evaluation of solution ***s*** is based on a multicriteria function measuring its ***efficiency*** (how successfully the disruption is absorbed), its ***complexity*** (the range of scheduling modifications the disruption caused) and its ***manoeuvrability*** (how the solution affects the scheduling liberty margin). These three pragmatic criteria enable considering solutions that solve efficiently a disruption with limited plan modifications as well as preserving the plan potential reactivity to future disruptions. Some examples on how such criteria can be measured are proposed.

Efficiency, noted ***E(s)***, is measured by the *mean delay* of the production tasks computed as described below. It can either be optimistic (delay and advance are compensable) or pessimistic (only delays are considered) and as well be computed locally for a machine or globally. Of course, other criteria can be included through a mulcriteria function.

Complexity, noted ***C(s),*** is defined as a weighted sum of the type and number of repair operations used (how many shift, permutation …) as well as the strategy used (internal solving, cooperative rescheduling …). For example shift operations can be seen as more easily to put in place (in the operator point of view) instead of permutations. The same logic is applied to weight repair strategies: task transfer for example involves only one machine whereas tasks exchange between two machine results in two workload modifications. Note that this complexity definition is also a measure of the disruption magnitude.

$$C(s) = S_s \times \sum_{i=1}^{op} n_i \times op_i \qquad (1)$$

with
- s : a repair solution
- S_s : weight of the strategy used to obtain s.
- op : number of repair operations used to obtain s.
- op_i : weight of the repair operation type i (e.g. shift).
- n_i : number of repair operation type i used to obtain s

Manoeuvrability, noted ***M(s),*** is computed as an average of the tasks margins. This avoids selecting solution resulting in over constrained plans which may jeopardize the plan stability if other disruptions occur. Individual manoeuvrability (M_m) is computed locally by each machine.

The solving process follows a contract net protocol (Smith and Davis (1998) where the disrupted machine plays the coordinator role. This role is attributed according to the disruption context and is not previously decided once for all. For example, in the task transfer strategy (unilateral cooperation of step 2), the disrupted machine sends a call for proposal to the machines able to carry out one of its tasks. If a machine finds a feasible solution, it is proposed to the disrupted machine. The latter can evaluate order and select the best solution.

4. MULTIAGENT MODELLING

Multiagent modelling enables defining the organisation of the agent society as well as the agent internal structure. Because of space limits, it is briefly exposed here (interactions will not be studied). However, it is interesting to note that multi-agent modelling conveyed the concepts needed to describe more precisely how the method could be undertaken in a workshop. Therefore, multi-agent modelling participates to the definition of the cooperative disruption management method.

4.1 Multi-agent Organisation

The multiagent organisation (figure 2) is composed of two main agent types: *service agents* and *scheduling agents*.

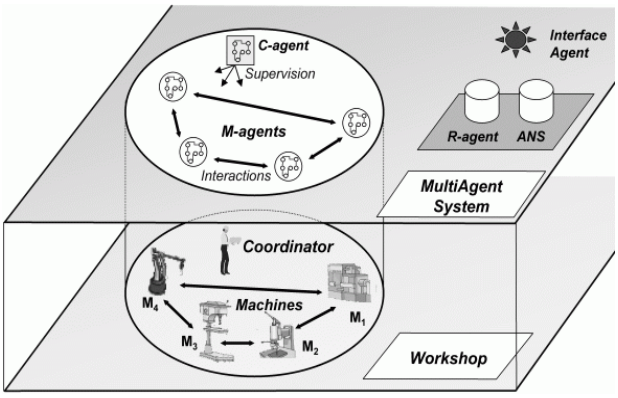

Figure 2. Multiagent Organisation

The *service agent* manages the MAS operations independently of the MAS application domain. For example, the *ANS* and *R-agent* agents can be defined as white and yellow directories maintaining information on names, physical address and competencies of all the agents. The *interface* agent allows interacting directly with the MAS either to feed the MAS with data or to probe agents' knowledge or running state.

The *scheduling agents* mirror the actual workshop organisation the MAS represent. The C-agent (representing the workshop coordinator) can access all the information from the M-agents it supervises. This role is minor until step 4; and thus we have stressed the modelling effort on the M-agent. However, it may intervene to translate external disruption into one or several internal disruptions. The solving process is presently mainly carried on by the M-agent (step 1 to 3). This agent can compute local rescheduling as well as cooperate with other M-agents to solve the disruption.

4.2 Agent Model

The proposed agent model defines the structure of the agent help specifying its individual and social behaviour. Each agent is built upon four modules as illustrated in figure 3.

The *Communication Module* manages message sending and receiving to and from other agent(s). It also insures messages are correctly expressed (ie follows ACL syntax), and understandable (i.e. referring to known cooperation abilities) as well as pertinent (i.e. not referring to old conversations).

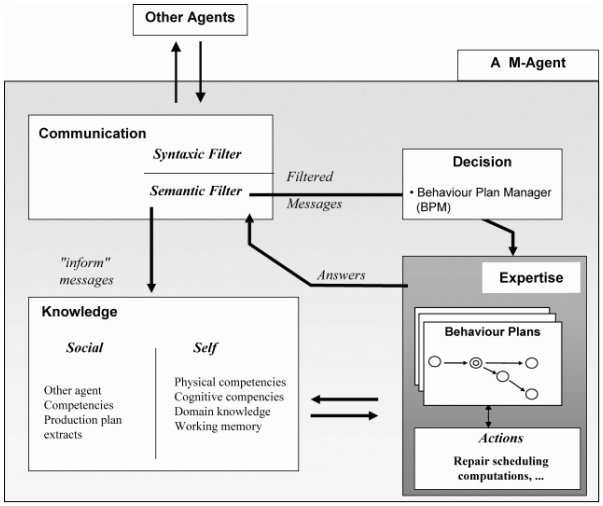

Figure 3. M-agent Model

The *Knowledge Module* contains the agent self and social knowledge. Self-knowledge defines what the agent knows about itself and its associated production machine. This includes its physical competencies (e.g. machine production technical properties), cognitive competencies (possible cooperation, expertise…), domain knowledge (machine state or current activities) and work memory (current working plans). Social knowledge defines what the agent knows about other agents i.e. their names and parts of their self-knowledge.

The *Decision Module* controls the actions undertaken by an agent. This implies the management of the Behaviour Plans execution as well as decision making support such as solution evaluation function. The agent model proposed here can thus be related to a Belief Desire Intention (BDI) approach (Fisher *et al.,* 1996), as agents behaviour are explained and designed by plans denoted Behaviour Plans (BP). Whereas the expertise in the *Knowledge Module* describes what the agents can do, the *Expertise Module* states how to carry out these actions. Thus, this module contains the BP specifying the agent behaviour schemes it can follow during its execution (disruption management, answers to queries...). The distributed algorithm, globally described in section 3.2, benefits from agent concepts (autonomy, communication protocol, behaviour definition through plans…). Therefore, multiagent modelling is a part of the cooperative repair method definition.

5. MULTIAGENT SIMULATION

A software prototype called MACSS has been developed to test and validate rescheduling strategies by simulation. Multiagent simulation with MACSS consists in providing each agent with the production plan of its associated machine. The platform is first introduced, before presenting and analyzing a simulation case.

5.1 An Agent based simulation platform

MACSS implementation is based on Java and Jess (see http://herzberg.ca.sandia.gov/jess/docs/index.html).
Java is widely used in agent technology because of its APIs and intrinsic properties. Jess is a free CLIPS Expert System building shell developed in Java [21] providing rule-based, object-oriented, and procedural programming. MACSS actually consists of several Jess Console instances connected through a TCP/IP network. Agents' communication follows the Agent Communication Language standard proposed by the Foundation for Intelligent and Physical Agent (http://www.fipa.org).

Concerning scheduler agents, only the M-agent has been developed as only strategies from step 1 to 3 are currently studied. Its architecture follows the agent model defined previously in Figure 4 and reflects the Communication, Decision, Expertise and Knowledge modules. Behaviour Plan (BP) representations in Jess associate each state and transition to a rule. State activation is then simulated by asserting and retracting facts which are in turn used by the Jess inference engine to activate transition rules. Java is preferred for procedural programming and when fast computation is required. Thus, message transport, agent network connectivity and scheduling repair computations are implemented in Java.

The agent architecture isolates domain dependent information or computations in the Knowledge and Expertise module. Therefore, to reuse this agent for another application, the MAS programmer only has to specify the agents knowledge through facts, write the adequate BP and if necessary the corresponding Java code. Thus, MACSS has already been used to design and implement a multiagent ecosystem management support system (Franchesquin *et al.,* 2003) and a supply chain management system (Ferrarini *et al.,* 2001).

5.2 Simulation Case and results

To illustrate how the system works, this section presents a simulation case where a workshop is composed of three machines. Each machine has a specific tool gear with different physical constraints thus restricting the set of possible cooperations. Each M-agent is named by its associated machine identification (i.e. Mx).

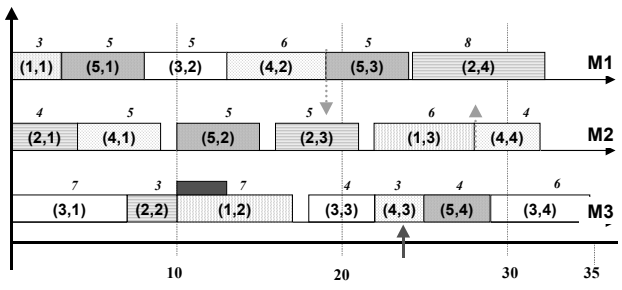

Figure 4. Disrupted Workshop Schedule

The simulation case states that agent M3 is informed of a disruption delaying task (2,2) end by 3 seconds (doubling its duration). Figure 4 shows the workshop initial state when the disruption occurs. The numbers above tasks represent their execution duration.

Because of lacks of margin, agent M3 tries a transfer cooperation consisting in requesting other machines to process task (5,4) in order to free sufficient space to absorb the disruption. In this case, only M2 owns the tool required. After verifying it can physically process the operation, M2 engages repair scheduling computations to see if it can respect task (5,4) constraints. Dotted arrows on figures 4 delimit the time window when the task must start in order to respect constraints from job 4.

M2 then computes that left shifting tasks (5,2) and (2,3) enable to add task (4,3) to its workload. After confirmation from M2, M3 in turn right shift tasks (1,2) and (3,3). Finally, both agents update their tasks margins value (as some beginning dates have been modified) and inform the other agents of these modifications. As shown in figure 5, each solution stores the criteria value, the sequence of repair operations used and the resulting plan.

```
Solution 1
 >> C = 8.0, R = 0.5, M = -0.5 => E = 8.0
Repair Scheduling Operations :
Left_Shift of Task ( 3 )=( 5,2 ) .
Left_Shift of Task ( 4 )=( 2,3 ) .
Insert        Task ( 5 )=( 4,3 ) .
Resulting Plan from : <INSERTION>
[...]
   Task ( 2 )=( 4,1 )   Start 4 for 5sec.
   Task ( 3 )=( 5,2 )   Start 9 for 5sec.
   Task ( 4 )=( 2,3 )   Start 14 for 5sec.
   Task ( 5 )=( 4,3 )   Start 19 for 3sec.
[...]
```

Figure 5. Solution representation in MACSS

5.3 Discussion on results

Experimentations (limited at the moment to two strategies) confirm the feasibility of a cooperative repair scheduling approach. Even if these results only partially validate this method, the cooperation processes did manage the disruption. The solving process can appear strongly dependant of sufficient margins (i.e. gap between d_min and d_max) in the workload. This is particularly true for the task transfer cooperation. However, different cooperation types such as task exchange cooperation (i.e. reciprocal transfer cooperation) or parallel modification of two agents plan this can lessen this margin dependence. This cooperation depends less on shifting repair operations but rather on permutation operations (within a machine workload or between two machines).

The simulation also shows MACSS ability to simulate complex interactions between cooperative agents, and so allows to validate this multi-agent approach for being the core of a future Decision Support System such a platform could either be a simulation based decision aid system to evaluate the impact of this disruption management method on industrial systems, or be a frame for a distributed control system.

6. CONCLUSION

This paper has presented a cooperative repair method aiming at minimizing the disruption effects on a workshop activity. This method is based on repair operations and strategies extending progressively the impact zone of the rescheduling process. Production machines are granted autonomy and local decision abilities in order to enable those machines absorbing disruptions through different cooperation modes. Applying this autonomy, each machine may use particular criteria and/or formula in order to adapt to the workshop heterogeneity. The disruption consequences are thus limited with the least modifications possible of the forecasted schedule.

If autonomy and cooperation are considered as unavoidable for agile manufacturing, industrial reality tempers this objective. Therefore distributed cooperative disruption management may not easily be undertaken in industrial workshop. Interestingly, as much as autonomy is needed at a micro level, it is compulsory at a macro level. Supply chain management for example is based on the assumptions that companies keep there autonomy, do not necessarily share all their information, are distributed (in all meanings) but try nevertheless to coordinate their activities to improve global behaviour. In such condition, our cooperative method does fit the objectives and constraints of supply chain management. Small adjustment are needed (machines are companies, tasks are products or lots) and the cooperative behaviours remain relevant. Current work (Dalmau, 2004) explores these opportunities.

REFERENCES

Clausen J., Hansen J., Larsen J. and Larsen A. (2001). Disruption Management. *OR/MS Today*.

Dalmau M. (2004). Résolution Coopérative et Distribuée de Problème : application à la gestion des perturbations dans les chaînes logistiques. *Master of Research MCAO*, University of Aix-Marseilles.

Denkena B., H. K. Tönshoff, M. Zwick, P.-O. Woelk (2002). Process Planning and Scheduling with Multiagent Systems. *Proceedings of BASYS 2002*. Kluwer Academic Pbs, 339-348.

Espinasse B. and E. Tranvouez (1998). Ordonnancement d'atelier coopératif et réactif : une approche multi-agent. *Journal of Decision Systems*, Vol 7, Special Issue, 215-237.

Ferrarini A., and J.C. Bertrand (1997). Distributed control of a flexible manufacturing line: the way of leading product. *Proc. IFAC-CIS'97*, Conference on Control of Industrial Systems, 366-371.

Ferrarini A., O. Labarthe and B. Espinasse (2001). Modelling and simulation of supply chains with a multiagent system. *Proceedings of ESS'01*, Workshop on Multiagent Modelling and Simulation, 893-897.

Fisher K., J.P. Müller, M. Pischel (1996). Cooperative transportation scheduling: an application domain for DAI. *Applied Artificial Intelligence*. Vol. 10, 1-33.

Franchesquin N., B. Espinasse, J. Serment (2003). Coordination for contract realisation in the hydraulic management of the Camargue. *Proceedings of ABS4*, Agent Based Simulations, Montpellier, France, April 21-23.

Grabot B., C. Berard and C. Nguyen (1999). An implementation of man-software cooperative scheduling: the IO Software. *Production Planning and Control*, Vol. 10 N°3, 238-250.

Liu J-S and K. Sycara (1998). Multiagent Coordination in Tightly coupled Task Scheduling, *Readings in agents*, M.N. Huhns and M.P. Singh Eds, Morgan Kaufmann Pbs., 164-171.

Miyashita K. (1998). CAMPS: a Constraint-Based Architecture for Multiagent Scheduling, *Journal of Intelligent Manufacturing*, Vol 9, 145-154.

Müller J-P. and H. V. D. Parunak (1998). Multi-Agent Systems and Manufacturing. *Proceedings of INCOM'98,* Nancy-Metz, France, June 24-26, 65-170.

Ounar F., E. Tranvouez, B. Espinasse and P. Ladet (2002). Pilotage par tentative d'ajustement du plan prévisionnel. In: *Méthodes du pilotage des systèmes de production*, Chap. 1, P. Pujo & J.-P. Kieffer Ed., Hermès-Lavoisier Pbs., pp. 27-60.

Pendharkar P.C. (1999). A computational study on design and performance issues of multi-agent intelligent systems for dynamic scheduling environments. *Expert Systems with Applications*, Vol. 16, 121-133.

Rabelo R.J. and L.M. Camarinha-Matos (1998). Generic Framework for conflict resolution in negociation-based agile scheduling systems. *Proceedings of 5th IFAC Workshop on Intelligent Sheduling/IMS'98*, 43-48.

Smith R.G. and R. Davis (1998). Frameworks for Cooperation in Distributed Problem Solving. *Readings in Distributed Artificial Intelligence*, Bond et Gasser (eds), Morgan Kaufmann, pp. 61-70.

Tharumarajah, A. (2001). Survey of resource allocation methods for distributed manufacturing systems. *Production, Planning & Control*, Vol. 12, N°1, 58-68.

Tranvouez E. (2001). IAD et Ordonnancement : une approche coopérative du réordonnancement d'atelier. *PhD Thesis*, University of Aix-Marseille.

Yu G., Qi X. (2004). *Disruption management : framework, models and applications*. Ed. World Scientific.

Zweben M., E. Davis, Daun B. and Deale M. (1993). Scheduling and Rescheduling with Iterative Repair. *IEEE Transactions on Man and Cybernetics*. Vol. 23, n°6, 1588-1596.