Proceedings of the 17th World Congress
The International Federation of Automatic Control
Seoul, Korea, July 6-11, 2008

A Petri net model of distributed control in a holonic Manufacturing Execution System

I. Demongodin, J-C. Hennet

LSIS, Faculté de Saint Jérôme, Avenue Escadrille Normandie Niémen,
13397 Marseille Cedex 20, France
(Tel: 33 4 91 05 60 00; e-mail: isabel.demongodin (jean-claude.hennet)@lsis.org).

Abstract: In a holonic manufacturing execution system, operations assignment and scheduling can be decided in real time by automatic negotiation between order holons and resource holons. Negotiation is used to conciliate the goals pursued by the software agents who constitute the control part of the holons. This paper proposes mathematical criteria to represent the agents' goals, a game theoretic approach to analyze the possible outcomes of the negotiation process, and a Petri net model to represent the interaction protocols between order and resource agents. This model is analyzed in view of verifying the required properties of the protocol. In particular, certain conditions are proposed under which the following requirements are verified: no-blocking, selection of exactly one resource for each order, protocol termination in a bounded time.

1. INTRODUCTION

For the last ten to twenty years, the agent paradigm has become essential to conceive, describe and implement distributed decision, execution and control processes in many application frameworks, particularly in manufacturing systems, supply chains and service networks. According to Fox and co-authors (2000), "an agent is an autonomous, goal-oriented software process that operates asynchronously, communicating and coordinating with other agents as needed". From this definition, it clearly appears that the actions performed by an agent tend to replace actions performed by human actors in more traditional organizations. These tasks may be rather complex, such as exchanging messages, taking decisions, and even developing strategies.

Multiple agent systems open new possibilities for distributed execution and control. In the manufacturing context, this trend to use intelligent agents has often been combined with the drive toward heterarchical and holonic organization. Autonomous cooperating agents associated with manufacturing objects are also called 'holons' (Van Brussel et al., 1998). In the holonic reference architecture for manufacturing systems, called PROSA, three basic types of intelligent agents were identified: order agents, product agents, and resource agents. In this architecture, staff agents were added to assist the basic agents with expert knowledge.

The first purpose of this paper is to identify the goals of the two types of agents involved in the execution stage: order agents and resource agents. The nature of these goals creates some competition: order agents compete for the use of resources, resource agents compete to execute the orders. In the considered distributed framework, the most appealing technique for solving conflicts is negotiation, or more precisely automated negotiation, since it only involves software agents. The second section of the paper describes the holonic model of the Enterprise network. Section 3 states that the outcome of an automated negotiation game could be theoretically predicted by game theory as a non-dominated solution. In practice, some key properties to be satisfied by a negotiation protocol are feasibility and termination in a finite time. The paper contribution in part 4 is to propose a negotiation protocol described as a time Petri Net and analyzed to show that these properties are verified.

2. MODEL OF AN ENTERPRISE NETWORK

2.1 Basic constructs

Several incentives have been initiated at the international level, to characterize the main Enterprise entities by basic constructs. In this matter, one of the leading incentives has been the UEML project (Vernadat, 2001) in which the proposed model involves Enterprise objects such as production units, machines, storage places, product structure, production orders, algorithms, and the enterprise agents who organize the objects in view of achieving particular goals.

2.2 Enterprise agents

A holonic system is a particular multi-agent system in which agents are associated with manufacturing processes on manufacturing objects. Holons are defined in the PROSA Reference Architecture as "autonomous co-operating agents" (Van Brussel et al., 1998). In the considered holonic execution system, the system operates by negotiation between order agents and resource agents. The considered structure is purely heterarchical, tasks being executed on resources according to local task schedulers run by resource agents. Agents of both types have to accomplish their tasks in an autonomous fashion, without any centralized control.

According to Zambonelli and co-authors (1994), there are two main types of multiple agent systems:

• Distributed problem solving systems, in which agents are permanent and explicitly designed to achieve a given goal.

• Open systems in which agents are designed to achieve an individual goal, can dynamically leave and enter the system.

Manufacturing systems include both types of agents. Resources are permanent objects in the manufacturing system. The operation of each resource is organized by its own resource agent. Resource agents and product agents are permanently assigned specific goals and specific roles in the system, that is, a well defined task or responsibility in the context of the overall system. On the contrary, orders are temporary objects dynamically created according to demands for end products and to material requirements for manufacturing intermediate and end products. Order agents are thus dynamic. Their interactions with the other agents are programmed in the database of the associated product agent, which contains the lists of necessary inputs and candidate resources with expected lead-times. To each input product correspond a new order agent and its associated data: product, quantity, due date. So, globally, the system is open since an order agent is created at each order arrival and destructed at order completion.

3. AGENTS GAMES

3.1 Concurrency and competition between order and resource agents

Agents are goal oriented and their goals may be antagonistic. Their behaviour and interactions can be analyzed with the help of Game Theory (see e.g. (Osborne and Rubinstein, 1994)): they constitute a finite set of players in a finite set of states, each player taking a decision on the basis of a preference relation represented by a utility function. Agents typically need to interact with each other in order to exchange knowledge, coordinate their activities and achieve their goal as well as possible. In terms of game theory, goal achievement corresponds to maximizing a utility function while taking into account that the other players (agents in our case) also maximize their own utility function.

For instance, order agents cannot accomplish their mission without using some resources. As a consequence of the interaction between each particular resource agent and several order agents, conflicts may be generated on the use of the resource and these conflicts have to be solved locally, by resource agent decisions, with possible prior negotiation between the order agents and the resource agent. In this respect, a clear limitation of software agents is the impossibility to create innovative solutions. They can only run optimization algorithms, heuristics or priority rules, or choose between several pre-programmed solutions. On the other hand, many advantages can also be identified: rapidity and memory size, rigor and objectivity. However, the agents' objectivity may not be sufficient to obtain globally optimal solutions, mainly because of the distributed decisional structure with limited information. Agents can be seen as players in a game. Being both reactive and proactive, they always produce the best local response (with respect to their own criterion) to the environment (created by the other players) that they are able to perceive.

i) The order agent goal

Basically, the goal of an order agent is to obtain the execution of its associated order, denoted o, by an appropriate resource, m, at the best possible date. The set of appropriate resources able to execute the order o is denoted $A(o)$. The information on the set $A(o)$ is provided to the order agent by its associated product agent. By construction of the data base of products, the constraint $A(o) \neq \varnothing$ is supposed always valid.

The order agent goal is then translated into a utility function to be maximized or, in the resource sharing context, into an economic criterion to be minimized, under the compatibility constraint:

$$m \in A(o) \tag{1}$$

Depending on the application case, this criterion, denoted C_o, may be defined in different ways. It provides the order agent goal with a quantitative index for comparing the proposals from resource agents, selecting the best resource and evaluating the level of achievement of the goal.

The desired time window of an order o is the interval starting with its release date, r_o, and ending with its due date, d_o. Let t_o^m be the starting time and p_o^m the processing time of order o on machine m.

The desired time window of order o is achievable on machine m if the processing interval is included in the time window:

$$t_o^m \geq r_o \tag{2}$$

$$t_o^m + p_o^m \leq d_o \tag{3}$$

a) In some cases, a minimum delay criterion is selected: $C_o = \max(t_o^m + p_o^m - d_o, 0)$ under constraints (1) and (2).

b) In other cases, the "makespan" criterion may be selected, when early intervals are preferred: $C_o = t_o^m + p_o^m$ under constraints (1) and (2).

c) Alternatively any deviation from the due date may be penalized with order dependent weights:

$$C_o = \beta_o \max(d_o - t_o^m - p_o^m, 0) + \gamma_o \max(t_o^m + p_o^m - d_o, 0)$$

under constraints (1) and (2).

ii) The resource agent goal

The main roles of a resource agent are to estimate the state of its resource, get information on the state of the resources connected to it, negotiate with the order agents for whom the resource is compatible, decide when the resource should be active or stopped, decide what tasks are to be processed, and monitor the resource execution processes. Generically, the

problem of deciding what orders should be processed and at what time, is referred as a scheduling problem. However, depending on the resource type, other Operations Research problems may better represent the addressed problem: bin packing problems for glass cutting machines (Blanc *et al*., 2006), knapsack problem for order selection under limited capacity, for instance.

The scheduling problem to be solved by each resource agent is generally a single machine problem. Nevertheless, it can often be globally efficient to include in the problem, in its criterion or in its constraints, the predicted effect of the solution on other resources, in particular on critical ones and on the ones frequently used just before or after.

Let O_m be a set of orders compatible with resource *m*. Sets O_m and $A(o)$ are dual in the representation of the compatibility relation:

$$o \in O_m \Leftrightarrow m \in A(o)\,. \qquad (4)$$

The optimization criterion that represents the goal of a resource agent is normally consistent with the order agents' criteria. For instance, if criterion b) is used by all the order agents, possible criteria that can be used by the resource agent are:

• the makespan criterion :

$C_m = \max_{o \in O_m} \left(t_o^m + p_o^m \right)$ under constraints (1) and (2).

• the sum of execution times :

$C_m = \sum_{o \in O_m} \alpha_o \left(t_o^m + p_o^m - r_o \right)$ under constraints (1) and (2).

3.2 The 1 order- q resources subgame

Game Theory relies on two main assumptions: players' rationality and integration of the information available on the other players' strategy. Accordingly, the decision of each player maximizes his utility function in the constrained set imposed by the other players. Several types of games have been defined, depending on the sequence of game stages. In a static (or strategic) game, there is only one stage and the players simultaneously select their strategy. In dynamic (or sequential games), players generally act and react iteratively in a predetermined order.

In the considered multi agent manufacturing execution system, the global assignment and scheduling game can be decomposed into subgames of the two following types:

• the *1 order - q resources* subgame: an order agent sends a request to *q* candidate resources, waits for the answers and selects the resource with the best offer for its execution,

• the *1 resource - n orders* subgame: a resource agent selects *s* among the *n* orders requests and proposes execution dates to the *s* order agents.

In order to avoid the possibility of failure in the negotiation protocol, the study is restricted to the case $s = n$ for the second type of sub game. This means that if a resource is technically eligible for executing an order, its associate resource agent must answer positively to the order agent, even if the due date cannot be satisfied. Note that this assumption is consistent with order agents criteria a), b), c) proposed in section 3.1.i), and for which late deliveries are allowed. Under this assumption, the negotiation game may proceed in no more than two stages:

• a Call For Proposals (CFP) is sent by an order agent to *q* candidate resources

• each of the *q* candidate resource agents sends a proposal and the best one is selected by the order agent.

In a general setting, combination of all the subgames of the two types (*1 order - q resources* and *1 resource - n orders*) leads to a complex sequential game for which the outcome is very difficult to predict and convergence is not guaranteed, at least in a pre-specified bounded time. On the contrary, the problem becomes much simpler if each resource agent applies an on line scheduling algorithm to position each order one after the other and waits for the result from the current order agent before processing the following order.

The multi-agent control system considered is an open system where orders keep arriving randomly and should be processed upon arrival. As stressed by several authors, such as Shen and co-authors (2006), real-time scheduling techniques are clearly the best ones for such systems. Many algorithms for single machine on line scheduling have been proposed in the literature, for instance the EQUI and LOGI heuristics in (Pesenti *et al.*, 2006). In a real-time scheduling algorithm, orders are treated one by one, sequentially. In the case of 2 orders arriving exactly at the same time, the tie can be broken arbitrarily. Treatment of an order results in provisionally assigning a time window for executing the order on the resource. This assignment defines the resource agent proposal to the order agent. Once the order agent has selected the best proposal, the execution time window is reserved on the chosen resource and liberated on the other resources. For data consistency, treatment of a new order should not begin before the decision on the preceding proposal has been taken and the planned execution schedule updated.

It can be noted that if orders are treated sequentially without any recursive change of previous decisions, then, for each resource $m \in A(o)$, the set O_m is simply the last arriving order: $O_m = \{o\}$ and the two current criteria C_m are equivalent and reduce to minimizing the delivery date: $t_o^m + p_o^m$, under constraints (1) and (2).

Under such an algorithm, each resource agent simply treats a *1 resource - 1 order* problem and such elementary problems are evaluation rather than optimization tasks. The negotiation game then reduces to a sequence of *1 order - q resources* subgames.

4 AGENTS' NEGOTIATIONS

Agents' negotiation can be defined as "the process by which a group of agents comes to mutually acceptable agreement on some matter" (Jennings *et al.*, 2001). It is thus considered as the most fundamental and powerful mechanism for managing inter-agent dependencies for decisions on task execution. In the *1 order - q resources* subgame, negotiation can be seen as the process through which one order agent maximizes its utility function by selecting the best candidate resource to process its order.

4.1 Negotiation protocols

The negotiation mechanism proposed in this study is a simplified version of the "contract net protocol" (CNP) (Smith, 1980). A semi-formal specification of the CNP protocol has been proposed in (Odell *et al.*, 2000), using Agent-UML. Some extensions of this protocol have also been proposed, to include in particular a multi-criteria evaluation of proposals (Ounnar and Pujo, 2005).

Here, the order agent acts as an initiator, in charge of getting its order processed in the most efficient manner. To this end, it sends a CFP (Call For Proposal) to all the resource agents eligible for processing the order (the participants in the CNP terminology). Each such resource agent acts as a potential bidder in submitting an execution proposal to the order agent. A proposal simply consists of a proposed time window for executing the order. For any order entering the system, the set of candidate resource agents is supposed never empty. Participants receiving a CFP cannot refuse it and should answer within a limited time. In any case, proposals are supposed honest and unbiased. The order agent evaluates the proposals and selects the current best one. As long as better offers arrive, the current best offer is updated. At each update of the best offer, the previous best offer is refused. Negotiation ends when all the proposals have been compared. Another difference with the CNP is that only the negotiation stage is represented. The purpose of these assumptions is to propose a negotiation protocol easily implemented in any ACL (Agent Communication Language) and to describe a framework in which a negotiation failure would never occur.

4.2 Order-resource negotiation models

Taking the negotiation requirements into account, a negotiation model based on the time Petri net formalism with coloured concept, is proposed. Coloured Petri Nets (CPN) (Jensen, 1992) are well-suited for systems that consist of a number of processes which communicate and synchronize. They can be used for design, specification, simulation and verification of systems. The marking of places, which describes the state of the system, corresponds to a set of coloured tokens. In ordinary Petri nets, tokens do not support information while in coloured nets, tokens are labelled by a type of data – possibility structured – called colour. For instance, the list of eligible resources for a given order can be associated with a data type in a place for this order model. Transitions of the Petri net model represent the change of states. In our model, each transition is associated with an action of an agent.

As previously described, time aspects are also very important in the approach. The assumption of representing program runs and evaluation tasks by time intervals seems reasonable and justifies the representation by Time Petri nets (Merlin, 1974). Time intervals are thus used to represent several types of actions: order scheduling by a resource agent, evaluation and comparison of proposals by the order agent. Consequently, time intervals have to be included in the negotiation model. Extended CPN with a time concept have been proposed to introduce a global clock and allow each token to carry a time stamp indicating when it is ready to be consumed by a transition. However, the time concept in the negotiation protocol should rather be associated with transitions, as they represent actions with bounded random duration. Moreover, when time is associated with transitions, logical conditions are tested first and time conditions are executed next. Using such formalism thus presents two interesting characteristics: on the one hand, it makes possible an explicit representation of parallelism, concurrency, synchronization and resource sharing. On the other hand, the time intervals attached to transitions introduce additional constraints on the protocol. In the considered Time Petri net, a temporal interval is associated with certain transitions. For the modelling purpose, the Time Petri net model used in this study has a strong firing rule and infinite-server semantics. In other words, an enabled transition must be fired at the static latest firing time and k firings of a transition can be performed simultaneously (or at the same time) if the transition is k-enabled.

The negotiation model for an order agent is proposed in Fig.1. Initialization is represented by transition t_0. The available order agent sends the CFP (place p_7) to each resource of list $A(o)$ and waits for a proposal (place p_2). When the first proposal arrives (place p_8) from a resource agent, the order agent initiates the best proposal (place p_6) and waits for other proposals (place p_3). When another proposal arrives from a resource agent, the order agent, which is in the state 'waiting for proposal' (place p_3), evaluates this proposal (place p_4) and compares it with the current best proposal, to select the best one. If the result of the comparison (place p_5) is positive, the evaluated proposal becomes the current best proposal (place p_6) and the last best proposal becomes a refused proposal (place p_{11}). Hence, place p_{11} informs the considered resource agent that its proposal has been refused. As long as better offers arrive, the current best offer is possibly updated. At each update of the best offer, the previous best offer is refused. When all resource agents proposals have been compared (place p_{13}), the negotiation is ended (place p_{14}) and the current best proposal is accepted (place p_{12}). In the coloured Petri net model, the exact number of tokens and their data values are determined by the arc expressions with respect to the firing rules of transitions.

The negotiation model for any resource agent is described on Fig.2. For each particular order agent, each resource agent acts as an evaluation module that produces an offer in a finite time. The q resource agent Petri nets of Fig.2 can be merged into the order agent Petri Net of Fig. 1 by merging the q+1 places labelled "CFP sent", the q+1 places labelled "Proposal

sent", the q+1 place "Proposal accepted" and the q+1 places "Proposal refused".

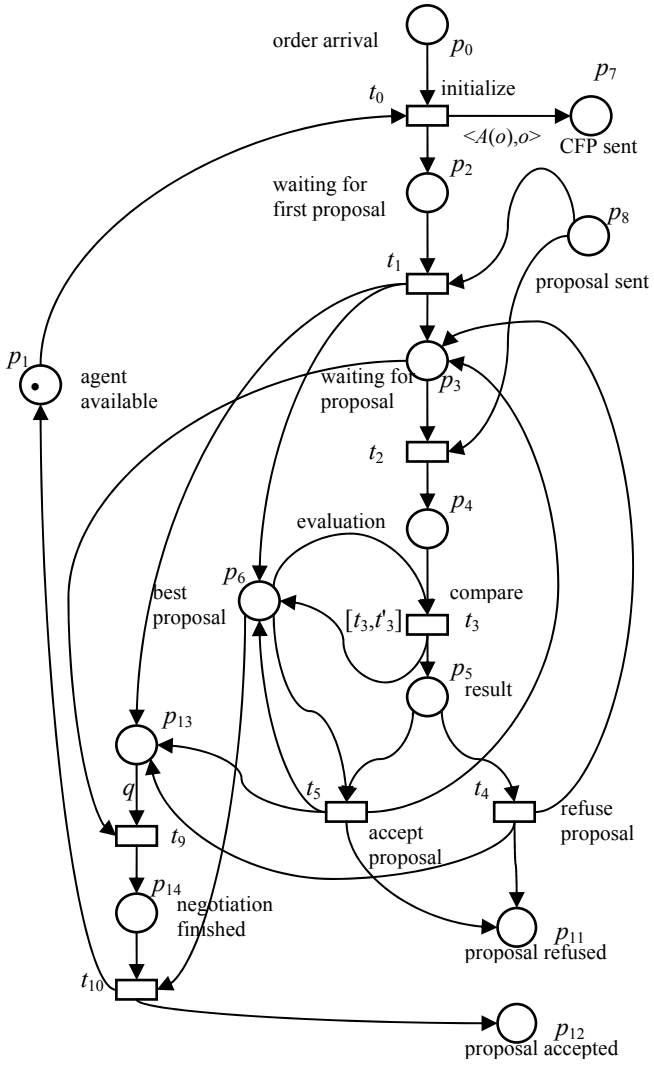

Fig.1 Negotiation model for order agents

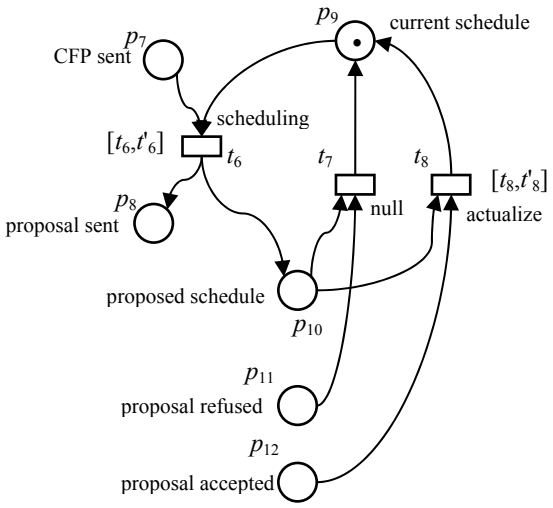

Fig.2 Negotiation model for a resource agent

4.3 Protocol requirements verification

In the analysis, randomness of processing times requires properties to be valid under any feasible firing trajectory. In the *1 order - q resources* interaction protocol, colours only represent the different candidate resources contained in $A(o)$. Thus the idea developed in this section is to aggregate colours representing resources. The aggregated negotiation model of the set of candidate resource agents has the same structure as in Fig. 2, an initial state with $q = Card(A(o))$ tokens in place p_9 and time intervals $[t_{6\min}, t'_{6\max}]$ and $[t_{8\min}, t'_{8\max}]$ respectively associated with transitions t_6 and t_8, with:

$$\begin{cases} t_{6\min} = \min\limits_{m\in A(o)} (t_6(m)), & t'_{6\max} = \max\limits_{m\in A(o)} (t'_6(m)) \\ t_{8\min} = \min\limits_{m\in A(o)} (t_8(m)), & t'_{8\max} = \max\limits_{m\in A(o)} (t'_8(m)) \end{cases}.$$

The order-resource negotiation protocol can then be verified and evaluated considering only the Time Petri net of Fig. 3, representing a *1 order - q resources* interaction protocol.

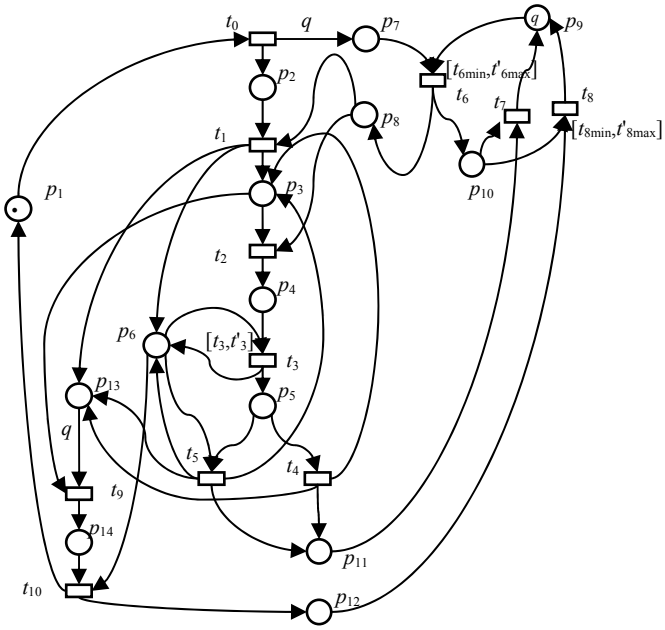

Fig.3 Global model with q resource agents

In a Time Petri net, the reachability issue leads to constructing an infinite number of states. To define a finite enumerative analysis method for characterizing the behaviour of Time Petri Nets, Berthomieu and Menasche, (1982) have proposed to introduce the notion of state class. Informally, a state class $C = (M;D)$ can be defined as the union of all firing time values which are possible from a given marking M. D represents the infinite number of firing times possible from M, defined by the set of all solutions of a system of inequalities. Using the enabling and firing rules, the state class graph can be constructed thanks to the software *Tina* (Berthomieu and Diaz, 1991).

In order to study structural and time properties of the proposed model in Fig. 3, place p_0 is not considered as it represents an arrival of orders (and a source place). For an initial marking equal to M_0 = [1 0 0 0 0 0 0 0 q 0 0 0 0 0], the

corresponding state class graph has a strongly connected structure. The following minimal invariants are computed.

Places invariants:
$X1$ = [1 1 1 1 1 0 0 0 0 0 0 0 0 1]
$X2$ = [1 1 0 0 0 1 0 0 0 0 0 0 0 0]
$X3$ = [0 0 0 1 1 1 0 1 1 0 1 1 0 0]
$X4$ = [q 0 0 1 1 0 1 1 0 0 0 0 1 q]
$X5$ = [0 0 1 (q-1) (q-1) 0 0 1 1 0 1 1 0 1]
$X6$ = [0 0 0 0 0 0 0 0 1 1 0 0 0 0]
Transitions invariants:
$Y1$= [1 1 (q-1) (q-1) 0 (q-1) q (q-1) 1 1 1]
$Y2$ = [1 1 (q-1) (q-1) (q-1) 0 q (q-1) 1 1 1].

For q = 3, the state class graph is composed of 28 state classes, 40 transitions. Due to page limits, this state graph is not presented in this paper.

From the analysis, the following properties are obtained.

• The system is structurally bounded as all places of the model are bounded, i.e. the set of all places is a conservative component.

• The system is consistent and structurally repetitive, i.e. there is a T-invariant strictly non negative Y, such that W.Y = 0 (where W is the incidence matrix of the model). Thus, the Petri net negotiation model has the non blocking property.

• The system is reversible (or reinitializable) for the initial marking of 1 token in place p_1 (order agent available) and q tokens in places p_9 (number of available resources), all other places being empty. In fact, the system is bounded and the corresponding class graph is strongly connected.

• The presence of initial tokens in each place invariant guarantees the non-blocking property.

• Given the structure of our model, a time bound of the negotiation process can be directly obtained from the worst case execution of the model in Fig.3. The time boundedness property of the negotiation protocol is thus established by considering in the longest possible path for reinitialization in the state class graph. For the *1 order - q resources* negotiation protocol analysed in this section, a global time bound B is given by:

$$B = (q-1) * t_3' + t'_{6\max} + t'_{8\max}. \quad (5)$$

5 CONCLUSIONS

Multi-agent execution systems have become popular in the context of distributed manufacturing systems, where reactivity and flexibility are critical issues. Negotiation is certainly the most appealing concept for achieving efficient coordination of task execution in such systems. However, very few studies have tackled the problem of verifying the required properties of such negotiation protocols, the main difficulty being in the evaluation of the outcome and time duration of complex tasks, in particular the ones involving execution of computer programs. The Petri net models proposed in this study rely on the approximation of the duration of computational tasks by time intervals. A simple automated negotiation protocol has been proposed and verified with respect to liveness, reversibility, structural and time boundedness.

REFERENCES

Berthomieu B. and M. Diaz, 1991. Modeling and verification of time dependant systems using time Petri nets. *IEEE Transactions on Software Engineering*, **vol. 17**, n°3, pp.259–273.

Berthomieu B. and M. Menasche, 1982. A state enumeration approach for analysing time Petri nets. Proc. ATPN 82, pp. 27–56.

Blanc P., I. Demongodin, P. Castagna, 2006. A holonic approach for manufacturing control: an industrial application. Proc. IFAC-INCOM'06, pp. 389-394.

Fox M. S., M. Barbuceanu, and R. Teigen, 2000. Agent-Oriented Supply-Chain Management. *Int. Journal of Flexible Manufacturing Systems,* **vol. 12**, pp.165–188.

Jennings N. R., P. Faratin, A. R. Lomuscio, S. Parsons, C. Sierra, M. Wooldridge, 2001. Automated negotiation: prospects, methods and challenges. *Int. Journal of Group Decision and Negotiation*, **vol. 10**, n°2, pp.199-215.

Jensen, K, 1992. Coloured Petri Nets, Basic Concepts, Analysis Methods and practical Use. Vol. 1: Basic Concepts, 1992. Vol. 2: Analysis Methods, 1994. vol. 3: Practical Use, 1997. Monographs in Theoretical Computer Science, Springer-Verlag.

Merlin P. M., 1974. A study of the recoverability of computing systems. PhD thesis, U. California, Irvine.

Odell J., H. Van Dyke Parunak and B. Bauer, 2000. Representing agent interaction protocols in UML. In P. Ciancarini & M. Wooldridge (eds), Proceedings of First International Workshop on Agent-Oriented Software Engineering, Springer-Verlag, pp. 121-140.

Osborne M.J. and A. Rubinstein, 1994. *A Course in Game Theory*. The MIT Press.

Ounnar F. and P. Pujo, 2005, Supplier evaluation process within a self-organized logistical network. *Int. Journal of Logistics Management,* **vol. 16**, n°1, pp. 159-172.

Pesenti R., F. Rosi, S. Salvador and W. Ukovich, 2006. An agent-based integrated resource control system. Proc. MITIP2006, Budapest.

Shen W., L. Wang and Q. Hao, 2006. Agent-based distributed manufacturing process planning and scheduling. *IEEE Trans. Syst., Man and Cybernetics*, Part C, **vol. 36**, n°4.

Smith R. G., 1980. The contract net protocol: High-level communication and control in a distributed problem solver. *IEEE Trans. Computers*, **vol. 29**, pp.1104-1113.

Van Brussel H., J. Wyns, P. Valckenaers, L. Bongaerts, P. Peeters, 1998. Reference architecture for holonic manufacturing systems: PROSA. *Computers in Industry* **vol. 37**, pp. 255–274.

Vernadat F., 2001.UEML: Towards a Unified Enterprise Modelling Language. Proc. MOSIM'01, pp. 3-9.

Zambonelli F.; N. Jennings and M. Wooldridge, 1994. Organizational rules as an abstraction for the analysis and design of multi-agents systems. *Int. Journal of Software Engineering and Knowledge Engineering*.