ELSEVIER

Computers in Industry 38 (1999) 103–113

COMPUTERS IN INDUSTRY

A distributed architecture and negotiation protocol for scheduling in manufacturing systems

Paulo Sousa *, Carlos Ramos [1]

Instituto Superior de Engenharia do Porto (ISEP / IPP), Departamento de Engenharia Informática, Rua São Tomé, 4200 Porto, Portugal

Abstract

This paper deals with a new architecture and negotiation protocol for the dynamic scheduling of Manufacturing Systems. The architecture is based on two paradigms: Multi-Agent Systems and Holonic Systems. The main contribution in the architecture is the development of Holons representing tasks and resources. The well-known Contract Net Protocol [R. Davis, R. Smith, Negotiation as a metaphor for distributed problem solving, Artificial Intelligence, 20, (1), (1983) 63–109] has been adapted to handle temporal constraints and to deal with scheduling conflicts. The purpose of this protocol is to dynamically assign operations to the resources of the Manufacturing System to accomplish the proposed tasks. This protocol involves a renegotiation phase whenever exceptions appear. It also deals with conflict situations, namely, with the case of the 'indecision problem'. The used approach assumes that deadlines are the most important constraints to consider. Thus the acceptance or refusal of a resource for a specific operation depends on the capability of executing the operation within the specified deadline.

Keywords: Intelligent Manufacturing Systems (IMS); Holonic Manufacturing Systems (HMS); Co-operation; Dynamic scheduling

1. Introduction

During recent years, it has been necessary to respond to emerging trends in the Manufacturing Systems. These new directions include the reduction of the dimensions of orders as well as the reduction of life cycles of products. Thus, Manufacturing Systems need to handle the dynamic nature of demands. In fact, in the dynamic scheduling, solutions need to be rapidly achieved. Scheduling of Manufacturing Systems corresponds to a distributed problem from the physical and from the logical point of view. Physically, the Manufacturing System involves several resources (NC machines, robots, AGVs, conveyors, etc.) and from the logical point of view it is also a distribution problem, because several tasks [2] can be carried out at the same time. Due to these reasons the framework of distributed artificial intelligence (DAI) and Holonic Manufacturing Systems for dynamic scheduling of industrial tasks is proposed.

The paper presents a new architecture and Negotiation Protocol for dynamic scheduling of Manufacturing Systems. It is made under the premise that due dates (deadlines) for the tasks to be met. Section 2

* Corresponding author. E-mail: psousa@dei.isep.ipp.pt
[1] E-mail: csr@dei.isep.ipp.pt.

[2] Task means a set of operations to perform resulting in a final item (e.g., make a chess pawn). An operation is a small part of a task, for example, in the 'make a chess pawn' task, one operation is drilling or painting.

0166-3615/99/$ - see front matter
PII: S0166-3615(98)00112-2

briefly describes some Distributed Intelligence concepts. This section also presents the concept of Holonic Manufacturing Systems (an extended explanation of these concepts was given by Sousa and Ramos [13]. Section 3 describes a new architecture for the dynamic scheduling of Manufacturing Systems and Section 4 illustrates how the Negotiation Protocol is established and presents the renegotiation phases for handling exceptions. Finally, conclusions are presented in Section 5.

2. Multi-Agent Systems and Holonic Manufacturing Systems

Two scientific communities are developing work related with Intelligent and Distributed Systems for Manufacturing. For Artificial Intelligence researchers, the Distributed Artificial Intelligence is the solution for distributed problems. On the other hand, some researchers from the Manufacturing area prefer to use the Intelligent Manufacturing System concept, where Holonic Manufacturing Systems is presented as the solution for distribution of tasks. As it is usual in these situations similar concepts appear with different names.

2.1. Distributed Artificial Intelligence and Multi-Agent Systems

During the last two decades, efforts have been made for a continuous enhancement of Intelligent Systems to deal with increasingly complex problems. Such systems soon became too heavy to be implemented as a centralised control program in only one computer. Fortunately, Distributed Computing has already given some answers to the problem of how to efficiently implement communities of interactive systems. A new research area appeared to cover the problem posed by the integration of Artificial Intelligence and Distributed Computing. This area was Distributed Artificial Intelligence. DAI was firstly identified as a kind of problem solving where the computation and inference were distributed in the logical or physical sense [8]. Davis [4] stated that DAI is concerned with those problems for which a single problem solver, single machine or locus of computation seems inappropriate.

DAI methodologies also lead to two different approaches for Intelligent Systems distribution: Distributed Problem Solving (DPS) and Multi-Agent Systems (MAS). In DPS, there is a set of modules or nodes co-operating to solve a specific problem. The knowledge about the problem and its solution is divided among all nodes of the system. In MAS the distinction between problem solving and co-operation is much clearer. The attention is on the co-ordination process between intelligent autonomous agents. The negotiation between different Agents is one of the most important problems to solve in Distributed Intelligent Systems. Whenever several systems, or resources, are able to execute the same tasks or operations a negotiation protocol establishing relationships between contractors and possible contracted Agents is very important (e.g., Contract Net Protocol, [5])

2.2. Intelligent Manufacturing Systems and Holonic Manufacturing Systems

Intelligent Manufacturing Systems (IMS) in general and Holonic Manufacturing Systems in particular are new concepts in the Manufacturing arena. Solberg and Kashyap [12] analysed the results of some independent studies on the changes that are rocking the manufacturing world. Among the trends which virtually all observers have noted are: globalisation; increasing complex and competitive markets; shortened product life-cycles; increasing requirements for quality; increasing customisation of products; faster paced advances in technology; rapidly expanding options in materials and processes; and increasing skill requirements. Additionally, one must consider the effect of environmental constraints forcing changes in products and processes and the cost of energy.

The new trends on manufacturing gave origin to a new concept: Intelligent Manufacturing Systems. Initially, IMS was a vision of a future manufacturing scenario [15]: ‘‘The Intelligent Manufacturing System takes intellectual activities in manufacturing and uses them to better harmonise human beings and intelligent machines integrating the entire corporation from marketing through design, production and distribution, in a flexible manner which improves productivity.’’ IMS will possess the innate ability to

respond, promptly and correctly, to changes in requirements. They differ from conventional Manufacturing Systems—even advanced ones—in their inherent capability to adapt to changes without external intervention.

The underlying idea behind Holonic Manufacturing Systems (HMS) is to provide a dynamic and decentralised manufacturing process, in which humans are effectively integrated, so that changes can be made dynamically and continuously. HMS are based on the notion of Holon [6]—combination from Greek *holos* (whole) with the suffix *on* which, as in prot*on* or neutr*on*, suggests a particle or part. A Holon means simultaneously a whole and a part of the whole. Thus, a Holon can be made up of other Holons. This feature distinguishes holons from agents. A Holon is autonomous and co-operative and sometimes intelligent. Each production unit (e.g. NC machine) can be a Holon and these Holons co-operate with each other in order to manufacture products. As it was observed by Valckenaers et al. [14], it is expected that rigid, static and hierarchical Manufacturing Systems will give way to systems that are more adaptable to rapid change. As an alternative to hierarchy it appears the concept of holarchy as a system of holons that can co-operate to achieve a goal or objective. A holonic Manufacturing System is a holarchy that integrates the entire range of manufacturing activities from order booking through design, production, and marketing to achieve the agile manufacturing enterprise. See Ref. [13] for a glossary of HMS concepts/properties and Refs. [1,2,7] for some examples of systems that have used holonic approaches in industrial scenarios.

3. Holonic architecture for Manufacturing Systems

There can be several holarchies in a Manufacturing System. However, this paper will focus principally on the Scheduling Holon.

3.1. The holonic architecture

Fig. 1 illustrates the proposed holonic architecture for Manufacturing Systems. The attention of this architecture will be focused in the areas of Process and Production Planning.

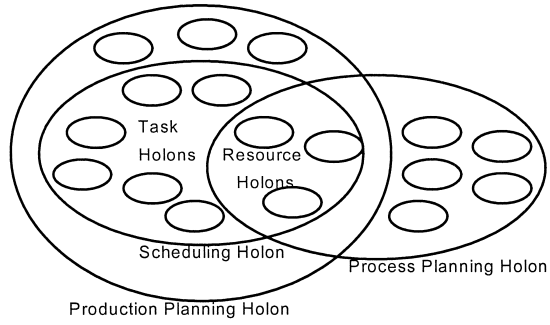


Fig. 1. The Holonic architecture.

This figure intends to illustrate part of a Manufacturing System holarchy and it is important for understanding the Holon concept. In Fig. 1 some resources (machines, robots, etc.) and tasks (manufacturing orders) are grouped in a Scheduling Holon (later, it will be defined how these holons co-operate for scheduling orders in a Manufacturing System).

Two important attributes of holons are depicted in this figure:

1. A Holon can be made up of others holons. For instance the Scheduling Holon can be grouped to other holons (e.g. Stock Management Holon, etc.) to define the Production Planning Holon; and
2. Holons can be part of several holons. For example, the Resource Holons of Fig. 1 are members of the Scheduling Holon and of the Process Planning Holon.

Clearly this kind of architecture is much more flexible than the static and traditional CIM architectures. However, a great effort must be put in the co-ordination between holons to maintain the overall systems coherence.

3.2. The scheduling holon

Bongaerts et al. [3] identified three types of basic building blocks—holons—in a HMS: (i) product holons, (ii) resource holons, and (iii) order holons. The proposed architecture also involves Resource Holons, but deals with manufacturing tasks instead of orders. The Product Holon is part of the Process Planning Holon and not of the Scheduling Holon. Fig. 2 represents the architecture of the Scheduling Holon and its components.

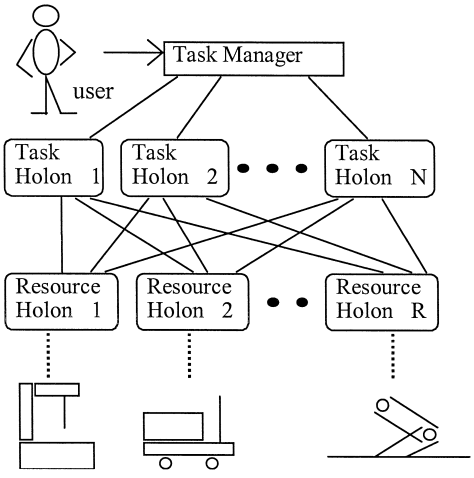


Fig. 2. Scheduling Holon's architecture.

The different components involved are listed below.

- *Resources* are the basic components of the Manufacturing System (robots, NC machines, conveyors, etc.) or they can be cells made up of basic resources. A Holon represents each resource and each cell (still in this case there is a holon for each resource in the cell). The number of resources does not vary, except when resources are added to or removed from the Manufacturing System.
- A *Resource Holon* represents the current situation of one resource (its status; delivered activity; and activity yet to be done). The activity of the resource is represented into an agenda. The agenda is the sequence of operations to be carried out and it specifies the expected duration for these operations as well as the free time intervals of the resource [10]. The activity of a resource changes according to the operation of the resource and according to the dynamic situation of the Manufacturing System (new orders, failures and delays in other resources, etc.).
- *Task Holons* represent the possibilities to execute a plan for a task into the plan structure [9]. Once launched, they directly negotiate with the appropriate Resource Holons. A Task Holon has a birth and death. During its existence each of these holons is responsible for monitoring the progress of the work, and act accordingly if anything unpredictable happens. The Task holon will have a Control System with a monitor–detect–react loop for process monitoring and control.
- The *Task Manager* is the user interface to the system. It receives orders for new tasks. This Holon is responsible for launching Task Holons whenever a new task is ordered. Besides, the Task Manager is responsible for dealing with dynamic changes of task conditions (e.g. when the user changes the deadline of a task).

Each holon has a *Constraints module* that handles some knowledge and heuristics from production for each special implementation. This module helps finding answers for questions like ''if there are two identical production units that only differ in the amount of time they take to perform a task (a new machine was bought but the old one is still running), which one is best to choose?''

4. Negotiation protocol

The proposed architecture supports a negotiation protocol between all parts of the system. This protocol is adapted to handle conflicts (mainly the indecision problem) and dynamic changes in the system.

4.1. Negotiation phase

The example used during this section describes a system with three resources—A, B and C—and one task consisting of tree operations—op1, op2 and op3—that must be done sequentially. Resource A can execute op1; Resource B op2; and Resource C both op2 and op3.

Fig. 3 illustrates the first two steps of the Negotiation Protocol for contracting resources for the execution of a task.

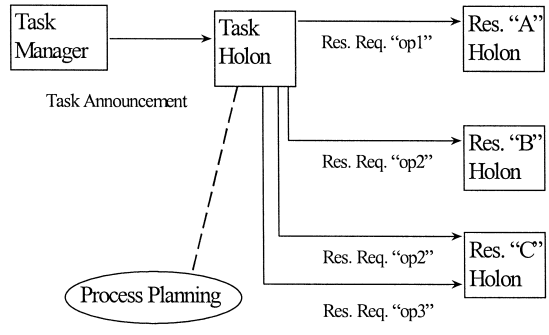


Fig. 3. Task announcement and resource request.

When a new request for task execution appears, the user specifies the name of the task to be carried out as well as its deadline. The Task Manager receives information from the user and will inform the task Holon about the deadline and other constraints of the task. The format [3] of the request for task execution is as follows:

(NoP, Item, TimeWindow, {Const}, TD)

where:

- *NoP* is the number of parts to manufacture;
- *Item* identifies what is to manufacture;
- *TimeWindow* is the time interval in which the scheduling makes sense (the superior limit is assigned based on the due date for the manufacturing order);
- *Const* represents a constraint imposed to the task (e.g. precision, repeatability, compliance, stability, uncertainty); and finally
- *TD* is the task descriptor for identifying the task.

This message is sent from the Task Manager to the Task Holon and it will be referred as the *Task Announcement Message*.

After receiving the announcement from the task manager, the first thing the Task Holon does is to get the basic plan—and some alternative plans too—for the manufacturing of the requested item. This is done by interfacing the Scheduling Holon with the Process Planning Holon. Plans are generated by a system called TPMS (Task Planning for Manufacturing Systems) [11]. TPMS automatically generates a high level plan for a task given its precedence graph.

The task Holon receives a message with the following information:

(BPlan, {Resource}, {BC}, {AltPlan, {AltRes}})

where:

- *BPlan* is the basic plan for executing one sequence of operations;
- *Resource* is a resource needed for this plan;
- *BC* is a basic component this task uses;
- *AltPlan* is an alternative plan for this task;
- *AltRes* is a resource used in the alternative plan.

[3] The following notation is used in message's format: () group or structure {} list/repetition.

A plan is a list of nodes, each node having information about an operation (name, resource, duration, tools and components).

At this moment, the Task Holon knows the sequence of resource operations to execute. The next step in the Negotiation Protocol is the proposal of operations to the resources or cells. However, to guarantee the deadline, the negotiation between the Task Holon and the Resource Holons will have two phases: the forward influence phase and the backward influence phase. The algorithm for scheduling has been adopted from a method developed for centralised dynamic scheduling that is well described by Ramos et al. [10].

The Task Holon will contact all resources able of performing each operation with the *Resource Request Announcement* message. This message has the following format:

(NoO, Op, TimeWindow, {Const}, TD, {PrevOp, {ResPrevOp}}, {NextOp, {ResNextOp}})

where:

- *NoO* is the number of operations to be executed;
- *Op* is the operation itself;
- *TimeWindow* is the initial time interval for scheduling;
- *Const* is a constraint;
- *TD* is the task descriptor;
- *PrevOp* is an operation that must be done prior to this one;
- *ResPrevOp* is a resource contacted for the previous operation;
- *NextOp* is an operation which Op precedes; and
- *ResNextOp* is a resource contacted for the next operation.

Once a Resource Holon receives a Resource Request Announcement it analyses its agenda marking time intervals where it can execute the requested operation (see Fig. 4).

The Resource Holon(s) contacted for the operation(s) with no predecessors will start the forward influence phase. This phase consists of 'influencing' the list of time intervals for the next operation with the resulting list from the previous operation(s)—the list of free time intervals of this resource is considered as the TimeWindow for the other operations that it precedes. Fig. 5 illustrates this step of the negotiation.

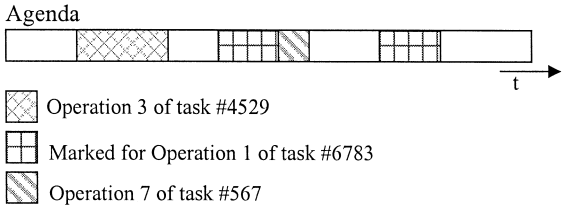


Fig. 4. Resource Holon's agenda.

Thus each Resource Holon has to wait for a message from each Resource Holon contacted for the operation(s) preceding it. For each *List of Intervals* message, the Resource Holon will generate an 'influenced' list of intervals it will pass to the Resource Holons of the next operation(s). The List of Intervals message has the following format:

(Res, Op, {TimeInterval}, TD, Dir)

where:

- *Res* is the sender resource;
- *Op* is the operation;
- *TimeInterval* is a time interval in which the operation could be done (this list is important for the scheduling algorithm of the resources [10]);
- *TD* is the task descriptor; and
- *Dir* is the direction (forward or backward)

This phase stops when a Resource Holon has no operations after it.

The backward influence phase starts now. The Resource Holon(s) of the last operation(s) will 'influence' the list of intervals of the operation(s) preceding it. The List of Intervals message is used to pass the list of intervals.

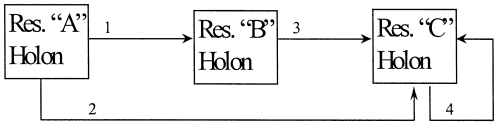
Each Resource Holon has to wait for a list of intervals from the Resource Holons contacted for operations after it. The Resource Holon will use this list to generate the final List of Intervals for this particular operation. This list is passed to the Resource Holons contacted for operations preceding this one. The backward influence phase ends when the Resource Holons for the first operation(s) receive the list of intervals from the operations proceeding them (see Fig. 6).

When each Resource Holon has the final list of intervals, it responds to the Task Holon with a message in the following format:

(Res, Op, {TimeInterval, FreeTime TillEnd}, TD)

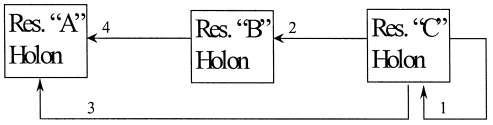
where:

- *Res* is the sender resource;
- *Op* is the operation to schedule;
- *TimeInterval* is a possible time interval for scheduling that operation in this resource;
- *FreeTime TillEnd* is the amount of time in which the resource is free till the specified deadline if the Task Holon selects this time interval; and
- *TD* is the task descriptor.



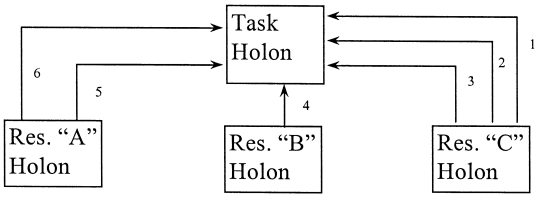
1 - Forward Influence of Op1 in A to Op2 in B (A - B)
2 - " " of Op1 in A to Op2 in C (A - C)
3 - " " of Op2 in B to Op3 in C (A - B - C)
4 - " " of Op2 in C to Op3 in C (A - C - C)

Fig. 5. Forward influencing the list of intervals of operations.



1 - Backward Influence of Op3 in C to Op2 in C (A - C - C)
2 - " " of Op3 in C to Op2 in B (A - B - C)
3 - " " of Op2 in C to Op1 in A (A - C - C)
4 - " " of Op2 in B to Op1 in A (A - B - C)

Fig. 6. Backward influencing the list of intervals of operations.



1 - Resource "C" bid for Op3 (A - C - C)
2 - Resource "C" bid for Op3 (A - B - C)
3 - Resource "C" bid for Op2 (A - C - C)
4 - Resource "B" bid for Op2 (A - B - C)
5 - Resource "A" bid for Op1 (A - B - C)
6 - Resource "A" bid for Op1 (A - C - C)

Fig. 7. Resource bid for operations.

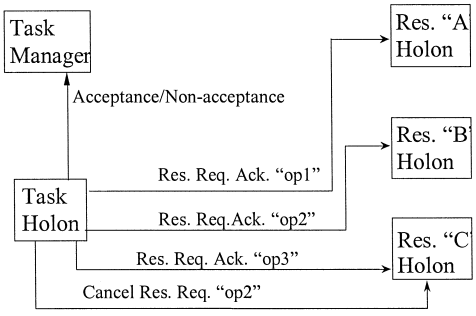


Fig. 8. Ending the negotiation.

This message will be referred as *Resource Bid* Message (see Fig. 7).

After receiving all Resource Bid Messages the Task Holon selects one Resource Holon that is able to handle each operation. This selection is based on heuristic rules (e.g. the resource having more free time till the deadline). If the co-ordination between resources is not possible, the Task Holon tries to change the selection of Resource Holons for the operations. If a solution can be found an *Acceptance* Message is sent to the Task Manager, otherwise it is sent a *Non-Acceptance* Message. Before sending the Acceptance Message the Task Holon will notify resource holons that are not contracted with the *Cancel Resource Request* Message. It will also confirm the execution of the operations to the selected Resource Holons with a *Resource Request Acknowledgement* message. Fig. 8 illustrates the final steps of the Negotiation Protocol.

The Resource Request Acknowledge message has the following format:

(Op, SelTimeInterval, TD)

where:

- *Op* is the operation;
- *SelTimeInterval* is the selected time interval for the operation;
- *TD* is the task descriptor.

When a resource Holon receives a confirmation of an operation it will mark the selected time interval as occupied and clears any marks it made for that operation in other time intervals. Similarly, if the resource Holon receives a Resource Request Cancel message it will clear all the marks it made for that operation (see Fig. 9).

This protocol serves the purpose of contracting resources for the execution of a task, but how to guarantee no deadlocks and conflicts between tasks and resources when there are several tasks negotiating at the same time?

4.2. The indecision problem

This protocol by itself does not avoid conflicts when multiple tasks negotiate with the same resources at the same time. Suppose a situation in which there are two tasks (T1 and T2) and 2 resources (RA and RB). T1 starts the negotiation of operation op1 with RA and RB, imposing a deadline for it. RA analyses the deadline for op1 and concludes that there is a free time interval in which it is possible to guarantee the execution of op1 without violating the deadline. Thus RA informs T1 of this free time interval. However, when T1 starts the analysis of the bids from RA and RB to decide which resource will take care of op1, RA receives a new request announcement from T2 for another operation—op2. Suppose that the only way to guarantee the deadline for op2 is to use part of the time interval previously obtained for op1. Now, RA has an indecision, since T1 has not still acknowledged RA to execute op1 (notice that T1 may choose RB).

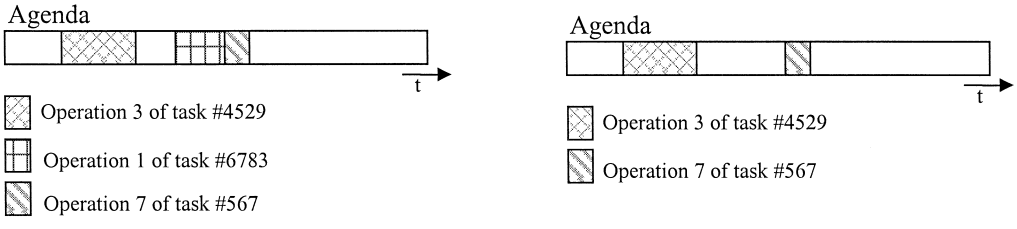


Fig. 9. Resource Holon's agenda (a) operation contracted; (b) operation cancelled.

Two inconvenient scenarios that may appear are as follows.

(A) RA considers the time interval as reserved for op1 and informs T2 that the execution of op2 is not possible in the specified deadline. This decision is bad if T1 does not contract RA for the execution of op1, since in this case RA could execute op2 in the deadline

(B) RA considers the time interval for op1 as free and informs T2 that the execution of op2 is possible in the specified deadline. This decision is bad if both, T1 and T2, acknowledge RA to execute op1 and op2. In this case one of the operations, op1 or op2, will not be accomplished in the specified deadline.

This kind of problem is called the 'Indecision Problem.' The solution for the 'Indecision Problem' can be the negotiation task by task or to have some care to avoid conflicts when supporting multi-task negotiation.

4.3. Changes in the negotiation protocol for conflict avoidance

As explained earlier, the Task Manager is responsible for launching Task Holons. However, there is another role for the Task Manager, so that it will help prevent the Indecision Problem.

The Task Manager will not launch Task Holons immediately after receiving requests from users. It maintains a priority list of waiting tasks. The next Task Holon to be launched is the one with highest priority and that does not conflict with other Tasks Holons that are still negotiating with Resource Holons. The possibility of a conflict is detected when a task needs a resource in part of the same time window that another task is currently negotiating. Task's priority is a function of the customer, the value of the order and the due date.

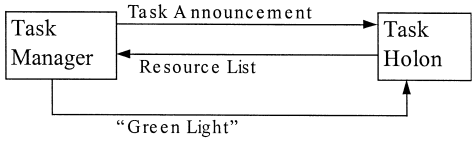


Fig. 10. New steps in the negotiation protocol.

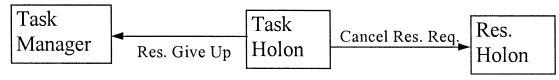


Fig. 11. Performance optimisation.

Fig. 10 shows the first change in the Negotiation Protocol. In order to avoid the Task Manager to know all the resources each task needs, each Task Holon—after being announced what to do—sends a message to the Task Manager with the list of resources it will negotiate. This is called the *Resource List* message. The Task Holon then waits for a signal from the Task Manager to proceed negotiation—the *Green Light* message. The Task Manager keeps a list of resources being negotiated and sends the 'Green Light' message if and only if there is no intersection between this list and the Task Holon's list. Note that if the time windows do not intersect the indecision problem does not arise.

Notice that there may be more than one task in negotiation at the same time and still the indecision problem do not arise. There are no simultaneous conflicting tasks, but there may be simultaneous tasks negotiating different resources.

The Task Holon can decide that a specific resource is no longer important—other resources have a better behaviour—for the task under negotiation. To improve the efficiency of the negotiation process, the Task Holon sends *Resource Give-Up* messages to the Task Manager (as shown in Fig. 11).

In this way, the Task Manager will receive information on availability of resources before the end of the negotiation. Then, other tasks, blocked by the non-availability of a resource, may start their negotiation process. The task manager will check its list of

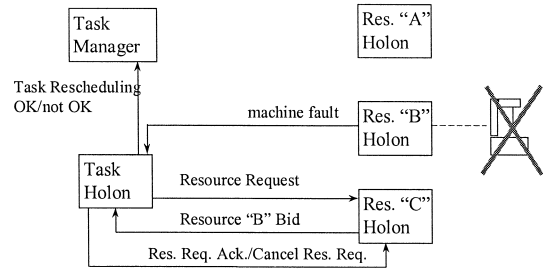


Fig. 12. An example of renegotiation.

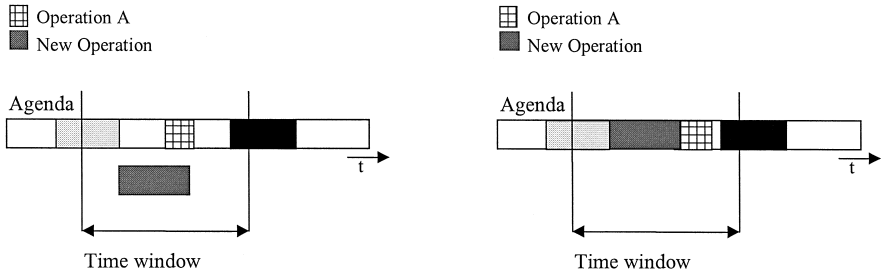


Fig. 13. Resource agenda: (a) no free interval for the scheduling of the requested operation; (b) reallocation of an already scheduled operation to give space for the new one.

waiting Task Holons whenever it receives a 'Resource Give-Up,' 'Acceptance' or 'Non-Acceptance' message to see if it can give green light to any of them.

4.4. Renegotiation phase

Ramos et al. [9] initially described this Renegotiation Protocol. Fig. 12 illustrates an example of this renegotiation phase.

When the Resource Holon detects a malfunction that it cannot recover it decides to inform every Task Holon that have contracted its work. Notice that the Resource Holon may analyse the situation and decide not to inform the Task Holons, then it will try to recover from the malfunction and execute its work with a slight delay. After receiving the *Machine Fault* message from the Resource Holon, the Task Holon will begin negotiations with other Resource Holons capable of performing the operations it needs. This negotiation has the same steps explained earlier. The Task Holon communicates the result of this negotiation to the Task Manager with a *Task Rescheduling OK* or *Task Rescheduling Not OK* message.

Another situation where renegotiation exists is when a rush order comes in or the due date of an already scheduled task is changed. The protocol is the same as explained in Section 4 except for a few changes when there is no free time intervals in the imposed time window. Suppose there is a resource with the agenda of Fig. 13a. This resource receives a request for an operation that takes longer than any free interval in the given time window. Taking into account the priority of each task it decides to reallocate operation A to free some time, in order to execute the new requested operation (see Fig. 13b).

This change may result in a delay for the task to which operation A belongs, or it may not if that task has some free time in its own time window.

This situation may involve a more drastic reallocation like the one in Fig. 14. This kind of change involves a negotiation task to task in which the new Task Holon has to persuade (or bargain, may even

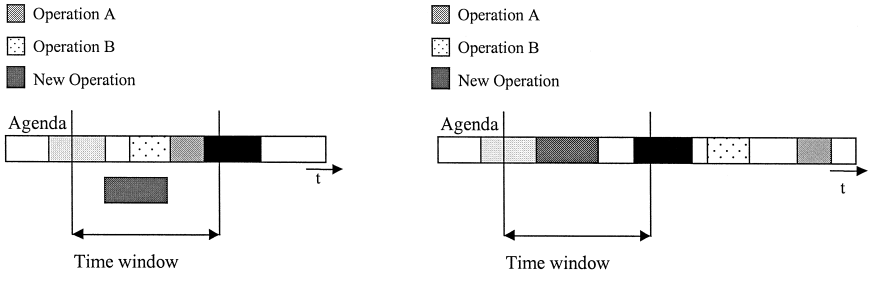


Fig. 14. Rescheduling for a new operation: (a) the request; (b) reallocation.

demand) the other task holons to give him the resource time it needs.

5. Conclusions

This paper has presented holonic architecture for the dynamic scheduling of Manufacturing Systems. It also presented a Negotiation Protocol based on the Contract Net Protocol and suitable for the dynamic scheduling of manufacturing tasks. This Negotiation Protocol is able to deal with deadlines and exceptions (since a Task Holon can start a renegotiation phase of the protocol if it is a high priority task just being launched or if its priority changes. It will try to persuade other Task Holons to give it the time it needs in the resources). While other co-operative communities operate with Agents representing resources or systems, the architecture proposed here combines Resource-based Holons with Task-based Holons. The main advantage is the easy access to task activities that are supported in Task-based Holons. In some other approaches the resources are not certain to be contracted when they offer their bids. This means that when a Resource Holon receives a new request, it does not know if the previously announced operations are to be contracted or not when several tasks are being contracted at the same time ('indecision problem'). This will lead to some uncertainty in the scheduling of the operations. The 'indecision problem' is solved without the need to negotiate task by task.

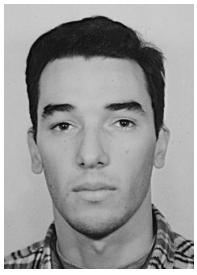
Future work will try to explore some new directions. In real-life Manufacturing Systems the failure in accomplishing a deadline does not imply to abort the task. Sometimes, the delay in the due date for a task leads to an additional cost (fines imposed by the consumer or who required the task). This will imply to have a second type of negotiation between Task Holons in order to achieve a compromise.

Acknowledgements

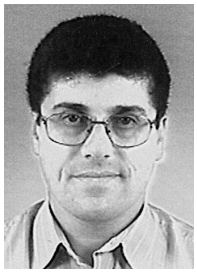
This work is partially supported by FLAD (Project 443/93), JNICT (Projects PBIC/TPR/2551/95 and PBIC/TPR/2556/95) and ESPRIT (IMS-WG Project 21955).

References

[1] J. Agre, G. Elsley, D. McFarlane, J. Cheng, B. Gunn, Holonic control of cooling control systems, Rensselaer's 4th Int. Conference on Computer Integrated Manufacturing and Automation Technology, 1994.
[2] L. Bongaerts, P. Valckenaers, H. Van Brussel, J. Wyns, Schedule execution for a holonic shop floor control system, Proceedings of the Advanced Summer Institute on Intelligent Control and Integrated Manufacturing Systems, Lisbon, Portugal, 1995.
[3] L. Bongaerts, J. Wyns, J. Detand, H. Van Brussel, P. Valckenaers, Identification of Manufacturing Holons, European Workshop on Agent Oriented Systems in Manufacturing, Berlin, Germany, 1996.
[4] R. Davis, Report on the workshop on Distributed Artificial Intelligence, SIGART Newsletter 73 (1980) 42–52.
[5] R. Davis, R. Smith, Negotiation as a metaphor for distributed problem solving, Artificial Intelligence 20 (1) (1983) 63–109.
[6] A. Koestler, The Ghost in the Machine. Hutchinson, London, 1967.
[7] D. McFarlane, B. Marett, G. Elsley, D. Jarvis, P. Wilbers, Application of Holonic Methodologies to Problem Diagnosis in a Steel Rod Mill, IEEE Int. Conference on Systems, Man and Cybernetics, 1995.
[8] N. Nilsson, Distributed Artificial Intelligence, Report SRI International, Menlo Park, CA, 1981.
[9] C. Ramos, A Holonic Approach for Task Scheduling in Manufacturing Systems, IEEE International Conference on Robotics and Automation, Minneapolis, USA, 1996.
[10] C. Ramos, A. Almeida, Z. Vale, Scheduling manufacturing tasks considering due dates: a new method based on behaviours and agendas, International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems; Melbourne, Australia, 1995.
[11] J. Rocha, C. Ramos, Plan representation and heuristic generation of operation sequences for manufacturing tasks, data and knowledge systems for manufacturing and engineering, Tempe, Arizona, USA, 1996.
[12] J. Solberg, R. Kashyap, Research in Intelligent Manufacturing Systems, Proceedings of the IEEE 81 (1) (1993) .
[13] P. Sousa, C. Ramos, Proposal of a Scheduling Holon for Manufacturing, PAAM97—The Practical Application of Intelligent Agents and Multi-Agent Technology. London, UK, 1997.
[14] P. Valckenaers, F. Bonneville, H. Van Brussel, L. Bongaerts, J. Wyns, Results of the Holonic Control System Benchmark at K.U. Leuven, Rensselaer's 4th International Conference on Computer Integrated Manufacturing and Automation Technology, 1994.
[15] H. Van Brussel, Working Group Proposal on Intelligent Manufacturing Systems, submitted to the Fourth Framework Programme of EC, 1995.



Paulo Sousa studied informatics at the Polytechnic Institute of Porto (ISEP/IPP), Portugal, where he graduated in Industrial Informatics. From 1993 to 1996, he was an application developer at a Portuguese Software house, in the area of client/server computing. He became an Assistant Professor at ISEP/IPP in 1996, and is currently a PhD student at Minho's University, Portugal, working in Holonic Manufacturing Systems. His main interests are Distributed Systems, Manufacturing and Computer Graphics. He can be reached by email psousa@dei.isep.ipp.pt and his URL is http://www.dei.isep.ipp.pt/~psousa.



Carlos Ramos received a Bachelor of Science degree in Electrical Engineering from the University of Porto (Portugal) in 1986. He received the Ph.D. degree from the same University in 1993. Currently, he is Co-ordinator Professor in the Institute of Engineering at the Polytechnic Institute of Porto (ISEP/IPP). He is Director of the CIM Centre of ISEP/IPP. His main areas of interest are Artificial Intelligence and Computer Integrated Manufacturing. He has more than 100 publications in these areas and co-ordinates several R&D projects. Readers can contact him at csr@dei.isep.ipp.pt.