FABRICARE: AN AGENT-BASED SYSTEM FOR MANUFACTURING

Paulo Sousa[1], Carlos Ramos[1], José Neves[2]

1- GECAD – Knowledge Engineering and Decision Support Group
Institute of Engineering – Polytechnic of Porto
Porto, Portugal
psousa@dei.isep.ipp.pt, csr@dei.isep.ipp.pt

2- Universidade do Minho
Braga, Portugal
jneves@di.uminho.pt

Abstract – Due to current and future trends in manufacturing, computer-supported manufacturing systems are now more distributed than the traditional CIM approach. This paper presents a prototype system for the problem of scheduling of manufacturing orders, focusing on the internal implementation of each type of agent in the system.

Keywords: holonic manufacturing system, scheduling.

I. INTRODUCTION

Society has experienced evolving mutations since the dawn of ages, especially on the last half of the twentieth century. In recent years several trends were observed in manufacturing and society, namely: market globalisation; increasing product/services customisation; increasing technology complexity; increasing number of competitors; decreasing product lifecycles; and increasing quality requisites [13] [20] [15]. Stability, certainty and predictability are giving place to change, uncertainty and unpredictability [11]. In addition, a shift towards flexible manufacturing and customized products is evident [3].

From a technological point of view, it was observed that current manufacturing systems (e.g. Computer Integrated Manufacturing architectures) pose several drawbacks, namely excessive rigidity and centralization; high implementation costs; and inflexibility [9] [31] [4] [16] [29] [8] [5] [10] [18]. Furthermore, it is expected that in the future manufacturing will be characterized by: globally distributed resources; small quantities and high variety of products; providing individual solutions tailored to each customer's specific needs; concurrent execution of all the activities in the manufacturing process [15] [6].

In order to deal with the identified problems with current manufacturing systems and prepare them for the expected future scenarios, the new generation of manufacturing systems must possess such attributes as decentralization, distribution, autonomy, adaptability, and incomplete information handling [27] [28].

II. FABRICARE'S DESCRIPTION

A. System Architecture

This work tries to present an integrated view of a manufacturing system, based on the Distributed Manufacturing System paradigm, able to accomplish the requirements for the post-XX century society [26]. In order to overcome these challenges, a solution based on Holonic Manufacturing Systems has been proposed using Multi-Agent Systems and Extended Logic Programming.

In this solution the main entities in the manufacturing process are modelled as holons [12], each one contributing with a small parcel of the overall system's functionality. To the scope of this work, a *holon* is understood as a logical design entity in the system architecture, and is implemented as an *agent* using extended logic programming. Thus, conceptually, a design entity in the architecture is a Holon while the software application that models that entity is an Agent.

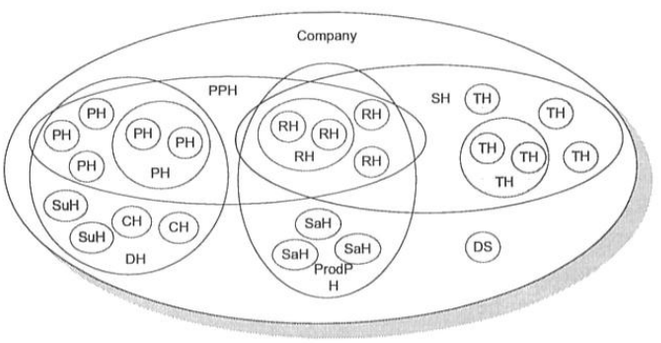


Figure 1 - The architecture of Fabricare

The *Fabricare* project uses a holonic architecture (Figure 1) where several key functions of the manufacturing process are identified and modelled as holarchies [23] [24] [21]. These holarchies are composed of 'basic' holons such as Product, Task and Resource, each one representing a core entity in the manufacturing system.

In the manufacturing arena there are several situations were the information needed is not fully available

(*i.e.*, incomplete information); thus, it was given a special attention to the representation and manipulation of incomplete information in an agent's knowledge base. For that reason, a notation based on [30] [2] [14] [21] is used for representing incomplete information in extended logic programs. This notation allows for: explicit negative information; unknown information; mutually exclusive information; and forbidden information. Additionally, a meta-interpreter was developed [21] to infer the truth-value of a question posed to the agent's knowledge base.

B. Interaction

The key entities participating in the scheduling process (activity chosen as test case) are the physical resources and the manufacturing orders. These two entities are represented in the system by resource holons and task holons respectively.

For the scheduling of task's sub-operations, the Task holon will negotiate with Resource holons, using an extension of the Contract Net Protocol [19] [7] with a cooperation phase between service providers (i.e. resource holons). The Resource holons will use constraint propagation in order to guarantee the relationships among different operations that aim at the same task. This new protocol is called *Contract Net with Constraint Propagation Protocol* (CNCPP) [22] [23] [24] [25] [21] (Figure 2).

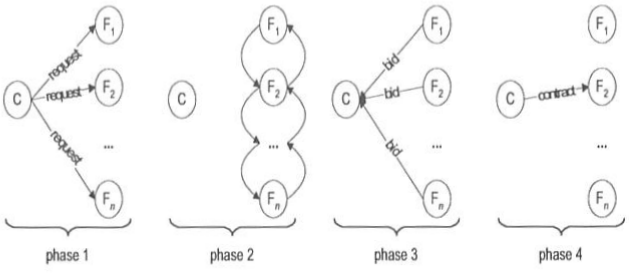


Figure 2 - Fabricare interaction

Since multiple tasks can be negotiated at the same time, conflicts may arise if some resources are used in the same time interval for different tasks. In these scenarios, resource holons have an indecision problem [23] since they cannot guarantee the delivery of both tasks. In order to overcome this problem, a solution is proposed which involves a pre-negotiation step in the protocol. Before beginning negotiation, each task holon will ask for authorization from the scheduling holon, which maintains a list of negotiating resources and respective time windows. Only in case of non-overlapping a "green light" will be given to the negotiation.

The scheduling procedure is adapted from a centralized method described in [17] [1], based on agendas, behaviours and due dates.

The number of exchanged messages is given by formula (1) [21] where the first term represents the number of exchanged messages in announcement, bid and contract phases and the second term represents the number of exchanged messages in backward and forward influence phases.

(1)

$$M' = 3 \times \left(\sum_{i=1}^{n} R(i) \right) + 2 \times \left(\sum_{i=1}^{n-1} \left(R(i) \times R(succ(i)) \right) \right)$$

Formula (1) is upper-bounded by expression (2) [21]:

(2) $M' \leq 3 \cdot n \cdot r + 2 \times (n\text{-}1) \times r^2 \Leftrightarrow M' \leq 3nr + 2nr^2 \Leftrightarrow$

$$M' \leq n \cdot (3r + 2r^2)$$

Thus, the complexity of the problem is *O(n)* in which respects the number of exchanged messages [21].

III. FABRICARE'S AGENTS

A. Task agent Internal architecture

A task holon represents a manufacturing order to execute a certain quantity of a specific product on the shop floor. This kind of holon has as its objective to schedule the order and monitor its execution.

The knowledge base of a Task agent has predicates of the following templates:

task(TId, NOF, Id-Product, Quantity, DueDate)
attribute(Parameter, Value)
plan(Id, Criteria, Attributes, Operations)

The predicate *task* represents the manufacturing order data such as order number (*NOF*), requested product (*Id-Product*), demanded quantity (*Quantity*) and due date (*DueDate*). Other task's attributes (*e.g.*, client id) are represented by means of the *Attribute* predicate. The predicate *plan* identifies the selected production plan for this task.

Its life cycle (Figure 3) begins when the manufacturing order is created (either to fulfil a customer order or to balance stocks). During its existence the task holon will negotiate with resource holons the execution of the operations needed to perform the ordered product. The holon will cease existing when the order is fulfilled or cancelled.

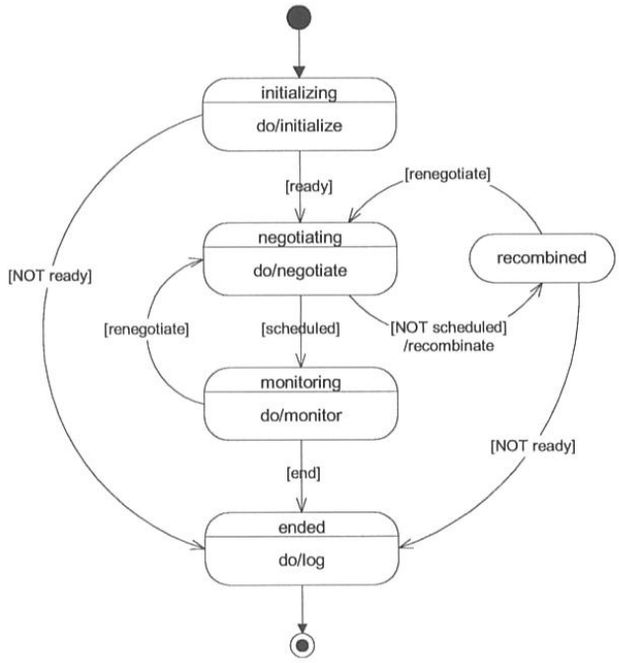


Figure 3 - task agent operation

Table 1 presents the identified cases of incomplete information for the predicates existing in each task agent's knowledge base [21].

Table 1 - resource agent incomplete information

Description	Type of null	predicate
There must exist a plan for this task	Closure	*plan*
The duration of an operation may be unknown	Unknown	*plan*
The value of an attribute of the task may be unknown	Openness	*attribute*

The knowledge base of every Task agent must be augmented with the following axioms:

negation(plan(I, C, A, O)) ←
not plan(I, C, A, O)
null(unknown_duration)
null(some_miliseconds)
null(some_seconds)
null(some_minutes)
exception_null(plan(I, C, A, Op1)) ←
plan(I, C, A, Op2) ∧
null_plan(Op1, Op2)
null_planl([X | T], [X | T2]) ←
null_plan(T, T2)
null_plan([node(I, H, E, _, P, S) | T], [node(I, H, E, Dur, P, S) | T2]) ←
null(Dur)
null_plan(_, _) ←
false

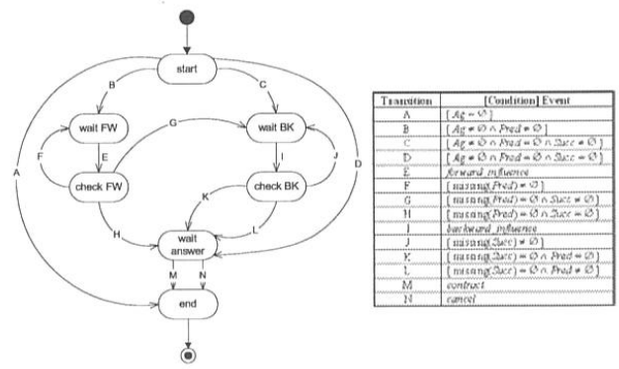
B. Resource agent Internal architecture

A resource holon represents the current state of a physical resource on the shop floor. The resource's list of activities is called agenda, stating what to do and when. The resource is able to perform operations necessary to execute products (e.g. drill). A resource holon can represent a single resource or a work cell composed of several resources.

The knowledge base of a Resource agent has predicates of the following templates:

resource(RId, Description, Creation-Date, Attributes)
ability(Id-Ability, Description, Duration, Cost, CNC, Arguments)
activity(TId, OpId, Qt, Start, End, DueDate, State, Pred, Succ)

The predicate resource represents general information about the physical resource in the manufacturing plant. The agent's KB has several instances of the predicate *ability*, one for each machining function the resource is able to perform (*e.g.*, drill). The predicates of type *activity* denote a commitment with a specific task agent (*TId*) to perform a specific operation (*OpId*) in the future; *i.e.*, the resource's agenda.



Transition	[Condition] Event
A	[Ag = ∅]
B	[Ag ≠ ∅ ∧ Pred = ∅]
C	[Ag ≠ ∅ ∧ Pred = ∅ ∧ Succ ≠ ∅]
D	[Ag ≠ ∅ ∧ Pred = ∅ ∧ Succ = ∅]
E	forward_influence
F	[missing(Pred) ≠ ∅]
G	[missing(Pred) = ∅ ∧ Succ ≠ ∅]
H	[missing(Pred) = ∅ ∧ Succ = ∅]
I	backward_influence
J	[missing(Succ) ≠ ∅]
K	[missing(Succ) = ∅ ∧ Pred ≠ ∅]
L	[missing(Succ) = ∅ ∧ Pred = ∅]
M	contract
N	cancel

Figure 4 – Resource agent operation

The objective of a resource holon is to control the physical equipment, provide information about its abilities and status to the system and manage the scheduled activities. Its life cycle is very long, since it is expected that a resource is fully operational for long periods of time. During its existence, the resource holon executes the commands sent by the resource controller and negotiates with task holons the scheduling of manufacturing orders.

Table 2 presents the identified cases of incomplete information for the predicates existing in each resource agent's knowledge base [21]

Table 2 - resource agent incomplete information

Description	Type of null	predicate
The resource holon knows all of its functionalities	Closure	*ability*
The resource holon knows all of its contracted activities	Closure	*activity*
The state of an activity may be unknown	Unknown	*activity*
The predecessor resources may be unknown	Unknown	*activity*
The successor resources may be unknown	Unknown	*activity*

The knowledge base of every Resource agent must be augmented with the following axioms:

negation(ability(I, D, P, A)) ←
not ability(I, D, P, A)
negation(activity(T, O, Q, I, D, Dt, E, P, S)) ←
not activity(T, O, Q, I, D, Dt, E, P, S)
exception_null(activity(T, O, Q, I, D, Dt, _, P, S)) ←
activity(T, O, Q, I, D, Dt, unknown_state, P, S)
exception_null(activity(T, O, Q, I, D, Dt, E, Pr1, S)) ←
activity(T, O, Q, I, D, Dt, E, Pr2, S) ∧
null_pred_succ(Pr1, Pr2)
exception_null(activity(T, O, Q, I, D, Dt, E, P, Su1)) ←
activity(T, O, Q, I, D, Dt, E, P, Su2) ∧
null_pred_succ(Su1, Su2)
exception_null(activity(T, O, Q, I, D, Dt, E, Pr1, Su1)) ←
activity(T, O, Q, I, D, Dt, E, Pr2, Su2) ∧
null_pred_succ(Pr1, Pr2) ∧
null_pred_succ(Su1, Su2)
null_pred_succ([X | T], [X | T2]) ←
null_pred_succ(T, T2)
null_pred_succ([_ | T], [unknown_holon | T2])
null_pred_succ(_, _) ←
false

Since each holon has complete knowledge of itself, it is necessary to close predicates *ability* and *activity*. However, it may be impossible to known certain characteristics of the holon's activities, such as the state of the activity (due to machine failure for instance) or the predecessors and successor resources in the chain of operations.

C. Prototype

Figure 5 presents the *Fabricare* Scheduling prototype suite, composed of several applications.

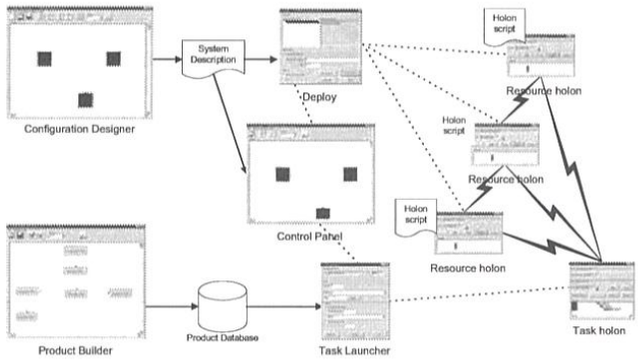


Figure 5 - Fabricare prototype

The *Configuration Designer* allows specifying the resource agents in the factory plant, and to some extent, represent graphically the physical layout of the resources. The system description is read by the *Deployment* tool, which launches *Resource Holons* on the desired machines. Each resource holon is composed of a kernel and individual characteristics and behaviours specified in the holon script written in Prolog representing the 'mental' state of the holon (e.g. resource's agenda), as well as specific clauses for the *Fabricare* holon kernel (e.g. name). The *Control Panel* is the interface to the system's operation, monitoring and controlling running holons. This tool also allows the user to launch tasks (manufacturing orders) in the system by evoking the *Task Launcher* tool, which prompts the user for data about the order and dynamically creates a *Task Holon* for that order. One last tool in the suite is the *Product Builder*, which allows to generate graphically a product's process plan. The several operations in the plan are the abilities of the physical resources (modelled in the resource holons).

The system is very dynamic in what concerns its holons, i.e. resource holons depend on the system description file; task holons depend on the existing tasks (dynamic events). Each negotiation uses the set of holons that are present and available at that time, thus giving the system a high degree of adaptability to the dynamic nature of the manufacturing arena (e.g. resource in maintenance or overloaded).

The holons are Extended Logic Programs written in Prolog with the ability to handle negative and incomplete knowledge [27]. The decision procedure is not yet totally driven by this kind of knowledge; however, real life scenarios where only partial information is available have been identified [27] [21] and modelled (e.g. resource holons will use this information to generate low commitment schedules into their agendas).

IV. Conclusions

The problem being addressed here is related to the ability to build and maintain computer-supported manufacturing systems able to cope with recent (and expected future) requirements, giving the social-economic context of the new digitised, customised, global society. In a way, the answer to this question can be given by

organising the manufacturing system into 'small' units structured according to the holonic theory.

Essentially, it is argued that in order to overcome the rising customisation of goods and services as well as other trends in society and economy, it is necessary to build systems with such characteristics as: distribution, decentralisation, autonomy, dynamism, reactivity, flexibility, adaptability and agility. Although current manufacturing systems do not present then, this kind of properties can be found in holonic systems. These were the principles and goals guiding the development of the *Fabricare* system.

This paper presented a system based on holonic concepts and a prototype implementation of that system that models the main entities in the manufacturing systems as holons and the core functions as holarchies;

This system resembles the distributed nature of manufacturing, thus allowing for a better modelling of the real system. While other co-operative communities operate with agents representing resources or systems, the *Fabricare* system 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 as well as high adaptability to the dynamic nature of resource conditions and availability.

For each agent in the system, the knowledge and control of each agent is separated through scripts; and the holons have the ability to represent and handle incomplete information in their knowledge bases.

The negotiation protocol presented to regulate the interaction among the several agents in the system, Contract Net with Constraint Propagation Protocol, which has as a main characteristic the explicit cooperation phase between service providers (i.e. resources) motivated by the need to coordinate temporal relations of task's operations;

The protocol also allows for dynamic participants, using the information of running agents stored in the directory service; and conflicts are avoided by serializing overlapping negotiation (concurrent time-windows for different resources/tasks);

The scheduling procedure (intimately related to the protocol just described) was adapted from a proven centralised method, but introduced some new characteristics, such as the use of a distributed approach and a cost function by operation by resource which allows a new parameter to obtain different solutions. The main advantage of this scheduling procedure is the fact that it allows the use of several resources for the same operation and allows for greater flexibility by not using behaviours as in the original method.

The paper also presented the knowledge base of the main agents in the system (Task and Resource).

One point of action for future work is the full use of incomplete information in the decision process of each agent. For the moment this use is still very limited. One possible added value of incomplete information is in the medium-long range planning.

Acknowledgments

The authors would like to thank FCT – "Fundação para a Ciência e Tecnologia" (Portuguese Foundation for Science and Technology) and programs POCTI and FEDER for their support in R&D projects. Acknowledgements are also given to the Scientific Council of ISEP/IPP and to FLAD – "Fundação Luso-Americana para o Desenvolvimento" (Portuguese-American Foundation for the Development). Additionally, we acknowledge the European Commission that supported some ESPRIT and IST programmes in which we were involved.

References

[1] ALMEIDA, A., 1995, Escalonamento Dinâmico de Tarefas Industriais Sujeitas a Prazos de Entrega. Dissertação de Mestrado. Faculdade de Engenharia da Universidade do Porto. (MSc Thesis in Portuguese)

[2] ANALIDE, C. and NEVES J., 1996, Representação de Informação Incompleta. Unidade de Ensino. Departamento de Informática. Universidade do Minho. Braga, Portugal. Novembro 1996. (Tech Report in Portuguese)

[3] ATKINSON, R. and COURT, R., 1998, *The New Economy Index: Understanding America's Economic Transformation* (Washington DC: Progressive Policy Institute).

[4] BONGAERTS, L., VALCKENAERS, P., Van BRUSSEL, H. and WYNS, J., 1995, Schedule Execution for a Holonic Shop Floor Control System. *Proceedings of the Advanced Summer Institute* (ASI'95). 24-28 June, 1995. Lisbon, Portugal.

[5] BUSSMANN, S., 1998, An Agent-Oriented Architecture for Holonic Manufacturing Control. IMSEurope'98, *Proceedings of the First International Workshop on Intelligent Manufacturing Systems*, pp.1-12.

[6] CVM, 1999, *Visionary Manufacturing Challenges for 2020*. (Washington: National Academic Press.)

[7] DAVIS, R., and SMITH, R., 1983, Negotiation as a metaphor for distributed problem solving, *Artificial Intelligence*, **20**(1), 63-109.

[8] GOU, L., and LUH, P., 1997, Holonic Manufacturing Scheduling: Architecture, Cooperation Mechanism, and Implementation. *Proceedings of IEEE/ASME International Conference on Advanced Intelligent Mechatronics*.

[9] HÖPF, M., 1994, Holonic Manufacturing Systems – The Basic Concept and a Report of IMS Test Case 5. In J. Knudesen *et al.* (Eds.) *Sharing CIM Solutions* (IOS Press).

[10]KÁDÁR, B., MONOSTORI, L., and SZELKE, S., 1998, An Object-oriented Framework for Developing Distributed Manufacturing Architectures, *Journal of Intelligent Manufacturing*, **9**(2), 73-179.

[11]KIDD, P., 1994, *Agile Manufacturing, Forging New Frontiers*, Addison-Wesley.

[12]KOESTLER, A.. 1967, *The Ghost in the Machine*, Hutchinson & Co, London.

[13]KUSIAK, A., 1990, *Intelligent Manufacturing Systems* (Prentice-Hall, Inc).

[14]NEVES, J., MACHADO, J., ANALIDE, C., NOVAIS, P. and ABELHA, A., 1997, Extended Logic Programming Applied to the Specification of Multi-Agent Systems and Their Computing Environment. *Proceedings of the 1997 IEEE International Conference on Intelligent Processing Systems*. Beijing, China.

[15]NGM, 1997, Next Generation Manufacturing – A Framework for Action. Next Generation Project Report, Agility Forum.

[16]PARUNAK, H., 1996, Applications of Distributed Artificial Intelligence in Industry. In O'Hare and Jennings (Eds.) *Foundations of Distributed Artificial Intelligence* (Wiley Inter-Science).

[17]RAMOS, C., ALMEIDA, A., and VALE, Z., 1995, Scheduling Manufacturing Tasks considering Due Dates: a new method based on Behaviours and Agendas. *Proceedings of the International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems*, pp.745-751.

[18]SHEN, W., and NORRIE, D., 1999, Agent-Based Systems for Intelligent Manufacturing: A State-of-the-Art Survey, *Knowledge and Information Systems* **1**(2), pp. 129-156.

[19]SMITH, R., 1980, The Contract Net Protocol, *IEEE Transactions on Computers*, **C-29**(12).

[20]SOLBERG, J., and KASHYAP, R., 1993, ERC Research in Intelligent Manufacturing Systems, *Proceedings of the IEEE*, **81**(1), pp.25-41.

[21]SOUSA, P., 2000, Agentes Inteligentes em Sistemas Holónicos de Produção. Tese de Doutoramento, Universidade do Minho. Braga, Portugal. (PhD Thesis in Portuguese)

[22]SOUSA, P., and RAMOS, C., 1997, Proposal of a Scheduling Holon for Manufacturing. PAAM'97, *Proceedings of Second International Conference on The Practical Application of Agents and Multi-Agents Technologies*, pp.255-268.

[23]SOUSA, P., and RAMOS, C., 1998, A Dynamic Scheduling Holon for Manufacturing Orders, *Journal of Intelligent Manufacturing* **9**(2), pp.107-112.

[24]SOUSA, P., and RAMOS, C., 1999a, A Distributed Architecture and Negotiation Protocol for Scheduling in Manufacturing Systems, *Computers in Industry*, **38**(2), pp.103-113.

[25]SOUSA, P., RAMOS, C., and NEVES, J., 1999b, Contracting Tasks Between Autonomous Resources – An application to dynamic scheduling of manufacturing orders. PAAM'99, *Proceedings of The Fourth International Conference on The Practical Application of Agents and Multi-Agent Technologies*, pp.345-362.

[26]SOUSA, P., RAMOS, C., and NEVES, J., 2000a, Fabricare: An Integrated View of a Distributed Manufacturing System. IAS-6, *Proceedings of the 6th International Conference on Intelligent Autonomous Systems*, pp.423-428.

[27]SOUSA, P., RAMOS, C., and NEVES, J., 2000b, Manufacturing Entities with Incomplete Information, *Studies in Informatics and Control Journal*, **9**(2), pp.79-88.

[28]SOUSA, P., SILVA, N., HEIKKILA, T., KALLINGBAUM, M., and VALCKNEARS, P., 2000c, Aspects of Co-operation in Distributed Manufacturing Systems, *Studies in Informatics and Control Journal*, **9**(2), pp.89-110.

[29]THARUMARAJAH, A., WELLS, A., and NEMES, L., 1996, Comparison of the Bionic, Fractal and Holonic Manufacturing Concepts, *International Journal of Computer Integrated Manufacturing*, **9**(3), pp.217-226.

[30]TRAYLOR, B. and GELFOND, M., 1993, Representing Null Values in Logic Programming. *Proceedings of the International Logic Symposium* (ILPS'93). Vancouver, British Columbia, Canada.

[31]UEDA, K., 1994, *Biological Manufacturing Systems* (Tokio: Kogyochosakai Pub. Co.)