DISTRIBUTED PRODUCTION PLANNING AND CONTROL AGENT BASED SYSTEM

Rui M. Lima, Rui M. Sousa, Paulo J. Martins

Department of Production and Systems, School of Engineering of University of Minho,
Campus de Azurém, Guimarães, Portugal

Abstract
A model of an Agent based Production Planning and Control (PPC) system able to be dynamically adaptable to local and distributed utilization of production resources and materials is presented. The PPC system is based on the selection of resources to deal with one order of different quantities of one product each time. In this way it is build one scheduling solution for that particular order. The production resources are selected and scheduled using a multiagent system supported by an implementation of the Smith Contract Net, using Java Spaces technology. The multiagent system is based on three main agents: Client, Resource and Manager. These agents negotiate the final product, and the correspondent components, requested by the client. An order for each product (component) triggers a process of dynamic design of a production system to fulfill that particular order. This system exists till the end of the order.

Keywords:
Distributed Production Planning and Control, Multi-Agent System

1 INTRODUCTION

New paradigms are necessary for enterprise representation and operation, and consequently also for production systems, in order to deal with the progressive reduction on time to market of new customized products and the ever-growing need for new enterprise competition approaches. These requirements make a further stress to the production Planning and Control System, which must be dynamically adaptable to both local and distributed utilization of production resources and materials.

Slightly different approaches to networks of enterprises like Virtual or Extended Enterprises are being referred in the literature. Virtual Enterprises are ephemeral associations of enterprises to give answer to a transitory opportunity, usually highly technological dependent, and Extended Enterprises results from a more steady association of enterprises throughout the manufacturing chain, usually centred on a dominant one.

Some paradigms for building the relations between production resources have been proposed, like Holonic Manufacturing and Fractal Factory.

Holonic Manufacturing Systems are based on production units inside other units making sub-systems that will build the final production system [1]. Each one of these units are Holons, which have simultaneously the all and the one characteristics. They have characteristics of the one because they have part of the functionality of the system, and on the other perspective they can be viewed as a system (the all) because they have some kind of autonomy (like the all system) and contain other Holons inside them.

A Holon can be defined by his functions or tasks, working on a holarchy restricted by some imposed rules. Changes in the environment cause reaction of Holons that communicate their change to the holarchy. Then, the holarchy can change their strategy (changing rules) in order to answer to the changing environment. A Holon is a holarchy of Holons that are controlled, in part, by the imposed rules. In other way, the Holon is subordinated to the rules imposed by the holarchy to which it belongs.

The Fractal Factory concept was introduced by Warnecke [2] based on fractal geometry. This factory is composed of similar units (fractal) that provide production services. The definition of the fractal by [2] is: *a fractal is an independently acting corporate entity whose goals and performance can be precisely described*. The fundamental characteristics of the fractal are: self-similar, self-organization, goal orientation and dynamics.

Bionic Manufacturing System is a production system concept conceived by Okino, and inspired on the biological living being systems [1]. Living beings are composed by entities that organise them selves and function in an autonomous way inside a hierarchical structure. Cells group them selves into tissues that create organs; these build organisms, living beings, and species. The cell is the smallest living organism entity capable of independent life, which can, isolated or interacting with other cells, execute all life functions. Production units of productions systems, equivalent to the biological system cell, get and send production objects (materials, products and information) to surround environment. Communication will be established by information and material flow, regulated by coordinating units, which will allow the integration of activities of the autonomous production units. These coordinating units will also have the function of linking cells in different levels of the hierarchy.

2 MULTI-AGENT PRODUCTION SYSTEMS

2.1 Software Agents

Software agents are components of software that represent user intentions. Table 1 presents definitions of three different authors.

Table 1 : Software Agent Definitions

Author	Definition
Nwana [3]	"*When we really have to, we define an agent as referring to a component of software and/or hardware which is capable of acting exactingly in order to accomplish tasks on behalf of its user.*"
Jennings and Wooldridge [4]	"*First, an agent is a computer system situated in some environment, and that is capable of autonomous action in this environment in order to meet its design objectives.*"
Parunak, Sauter, Fleischer, and Ward [5]	*"Agents add two things to (passive) objects: a local thread of control, and local initiative (usually expressed as local goals). Together, these enable the agent to monitor and respond to its environment autonomously (that is, without being externally invoked)."*

These definitions have some common characteristics like agent autonomy and project-defined objectives. In this context, autonomy is the software component ability to keep on executing their processes independently of interacting software or users. If the agent is not subject to direct interferences it can refuse task execution requests. Technically, agents respond to task requests in the opposite of objects that react by task invocation.

The agent autonomy must be integrated with some kind of environment monitoring ability in such a way that the agent actuation is compliant to the project objectives. These objectives are directly connected with the user represented by the agent. The action and reaction executed by agents depends on objectives introduced during the project. Depending on requests or environment changes, the agent must evaluate its internal state and objectives to be able to deliver an answer.

2.2 Multi-agent Systems

Multiagent systems are characterized by communities of agents whose interaction allows the achievement of system objectives. Nwana and Ndumu [6], argues that multi agent systems are created with the intention to establish connections between agents developed separately, allowing the overall capacity to go beyond the sum of the individual agents capacities.

In multiagent systems a common notion of the involved concepts must exist, in order to reach their objectives. These concepts can be explicitly defined in the ontology of the system, or can be implicitly defined in the knowledge database of each agent. Despite having a common knowledge of concepts of the system, agents can also have knowledge on the behaviour and reaction of the system due to other agents and environment changes. It can be said that software agents can have models of other agents and systems in which they are integrated, based on their project functionalities and behaviour.

Communication between agents depends on system architecture and can be done by message exchange. This exchange of messages can rely on known standard languages or on a specifically defined message protocol. System (environment) monitoring depends on its objectives, architecture and agents, and can be done by interception of requests that circulate in the system, by sensors reading, or by direct inquiries to the agent user or to other agents.

The connection structure of agents can be based [7, 8] on the following three architectures:

- Hierarchy of agents
- Federation of agents
- Autonomous Agents

The hierarchy is characterized by control relations from some agents over others. In that case, the autonomy of the agents is restricted by some degree of control imposed by dominant agents.

In a federation, individual agents or groups of agents communicate through mediator agents who supply communication services. These agents of communication can, basically, be of three types: facilitator; broker; match maker.

The facilitator is connected to a set of agents and communicates with other facilitators to supply communication services that allow system operation (examples in [9] and [10]).

The broker is an agent who actuates in one specified market, between supplier agents and client agents, supplying communication services that makes transparent the connection between them (examples in [5], [11], [12] and [13]).

The match maker supplies services similar to those supplied by the broker, being able, after that, to abandon the system, because the agents start to communicate directly between themselves (examples in [5] and [9]).

The architecture based on autonomous agents is an architecture where the agents can communicate between themselves without appealing to mediator agents. This type of architecture leads to systems where agents know from each other, or to systems where exists a common platform to transmit information available to all agents (examples in [7], [14], [15] and [16]).

3 DISTRIBUTED PRODUCTION SYSTEM DEFINITION

In general way, production systems are made of different processing elements that work in heterogeneous environments, simultaneously in different processes of different products, communicating in several ways. The term distributed, associated to the production system, emerges from the identification of new organising and management needs. These organising and management concepts must solve problems associated with the growing need of adaptability to change. This lead us to the following definition [17]:

A Distributed Production System is a production system composed by a network of autonomous processing elements, with the capability of rapid dynamic reconfiguration.

This definition excludes traditional production systems that do not allow instant dynamic reconfiguration. Moreover, new ways of management are necessary to allow this reconfiguration, integrating, in the same model, all parts of the system, including processing elements, inputs, outputs and communication systems.

4 AGENT PRODUCTION SYSTEM MODEL

In this model, software agents represent every production system element. These elements will be able to communicate with all other elements in an autonomous and interactive way.

4.1 System Requirements

The project of the proposed Agent Production System Model (MSDP) is based on the following requirements:

1. Product information input.
   i. Product specification.
   ii. Product structure definition.
   iii. Production processes specification.
2. Client information input.
   i. Identification.
   ii. Product orders generation.
3. Product orders input.
   i. Product Definition.
   ii. Quantity definition.
4. Order management agents should be able to:
   i. Select production resources.
   ii. Handle resource failure.
   iii. Do order monitoring.
5. Resource agents should be able to:
   i. Represent a production resource.
   ii. Register types of production processes that can be executed by the resource.
   iii. Publish lead time and costs for order requests.
   iv. Answer to agenda request.
   v. Do task scheduling.
   vi. Register confirmed orders.
   vii. Do monitoring tasks.
   viii. Process orders.
   ix. Act in conformity with defined objectives.

6. Evaluation criteria
   i. Order request answer time.
   ii. Lead time.
   iii. Negotiated due date fulfilment.
   iv. Cost.
7. Some of these requirements demand:
   i. Agent communication.
   ii. Actuating on environment.
   iii. Reacting to environment changes.

4.2 System Model

The project of a Distributed Production System model presented in this work, that fulfils the system definition and, partially, the requirements, is based on a multiagent system. A multiagent system is adequate for modelling and implementation purposes due to the following reasons:

- Distribution – Multiagent systems are adequate for the implementation of distributed and complex systems with resource allocation tasks.
- Autonomy – This is simultaneously an attribute of the distributed production system and of the agent definition, so agents can be used to represent autonomous production processing elements.
- Reconfigurability – The reconfigurability of distributed production systems can be implemented by proper coordination mechanisms of the multiagent system.

In this model, all elements of the Distributed Production System are represented by agents, which communicate with, and in name of, each element. The production resources, i.e. production system processors, delegate their representation on agents. The clients, direct or indirect users of resources, are also represented by agents. An order management agent is responsible for coordinating the resource allocation task based on the Smith Contract Net protocol.

According to Rich and Knight [18], in this type of coordination an agent decomposes the problem and negotiates the attribution of subtasks with other agents. In this contract net, agents may have two roles: Manager, who decomposes the problem, looks for contractors to execute parts of its problem, and supervises the execution; Contractor, who executes subtasks or starts looking for contractors to execute part of the work, becoming thus a manager.

In this model it is assumed that resource agents know the resource processing ability and capacity for executing tasks of known products. So, based on the knowledge about the transformations that the represented resource can execute, the agent can try to obtain orders for that resource.

4.2.1. Implementation

The system is implemented with agents distributed by different places, communicating through a shared repository. This form of communication is similar to a black board. All system agents access to this repository, monitoring thus the system activity.

JavaSpaces Technology (http://java.sun.com/javaspaces) included in Jini Network Technology was used for implementation purpose. This technology is described by Halter [19] and Bishop and Warren [20] as a system that delivers a set of services to manage distributed software objects. The objects can be placed on a “space”, named JavaSpace, where they can stay in a persistent way. The JavaSpace works like a Jini service, which allows the space clients to store and share software objects.

The software agents run on own threads, possibly in different machines, and send messages to other agents through a JavaSpace (Figure 1). These messages are in conformity with a message protocol, which is defined based on a common information model of the Distributed Production System. In some cases, messages don’t have to be sent to a particular client, and can be addressed to a given type of agents. For example a request for bids on some task for some order can be placed in the JavaSpace and several resources can answer to that bid. Figure 1 represents two resource agents, two client agents, an order management (OrderMgm) agent, three different messages flowing between several agents and the JavaSpace.

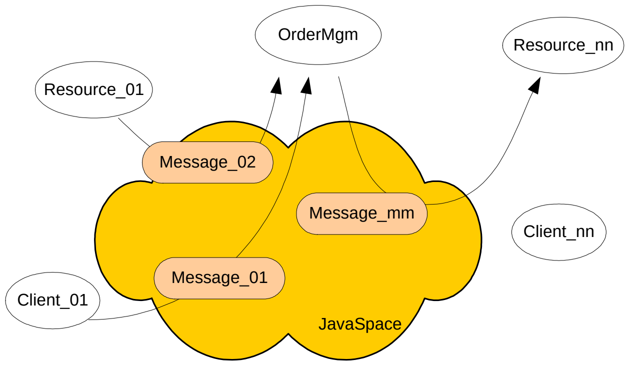

Figure 1: General Structure of the Agent Model.

4.3 Agents Specification

Agents Client represents each client and can, after registration in the system, send an order request for a particular product, through messages. This message is read by the order management (OrderMgm) agent, which will divide this in suborders for each element of the product structure. Each of these suborders will be recursively published in the JavaSpace. All interested Resource agents can make a bid for each of those suborders. Figure 2 illustrates the process of suborder message publication and the bid messages answers from interested resources.

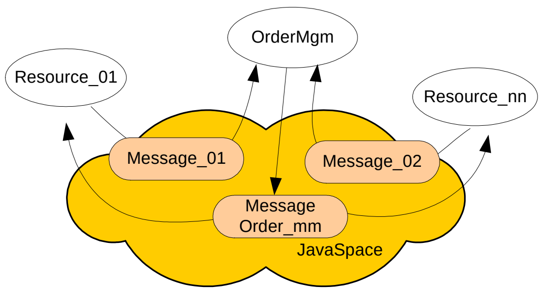

Figure 2: Elementary order task negotiation.

The Agent OrderMgm negotiates (requests) the final product requested by the client and respective components with Agents Resource. Each component has a detailed structure, associated to the production type of processes in all structure levels. The type of process is the transformation needed to get the component. The Agent OrderMgm is responsible for the selection of the resources needed for the production (candidate selection followed by particular resource(s) selection). An order for each product (component) triggers a process of dynamic design of a production system to fulfil that particular order. This system exists till the end of the order and is related with a particular set of resources selected. These resources are allocated to execute all tasks needed to make the production of all elements of the product structure. Figure 3 illustrates a configuration for a product P_1, which structure refers two components and one of them also has two components.

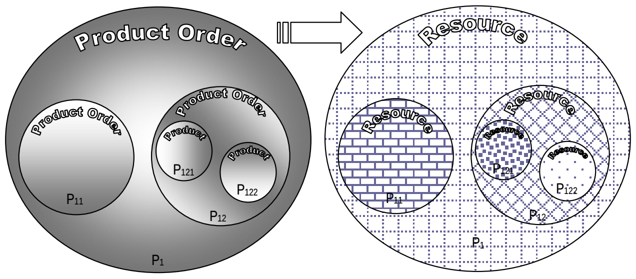

Figure 3: Production System example.

Each Agent Client is able to store information about the required products; the production means to execute them, the required processes and the orders already done. Each order a Client Agent makes is related to a final product that can be produced in different resources related to different types of processes.

4.3.1. Agent Client

Each client has an Agent Client that represents him and makes possible the orders input into the system. The introduction of an order, sending a message to the JavaSpace, will be the initiating event for the formation of each production system. Thus being, the Agent Client creates orders of a product in some required quantity.

Each Agent Client must be capable to identify the client it represents, to store information about products, orders, candidate and selected resources.

The ordered product results from the execution of some transformations that can be executed by some resources, if these have the required abilities. The Agent OrderMgm is responsible for the selection of offers, that is, for the attribution of some amount of work for candidate resources. This agent is also responsible for monitoring the execution of this order. An Agent Client creates an order for an intended product and sends a message for an Agent OrderMgm for the management of this order.

An order success should be reflected on the information management carried by the client. The client should, at least, store information for future reference about lead time, quality, cost and resources involved.

An order failure generates an internal conflict on the client, which the Agent should resolve, abdicating on the order or generating a new order. The first one of these solutions has implications on the user of this agent, who should be responsible for this decision, by inquiry or delegation. The decision to generate a new order leads again to the activity execution of launching an order.

There are more business activities related to clients that are not object of analysis in this study, because the objectives defined for this work were mainly related with production planning and control activities for distributed production systems.

4.3.2. Agent OrderMgm

The Agent OrderMgm will negotiate / request the product intended by the customer. The Agent OrderMgm will search for resources able to execute the necessary processes of production. This search process could be done by Agents with the ability to search information on the net, that is, to search information on operations and/or processes published by different resources. In a federate architecture, broker agents would have the responsibility to find the appropriate resources. In this work it is used a shared repository of information, where interest in the attainment of one determined product is published, and in which can be collected information on production means that can execute the required order.

Resource selection depends on the offers made by all resources for a particular order of a product. The interested resources have to publish their offers for execution of the work, indicating the execution time.

An activity of resource selection is composed, in general, by several sub-activities, that can be resumed by: resource selection, monitoring the order execution and communication with the customer. In this model, the production resources selection is made by the Agent OrderMgm, executing the actions presented in the UML [21] activity diagram represented in Figure 4. This activity is initiated by publishing the order requirements, followed by offers gathering. If exist offers from resources, then the control will be transferred to the “resource selection” action, else it will have to communicate that fact to the client.

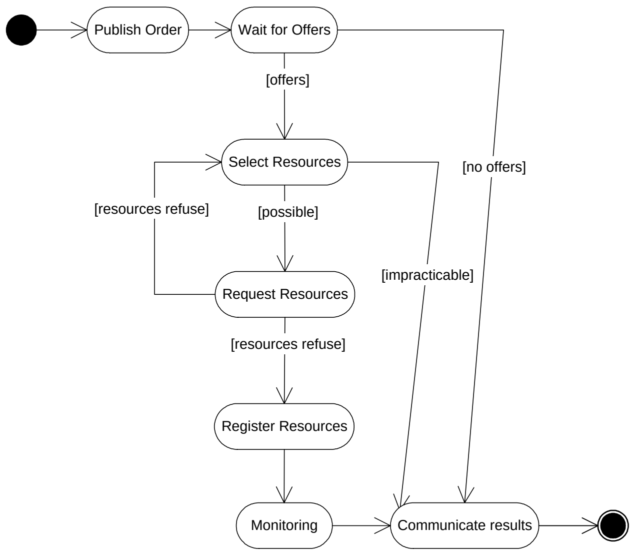

Figure 4: “Resource Selection” Activity Diagram.

If the resources selection (“Select Resources” action state) is successful, then the Agent OrderMgm executes the activity of effective solicitation of resources (“Request Resources” action state). The impossibility to make the selection of resources, due to the incapacity of the resources or to the application of the selection strategy, will be communicated to the order manager agent. This Agent OrderMgm will inform the Agent Client that should act in compliance with its objectives.

4.3.3. Agent Resource

The Agents Resource represents the available production resources, with the objective of obtaining work to be executed by the resources that they represent. When the Agent OrderMgm places information about an order in the shared JavaSpace, the Agent Resource can answer placing offers for this order in the space. Each of these offers depends on the request analysis and a decision is made, based on the agenda, capacity and abilities of the resource that is represented by the agent. This offer should have all information necessary for proper evaluation. The necessary information depends on the strategy used in the system for selection of resources. One of the functions of the agent is to keep its agenda updated, considering the selected orders and synchronized with the “real” resource.

The Agent Resource stores information about the resource it represents, the production processes that can execute, the orders for which it made offers and on those where were selected.

Figure 5 represents the activity diagram illustrating the activity of creation of offers for an order by the Agent Resource. This activity is initiated with the reception of a notice about publication of an order. This is followed by the execution of two parallel actions: one to update the resource agenda, in accordance with its internal information and the information of the resource it represents; other to search the order information in the shared JavaSpace. If any one of these actions fails, the activity finishes without creation or publication of offers.

The success of both of these actions allows to analyze the possibility of order fulfilment by the resource, that is, to analyze its aptitude, the availability of material and capacity and the fulfilment of the defined objectives. If the analysis is negative, then it refuses the order and finishes the activity without publishing any offer. The internal acceptance of the order leads to the creation and publication of an offer. The offer must include the lead time and associated cost.

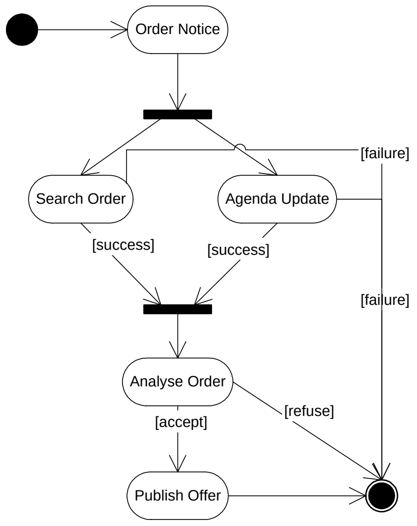

Figure 5 : Agent Resource “Offer” Activity Diagram.

If offers published by the Agent Resource are selected by an Agent OrderMgm, then there will be a request of a particular amount of work to be processed by the resource. In this case, the Agent Resource “work” activity (Figure 6) is similar to the “offer” activity. With this activity the Agent Resource has the objective of verifying the possibility of acceptance of the work request made by the Agent OrderMgm.

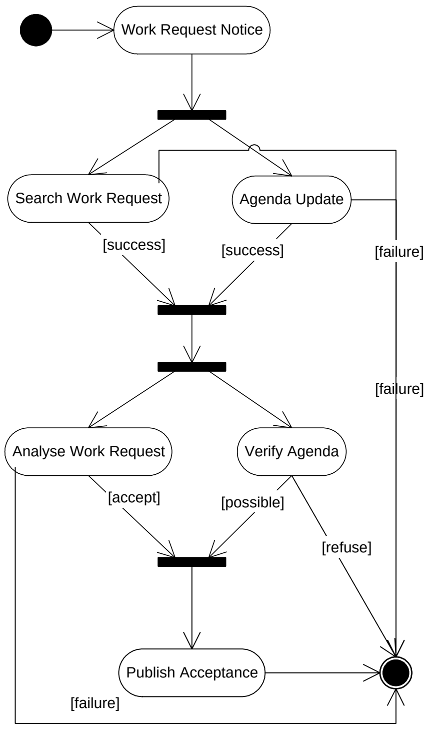

Figure 6: “Work Acceptance” Activity Diagram.

This “Work Acceptance” activity initiates with the reception of a work request, followed by the execution, in parallel, of two actions: one action to search the work request in the JavaSpace; other action to update the resource agenda. Having information about the work request and the agenda updated makes possible to take a decision about the acceptance of the request. This decision is based on two parallel actions: one to analyse the work request in relation with a previously made offer; other to verify the updated agenda in relation with a previously made offer. The success of these two activities allows the agent to publish the acceptance. This acceptance implies the commitment of the resource with the execution of the work.

4.3.4. *Coordination of Agents*

The definition of the Distributed Production System, the Agent Production System requirements and model place several orienting boundaries for the coordination of the agents.

The production system results from the selection of resources that are able to execute product transformations for the order. Making the selection of resources for each product transformation and grouping the *partial* solutions for each part of the product structure, makes possible to complete the order. This selection results, like referred on section 4.2, from a recursive application of a Contract Net negotiation protocol supported by message exchange through a JavaSpace. The implementation of this coordination mechanism is based on the exchange of object messages represented in Figure 7. This simple object allows the implementation of all communication protocol. Attributes “origin” and “destination” can identify specific agents. Attributes “type” and “param” are related with the message content. The first can be, for instance, an order or an offer and the “param” attribute is a vector data type containing information about the offer or order. In this object, only the first and the third of these attributes are mandatory.

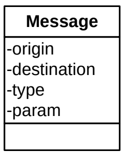

Figure 7 : Message Object

In the resulting production system, the order can be executed by one Agent Resource or several agents. This depends on the existence of the required ability and capacity, and the application of the selection strategy. In the developed system the selection strategy depends only on the lead time offer of each resource for each order.

A model for processing orders can be extremely detailed [22], with specification of functions, objects flow, control flow, materials flow, use and responsibility of functions, objectives, and organizational elements. In this model it is specified the order creation and publication activities and the resources selection, being encapsulated, in the Agent Resource, the order processing.

5 CONCLUSION

A definition and a model for Distributed Production Systems are presented in this work. This model is based on three main characteristics: Distribution, Autonomy and Reconfigurability. These are fundamental characteristics for new paradigms of production systems that propose to deal with highly dynamic environments.

The Distributed Production System Model is conceptually modelled and implemented as a multi agent system based on JavaSpaces technology. This implementation made possible the verification of the model validity in respect with production planning and control concepts. This system is able to make a production resource selection to fulfil a particular order considering the resources agenda. Moreover, it does this with distributed resources, which can accept or refuse orders depending on their own strategy. So, in this way, the production resources autonomy is respected and it is possible to deal with their distribution. Reconfigurability exists for each order, that is, there is one configuration of resources for each order. If one resource fails then it is also possible to build a new fraction of the system to fulfil the required part of the product structure.

There are three basic agents on the system: Agent Client, Agent OrderMgm and Agent Resource. These three agents have a common knowledge about the system and the environment, allowing communicating with each other. The presented UML specification of these agents is based on the description of the main activities they execute and that support their interaction.

In this work it is also presented the description of the communication protocol for sending message objects to a shared space and the coordination mechanism based on the application of the Smith Contract Net protocol.

Future work is planned in order to extend this model in functionality and also in different directions. The model functionality should be incremented in order to apply and compare different criteria for selection of resources, like cost or resource utilization. Moreover, it should allow implementing different architectures: a hierarchical architecture with optimized resource selection methods; creation of orders for the parts by the resource agents, which could, in some cases, ask for different types of operation that could deliver the same result; implementation of the model in industrial environments.

6 REFERENCES

[1] Tharumarajah, A., Wells, A. J., and Nemes, L., "Comparison of the Bionic, Fractal and Holonic Manufacturing System Concepts," *International Journal of Computer Integrated Manufacturing*, vol. 9, 1996, pp. 217-226.

[2] Warnecke, H. J., *The Fractal Company*: Springer-Verlag, 1993.

[3] Nwana, H. S., "Software Agents: A Overview," *The Knowledge Engineering Review*, vol. 11, 1996, pp. 205-244.

[4] Jennings, N. R. and Wooldridge, M., "Applications of Intelligent Agents," in *Agent Technology Foundations, Applications and Markets*, N. R. Jennings and M. Wooldridge, Eds.: Springer Verlag, 1998.

[5] Parunak, H. V. D., Sauter, J., Fleischer, M., and Ward, A., "The RAPPID Project: Symbiosis between Industrial Requirements and MAS Research," *Autonomous Agents and Multi-Agent Systems*, vol. 2, 1999, pp. 111-140.

[6] Nwana, H. S. and Ndumu, D. T., "A Perspective on Software Agents Research," *The Knowledge Engineering Review*, vol. 14, 1999, pp. 1-18.

[7] Shen, W., Norrie, D. H., and Barthès, J. P., *Multi-agent Systems for Concurrent Intelligent Design and Manufacturing*: Taylor & Francis, 2001.

[8] Shen, W., "Agent-Based Cooperative Manufacturing Scheduling: an Overview," *COVE News*, 2001.

[9] Shen, W., Maturana, F., and Norrie, D. H., "MetaMorph II: an agent-based architecture for distributed intelligent design and manufacturing," *Journal of Intelligent Manufacturing*, vol. 11, 2000, pp. 237-251.

[10] Sun, J., Zhang, Y. F., and Nee, A. Y. C., "A distributed multi-agent environment for product design and manufacturing planning," *International Journal of Production Research*, vol. 39, 2001, pp. 625-645.

[11] Baker, A. D., Parunak, H. V. D., and Erol, K., "Agents and the Internet: Infrastructure for Mass Customization," *Ieee Internet Computing*, 1999, pp. 62-69.

[12] Kim, Y., Choi, Y., and Yoo, S. B., "Brokering and 3D collaborative viewing of mechanical part models on the Web," *International Journal of Computer Integrated Manufacturing*, vol. 14, 2001, pp. 28-40.

[13] Carvalho, J. D. A., Putnik, G. D., and Cunha, M., "Infrastructures for Virtual Enterprises," *Cadernos do Departamento de Produção e Sistemas, Escola de Engenharia, Universidade do Minho (disponível em http://www.dps.uminho.pt/cad-dps)*, vol. DPS-03, 2003.

[14] Wiendahl, H. P. and Ahrens, V., "Agent-based control of self-organised production systems," *Annals CIRP*, vol. 46, 1997, pp. 365-368.

[15] Parunak, H. V. D., Baker, A. D., and Clark, S. J., "The AARIA Agent Architecture: An example of requirements-driven agent-based system design," presented at Proceedings of the First Intenational Conference on Autonomous Agents, Maryna del Rey, CA, 1997, pp. 482-483.

[16] Leitão, P., Restivo, F., and Putnik, G. D., "A Multi-Agent Based Cell Controller," presented at Proceedings of the 8th IEEE International Conference on Emerging Technologies and Factory Automation - ETFA 2001, Antibes - Juan les Pins, France, 2001, pp. 463-470.

[17] Lima, R. M., Silva, S. C., and Martins, P. M., "Sistemas Distribuídos de Produção," presented at 1º Congresso Luso-Moçambicano de Engenharia, Maputo, Moçambique, 1999, pp. B13-B20 (in portuguese).

[18] Rich, E. and Knight, K., *Artificial Intelligence*, 2nd ed: McGraw-Hill, 1991.

[19] Halter, S. L., *JavaSpaces: Example by Example*: Prentice Hall, 2002.

[20] Bishop, P. and Warren, N., *JavaSpaces in Practice*: Addison-Wesley, 2003.

[21] OMG, "Introduction to OMG's Unified Modeling Language™ (UML) http://www.omg.org/gettingstarted/what_is_uml.htm," OMG, 2005.

[22] Scheer, A.-W., *ARIS - Business Process Frameworks*, 3rd ed: Springer-Verlag, 1999.