4 HOLONIC MANUFACTURING CONTROL: A PRACTICAL IMPLEMENTATION

Paulo Leitão[1], Francisco Casais[1], Francisco Restivo[2]
[1]*Polytechnic Institute of Bragança, Quinta Sta Apolónia, Apartado 134, 5301-857 Bragança, PORTUGAL, {pleitao,fcasais}@ipb.pt*
[2]*Faculty of Engineering of University of Porto, Rua Dr. Roberto Frias, 4200-465 Porto, PORTUGAL, fjr@fe.up.pt*

The ADACOR holonic architecture for manufacturing control addresses the agile reaction to unexpected disturbances at the shop floor level, by introducing supervisor entities in decentralised systems characterised by the self-organisation capabilities associated to each ADACOR holon. The result is an adaptive control architecture that balances dynamically between a more centralised structure and a more decentralised one, allowing the combination of global production optimisation with agile reaction to unexpected disturbances. The validation of the proposed architecture is required to verify the correctness and the applicability of its concepts. This paper describes the implementation of ADACOR concepts using multi-agent systems, especially through the use of the JADE agent development platform.

1. INTRODUCTION

Companies, to remain competitive, need to answer more closely to the customer demands, by improving their flexibility and agility while maintaining their productivity and quality. The traditional manufacturing control systems respond weakly to the emergent challenges faced by the manufacturing systems, given their poor capability to adapt with agility to unexpected internal disturbances and to external environment volatility. This weakness is mainly due to the rigidity of the current control architectures.

Several manufacturing control architectures using emergent paradigms and technologies, such as multi-agent and holonic manufacturing systems, have been proposed (see [1-5]). One of the proposed holonic architecture is the ADACOR (ADAptive holonic COntrol aRchitecture for distributed manufacturing systems) architecture [6], which addresses the agile reaction to disturbances at the shop floor level, increasing the agility and flexibility of the enterprise.

ADACOR architecture is build upon a set of autonomous and cooperative holons, each one being a representation of a manufacturing component that can be either a physical resource (numerical control machines, robots, pallets, etc.) or a logic entity (products, orders, etc.). The holon is a concept first introduced by Koestler [7] to represent the interactions in social organisations, later introduced in manufacturing by the HMS consortium (see http://hms.ifw.uni-hannover.de/).

A generic ADACOR holon comprises the Logical Control Device (LCD) and, if exists, the physical resource, capable to perform the manufacturing task. The LCD is responsible for regulating all activities related to the holon and comprises three main components: decision, communication and physical interface [8]. In ADACOR agents are used to implement the logical part of the holon, i.e. the LCD device.

ADACOR architecture defines four manufacturing holon classes: product, task, operational and supervisor. The product, task and operational holons are quite similar to the product, order and resource holons defined in PROSA reference architecture [2], while the supervisor holon presents characteristics noot found in the PROSA staff holon. The supervisor holon introduces coordination and global optimisation in decentralised control and is responsible for the formation and coordination of groups of holons.

The ADACOR adaptive production control approach is neither completely decentralised nor hierarchical, but balances between a more centralised approach to a more flat approach, passing through other intermediate forms of control [6], due to the self-organisation capability associated to each ADACOR holon, translated in the autonomy factor and in the propagation mechanisms [8]. For this purpose, ADACOR evolves in time between two alternative states: the stationary state, where the system control relies on supervisors and coordination levels to achieve global optimisation of the production process, and the transient state, triggered with the occurrence of disturbances and presenting a behaviour quite similar to the heterarchical architectures in terms of agility and adaptability.

The validation of these concepts requires their implementation and testing, to analyse their correctness and applicability. This paper describes the implementation of ADACOR concepts, at the Laboratory of Automation in the Polytechnic Institute of Bragança, Portugal, to verify their applicability and if the system works as specified, either in normal operation or in presence of disturbances.

Along the paper, the implementation of the behaviour of each ADACOR holon class, communication infra-structure, manufacturing ontology, decision-making mechanisms, graphical user interfaces, customisation of manufacturing holons and connection between the holonic control system and the physical manufacturing devices will be described.

2. AGENT DEVELOPMENT PLATFORM

The development of holonic manufacturing control systems based in the ADACOR architecture requires the previous implementation of their concepts in a prototype.

The ADACOR prototype uses agent technology to implement each holon. Multi-agent systems can be adequately developed using object-oriented languages, such as Java. However, the development of multi-agent systems requires the implementation of features usually not supported by programming languages, such as message

transport, encoding and parsing, yellow and white pages services, ontologies for common understanding and agent life-cycle management services, which increases the programming effort. The use of agent development platforms which implement these features makes the development of agent-based applications easier and reduces the programming effort.

A significant set of platforms environments for agent development is available for commercial and scientific purposes, providing a variety of services and agent models, which differences reflect the philosophy and the target problems envisaged by the platform developers. Surveys of some agent development platforms can be found in [9-10].

The choice of an agent development platform obeyed to a set of criteria: to be an open source platform with good documentation and available support, ease to use, low programming effort, use of standards, features to support the management of agent communities like white pages and/or yellow pages and facilities to implement rule oriented programming.

The chosen platform was JADE (Java Agent Development Framework), provided by CSELT and available on http://jade.cselt.it/, because it responds better to the mentioned requirements. In fact, JADE simplifies the development of multi-agent systems by providing a set of system services and agents in compliance with the FIPA (Foundation for Intelligent Physical Agents) specifications: naming, yellow-page, message transport and parsing services, and a library of FIPA interaction protocols [11]. JADE uses the concept of *behaviours* to model concurrent tasks in agent programming and all agent communication is performed through message passing, using FIPA-ACL as the agent communication language. JADE provides the FIPA SL (Semantic Language) content language and the agent management ontology, as well as the support for user-defined content languages, which can be implemented, registered, and automatically used by the agents.

The agent platform provides a Graphical User Interface (GUI) for the remote management, allowing to monitor and control the status of agents, for example to stop and re-start agents, and also a set of graphical tools to support the debugging phase, usually quite complex in distributed systems, such as the Dummy, Sniffer and Introspector agents.

JADE offers also an easy and full integration with other useful tools, such as JESS (Java Expert System Shell) and Protegé 2000 (for knowledge based system development and management), and provides other features such as an active mailing list to support technical problems.

3. GENERAL ADACOR IMPLEMENTATION

In this section, the main issues related to the implementation of the ADACOR architecture prototype using the JADE framework will be described.

3.1 Internal Architecture of an ADACOR Holon

An ADACOR holon is a simple Java class that extends the **Agent** class provided by the JADE framework, inheriting its basic functionalities, such as registration services, remote management and sending/receiving ACL messages [11]. These

basic functionalities were extended with features that represent the specific behaviour of the ADACOR holon, like the *HandleReceiveMessages*, *AllocateTask* and many other behaviours.

The start-up of an ADACOR holon comprises its initialisation (read the configuration files and load the behaviours) and its registration according to the initial organisational structure, defined by the configuration files, and is followed by the actual start-up of the holon's components, i.e. the communication, decision and physical interface components.

The behaviour of each ADACOR holon uses multi-threading programming, over the concept of JADE's behaviour, allowing to execute several actions in parallel. The behaviours launched at the start-up and those which can be invoked afterwards are provided in the form of Java classes.

The communication between holons is done over the Ethernet network, using TCP/IP protocol and is asynchronous, i.e. the holon that sends a message continues the execution of its tasks without the need to wait for the response. The messages specified in the ADACOR architecture are encoded using the FIPA-ACL agent communication language, the content of the messages being formatted according to the FIPA-SL0 language. The meaning of the message content is standardised according to the ADACOR ontology.

One of the holon's concurrent tasks (behaviour) waits continuously for the arrival of messages using the *block()* method to block the behaviour until a message arrives. The arrival of a message triggers a new behaviour to handle the message (the *HandleReceiveMessages* behaviour), thus implementing an asynchronous communication mechanism over the JADE platform.

Each supervisor holon has embodied a DF (Directory Facilitator) that provides yellow pages functionalities, allowing to locate holons within its group by their capabilities.

3.2 ADACOR Ontology

ADACOR defines its own manufacturing control ontology, expressed in an object-oriented frame-based manner, as recommended by FIPA Ontology Service Recommendations (see http://www.fipa.org/). This recommendation refers to the development of classes describing concepts and predicates, and their registration as a part of the application ontology, allowing a practical and fast way of creating an ontology with an immediate underlying implementation.

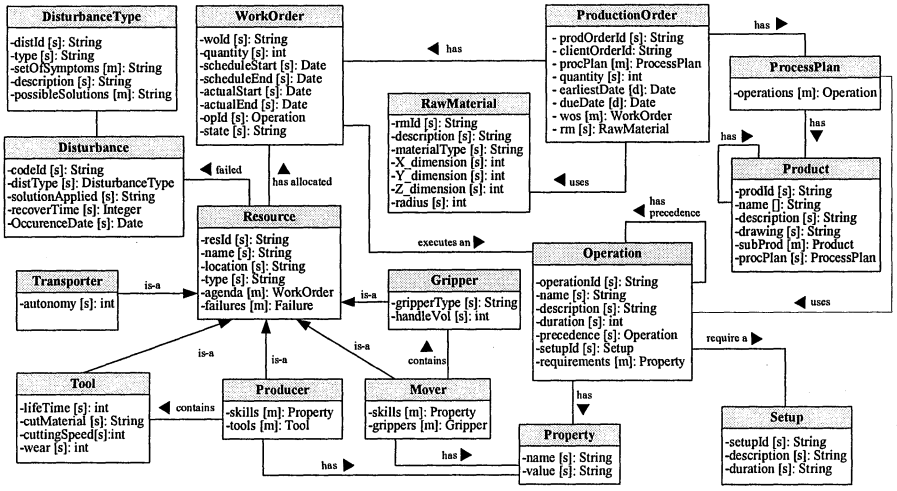

Figure 1 – ADACOR Ontology for Manufacturing Control

The manufacturing control ontology used in ADACOR is based in the definition of a taxonomy of manufacturing components, which contributes to the formalisation and understanding of the manufacturing control problem. These components are mapped in a set of objects, illustrated in the Figure 1, which defines the vocabulary used by distributed entities over the ADACOR platform, and indicates the concepts (classes), the predicates (relation between the classes), the terms (attributes of each class), and the meaning of each term (type of each attribute).

The ADACOR ontology was translated to Java classes according to the JADE guidelines that follow the FIPA specifications for the development of ontologies. The main class of the ADACOR ontology describes the concepts and predicates defined in the ontology, indicating its ontological role. Each ontological role is characterised by a name and a structure defined in terms of a number of slots that represent the attributes of the concept or predicate.

Instances of the ontological roles can be conveniently represented inside an agent as instances of application-specific Java classes each one representing a role. The methods defined in each individual class used to describe each concept or predicate, allow to handle the data related to the object.

3.3 Decision-Making Mechanisms

The ADACOR decision component, illustrated in Figure 2, uses declarative and procedural approaches to represent knowledge and to regulate the holons behaviour. The knowledge base of each ADACOR holon is dependent of its type, objectives, skills and behaviour.

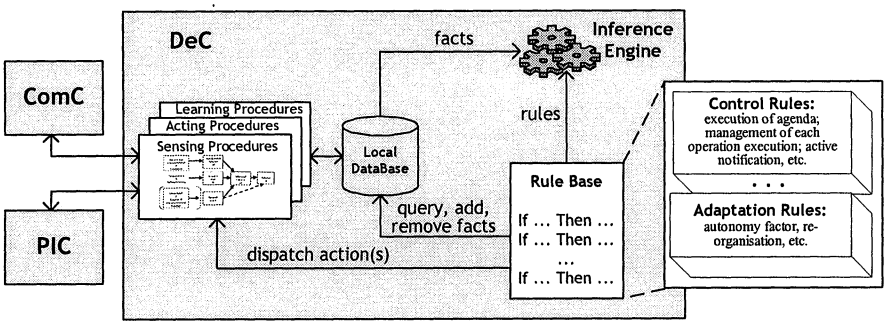

Figure 2 – Decision Component Architecture

The central element in the decision component is the rule-based system, which applies declarative knowledge, expressed in a set of rules. The advantage of this type of knowledge-based system is related to the simple and very comprehensive way to represent the reasoning capability of one holon. The simplicity and the associated high abstraction level of this approach compensates the typical weaknesses of these systems to handle incomplete, incorrect and uncertain information, or to implement complex systems, that require a large number of rules and can become very slow.

The rule-base system uses the JESS tool, which is a rule oriented programming infrastructure (Java based and JADE compatible) developed using the CLIPS (C Language Integrated Production System) language and uses the Rete algorithm as inference engine [12]. JESS handles data structures, functions and rules, requiring the use of a *clp* file to store the application knowledge base.

Each ADACOR holon class has its own *clp* file containing its knowledge base. The decision mechanisms that are common to all the ADACOR holons classes, such as the active notification, are placed in a special common *clp* file. The local database stores the short-term memory, i.e. the facts that represent the current state of the holon at a particular moment.

The set of rules defined in the knowledge base represents the behaviour of the ADACOR holon. As an example, the behaviour rule illustrated in Figure 3 is defined in the implementation of the task holon knowledge base.

```
(defrule Transport    "Will start the transport of the part"
   ?fact1 <- (Transport)
   ?fact2 <- (executing (jobInExecution ?wo))
   ?fact3 <- (WorkOrder (woID ?wo)(state ?state)(resName ?res)
             (location ?location)(precedence ?precedent))
=>
   (retract ?fact1 ?fact3)
   (assert (WorkOrder (woID ?wo)(state TRANSPORT)(resName ?res)
           (location ?location)(precedence ?precedent)))
   (ExecuteTransport ?wo ?*transport-path*)
)
```

Figure 3 – Invoking JADE Procedures from the JESS Environment

This rule has three conditions: the first condition is satisfied if the fact **Transport** is true; the second condition determines the name of the work order that is currently in execution; at last the third condition gets the information related to the operation in execution. When the three conditions are satisfied, the rule is selected and three actions are executed: first, the facts *fact1* and *fact3* are removed from the knowledge base; then the state of the work order is changed to **Transport**; finally, a behaviour in the JADE environment (linked to a simple Java class) that will be in charge to execute the transportation of the part is triggered.

Another type of connection supports the invocation of JESS commands from the JADE environment. This is done by introducing commands lines embedded in the Java program. In Figure 4, the extract of code illustrates the assertion of a new fact in the knowledge base, in this case a new work order.

```
...
decEngine.executeCommand("(assert (WorkOrder (woID " + value1 + ")
     (state " + value2 + ")(precedence " + value3 + ")))");
...
```

Figure 4 – Invoking JESS Commands from the JADE Environment

ADACOR holons also use procedural knowledge to represent knowledge. This type of knowledge is embodied in procedures, which are triggered as actions by some rules, each one being responsible for the execution of a particular set of actions. The scheduling algorithm is an example of this type of knowledge representation. Some other procedures are related to the acquisition of information by handling the arrival of messages from other holons or getting local information through the access to the physical manufacturing resource.

3.4 Graphical User Interfaces

In the ADACOR prototype, the operational and supervisor holons have graphical user interfaces to support the interaction with the user, illustrated in Figure 5.

The graphical user interface for the operational holons allows to visualise the local schedule, using a Gantt chart to show the work orders executed by the resource, to configure some operational holon parameters, such as the scheduler type or the activation of the autonomy factor, and to display statistical information related to the resource performance, such as the degree of utilisation, the number of work orders executed and the number of work orders delayed.

The graphical user interface for the supervisor holon allows to visualise the global schedule, using a Gantt chart to show the work orders executed by each lower-level resource, to display the resources under its coordination domain and their characteristics, and to configure some holon parameters, such as the schedule algorithm.

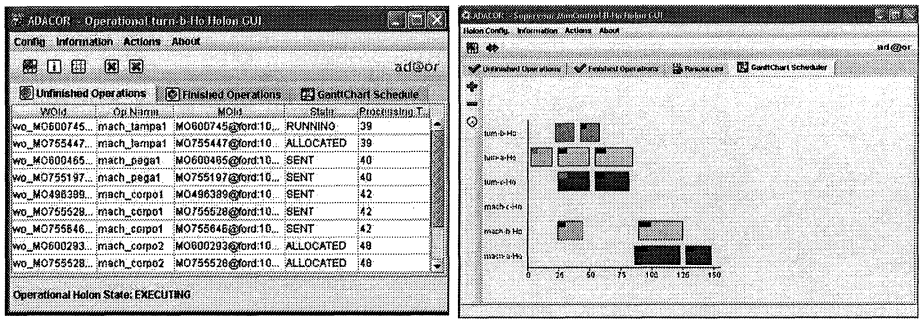

Figure 5 – Graphical User Interface of an Operational and a Supervisor Holon

Each ADACOR holon uses log files to store the relevant data during its life cycle, to support posterior analysis or to execute backup procedures in case of holon crash. In case of operational holons, this log file stores information about rejected, cancelled, failed and executed work orders. The task holon uses a log file to store the relevant information associated to the execution of the production order and to store the statistical report about the performance of the execution of the production order, such as the manufacturing lead time, tardiness, processing time and idle time.

4. APPLICATION SPECIFIC COMPONENTS

The holonification of the manufacturing components requires the configuration and customisation of the ADACOR holons and the development of wrappers that allow the connection between the control system and the physical devices.

4.1 Configuration Files of ADACOR Holons

The characteristics of each manufacturing holon are configured using a XML (eXtensible Markup Language)-based configuration file.

In case of the product holon it is necessary to introduce the product data model and the process plan. The description of the product structure is represented by a list of objects formatted according to a data structure, which includes the name of the sub-part, the number of parts necessary to produce the product and the estimated time to produce the part. The process plan defines the sequence of operations that must be executed to produce the product, containing a list of operations formatted according to an appropriated data structure, which mainly describes the name of the operation, a brief description about the operation, the estimated time to execute the operation, the reference to the part, a list of requirements associated to the operation, and the name of the operations that have precedence over this operation.

The characteristics of each resource are mapped in a XML-based configuration file that will be read by the operational holon. The data structure represents the resource model, which attributes reflect mainly the type of resource, the list of skills that the resource possesses and its location.

The organisational structure of the factory plant is defined in a XML-based configuration file that comprises the information related to the cell organisation and to the shop-floor layout. This organisational structure describes the possible manufacturing cells and associated coordinator entities, which will be converted into supervisor holons. With this organisational structure XML-based file, the operational holons can find their supervisor holons and their auxiliary resources, and the supervisor holons can find the list of holons that are in its coordination domain.

4.2 Physical Resource Interfaces

The implementation of operational holons that represent physical manufacturing resources requires the development of wrapper interfaces, supporting the integration of those resources. In the ADACOR architecture, the virtual resource concept was introduced to make transparent the intra-holon interaction [13].

The development of a virtual resource for each manufacturing device encompasses the implementation of the services at the server side, which will be invoked on the client side (PIC component from the operational holon). The client ignores the details of this implementation and each virtual resource can be re-used by other similar resources or holonic control applications.

Leitão et al. [13] describe the implementation of two different virtual resources to integrate two different automation resources, a PLC and an industrial robot. These two virtual resources implement the same services so that from the client side, whatever the resource is accessed, the invocation made is unique.

Here the virtual resource for an ABB IRB1400 load/unload industrial robot is briefly described, by the illustration of the implementation of the *read* service, as showed in the Figure 6.

```
public int read(String var,String type) {
  String[] vname=new String[1];
  vname[0]=var;
  short[] progNo=new short[1];
  progNo[0]=0;
  short varvalue=0;
  ...
  try {
     varvalue=h.s4ProgramNumVarRead(vname,progNo);
  }catch(IOException ioe) {System.out.println("Problem: " + ioe);}
  return ((new Short(varvalue)).intValue());
}
```

Figure 6 – *Read* Service for the ABB IRB 14000 Virtual Resource

The services provided by the virtual resource were developed using the RobComm ActiveX supplied by ABB [14], and accessed through TCP/IP. The major problem was the access to ActiveX from a Java program, since the ActiveX components are adequate to be manipulated by Windows-based programming environments. To overcome this problem, it was used the Jintegra tool (see http://j-integra.intrinsyc.com/) to convert the ActiveX component into a Java package [13].

The platform used to support the client-server interaction was CORBA. The analysis of the experimental implementation of the resource integration, by comparing the performances of CORBA, RMI and RMI-IIOP, is described in [13].

5. AUXILIARY TOOLS

During the implementation of the ADACOR prototype, a set of auxiliary tools was developed to support the configuration, operation and debugging of manufacturing control applications, providing functionalities to configure products and to supervise the factory plant in an integrated and global view.

As the global visualisation of all activities in the factory plant is difficult due to the use of multiple graphical user interfaces, the ADACOR Factory Plan Supervisor (AFPS) is used to monitor the production activities in the factory plant. Its graphical user interface is represented in Figure 7.

This tool allows to visualise the production process by enabling the visualisation of the manufacturing resources present in the factory plant, indicating their state and characteristics. In this tool, the visualisation of the transport resource has animation capabilities to help understanding of the material flow in the factory plant, indicating the direction of the movement and its actual load.

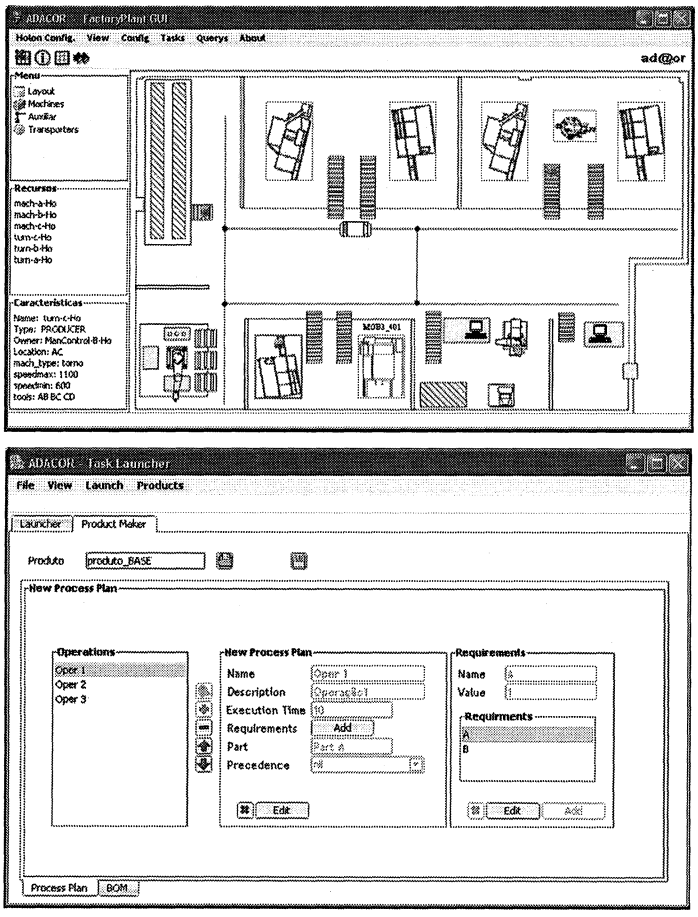

Figure 7 – Graphical User Interface of the Auxiliary ADACOR Agents

The ADACOR Product Manager (APM) agent, which graphical user interface is also represented in the Figure 7, allows defining new products in the system, by introducing the structure of the product and the process plan that defines the sequence of operations to execute the product. It also allows launching individual or pre-defined sequences of production orders to the factory plant, making easier the execution of experimental tests.

6. CONCLUSIONS

This paper described the implementation of the ADACOR holonic manufacturing control architecture concepts into a prototype.

Firstly, the implementation showed the capability of ADACOR architecture to represent and control a real environment. An application to a flexible manufacturing system was completed as a part of the doctoral thesis of an of the authors. The experience gained during the prototype implementation, debugging and testing, allowed proving essentially the applicability of the ADACOR concepts and the merits of the ADACOR collaborative/holonic approach.

The use of agent technology to implement the holonic manufacturing control prototype brings some important benefits: the software necessary to develop the application is simpler to write, to debug and to maintain, due to the smaller size of each distributed component. The use of Java language contributes for the platform independency, which is mandatory in manufacturing environment, due to its heterogeneous environment.

The use of JADE agent development tool brings several advantages in the development of holonic and multi-agent systems, such as the reduction of the development time and complexity. For this fact contributes the good documentation, efficient technical support and the set of functionalities provided by the platform that simplifies the development of multi-agent systems, such as the communication infra-structure, yellow and white pages and debugging tools.

7. REFERENCES

1. Parunak, H. Van Dyke, Baker A. and Clark S. The AARIA Agent Architecture: from Manufacturing Requirements to Agent-based System Design, Workshop on Agent-based Manufacturing, 1998.

2. Van Brussel, H., Wyns J., Valckenaers P., Bongaerts L. and Peeters P. Reference Architecture for Holonic Manufacturing Systems: PROSA, Computers In Industry, vol. 37, 1998, pp. 255-274

3. Brennan, R., Balasubramanian, S. and Norrie, D., A dynamic control architecture for metamorphic control of advanced manufacturing systems. Proceedings of the International Symposium on Intelligent Systems and Advanced Manufacturing, 1997, pp. 213-223.

4. --: Special Issue on Industrial Applications of Multi-Agent and Holonic Systems. Journal of Applied Systems Studies, 2(1),2001.

5. Deen S.M. (ed.): Agent-Based Manufacturing: Advances in the Holonic Approach. Springer Verlag Berlin Heidelberg, 2003.

6. Leitão P. and Restivo F.: Holonic Adaptive Production Control Systems. Proceedings of the 28th Annual Conference of the IEEE Industrial Electronics Society, Sevilla, Spain, 2002, pp. 2968-2973.

7. Koestler, A.: The Ghost in the Machine. Arkana Books, London, 1969.

8. Leitão P. and Restivo F.: Agent-based Holonic Production Control. Proceedings of the 3rd International Workshop on Industrial Applications of Holonic and Multi-Agent Systems (HoloMAS), Aix en Provence, France, 2-6 September, 2002, pp. 589-593.

9. Vrba P.: Java-Based Agent Platform Evaluation. In V. Marík, J. Müller and M. Pechoucek (eds), Multi-Agent Systems and Applications III, Volume 2691 of LNAI, Springer-Verlag, 2003, pp. 47-58.

10. Barata, J., Camarinha-Matos, L., Boissier, R., Leitão, P., Restivo, F. and Raddadi, M.: Integrated and Distributed Manufacturing, a Multi-agent Perspective. Proceedings of 3rd Workshop on European Scientific and Industrial Collaboration, Enschede, Netherlands, 27-29 June 2001, 145-156.

11. Bellifemine F., Caire G., Trucco T. and Rimassa G.: JADE Programmer's Guide. 2002.

12. Friedman-Hill E.J.: JESS, The Java Expert System Shell. Sandia National Laboratories, 1999.

13. Leitão P., Boissier R., Casais F. and Restivo F.: Integration of Automation Resources in Holonic Manufacturing Applications. In V. Marík, D. McFarlane e P. Valckenaers (eds), Holonic and Multi-Agent Systems for Manufacturing, Volume 2744 of LNAI, Springer-Verlag, 2003, pp. 35-46

14. --: RobComm User's Guide, version 3.0/3. ABB Flexible Automation, 1999.