# Integration of Automation Resources in Holonic Manufacturing Applications

Paulo Leitão[1], Raymond Boissier[2], Francisco Casais[1], and Francisco Restivo[3]

[1] Polytechnic Institute of Bragança, Quinta Santa Apolónia, Apartado 134, P-5301-857 Bragança, Portugal,
{pleitao, fcasais}@ipb.pt

[2] Université Paris-Nord, GRPI. 93206 Saint Denis Cedex 1, France,
boissier@iut-stdenis.univ-paris13.fr

[3] Faculty of Engineering, University of Porto, Rua Dr. Roberto Frias, P-4200-465 Porto, Portugal,
fjr@fe.up.pt

**Abstract.** Holonic and agent-based paradigms are very suitable in the development of distributed manufacturing control systems, taking advantage of their modularity, decentralization, and ability to support dynamic and complex system design features. However, the integration of manufacturing resources within the holonic manufacturing control applications remains a problem, because no efficient standard allows an easy, transparent and essentially independent integration. This paper proposes an integration process that allows the integration of automation resources independently from the holonic control application, using some concepts derived from the Manufacturing Message Specification (MMS) application protocol and implemented over a distributed object platform.

## 1 Introduction

Nowadays, in order to face stochastic and volatile environments, manufacturing systems must exhibit increasing agility. This implies the corresponding control application must also adapt to the occurrence of unexpected disturbances, very likely through dynamic and distributed structures.

The Holonic Manufacturing System (HMS) paradigm seems a promising approach to support these actual and emergent requirements. HMS translates to the manufacturing world systemic concepts developed by A. Koestler concerning living organisms and social organizations, mainly that complex systems are hierarchical systems formed by intermediate stable forms, being simultaneously a part and a whole [1]. In industry, the word holon, which illustrates this hybrid nature, represents the manufacturing components and activities, such as machines, products and parts. Their behavior is determined by their cooperation with other holons, as opposed to being determined by a centralized mechanism.

Due to the heterogeneous manufacturing environment, it is hard and cumbersome to develop holonic manufacturing applications that integrate manufacturing resources, because the holonic application is highly dependent of the resource interfaces, normally developed one-of-a-kind. The solution requires a standardized integration process that makes transparent the interface between the holonic manufacturing applications and the manufacturing resources. Some relevant approaches, such as the MMS protocol and the OPC (OLE for Process Control), do not cover integrally the manufacturing resource characteristics and complexity, and do not support the independency between the control and the integration domains.

Our approach to the problem is the re-use of the basic MMS concepts over a distributed object based platform, thus forming a set of objects that are invoked remotely by the holonic control application. These objects are always the same on the client side (i.e. the controlling application), but are customized in the server side (i.e. the controlled programmable manufacturing device), according with the resource details.

In this paper, initially a holonic manufacturing control architecture is briefly described, which demands for a transparent resource integration in order to support heterogeneity and interoperability. In section 3 an overview of available technologies to support the resource integration is presented, while section 4 describes the proposed resource integration mechanism based in the virtual resource concept and in a client/server distributed object platform. In the last part, the implementation of two virtual resources, one for a programmable logic controller (PLC) and other for an industrial robot, that validates the proposed approach, is described.

## 2 Holonic Manufacturing Control System

The ADACOR (Adaptive Holonic Control Architecture for Distributed Manufacturing Systems) architecture, proposes a new holonic approach for flexible manufacturing systems control, focused on distributed manufacturing shop floor control, and facing the dynamic and agile adaptation to disturbances. The architecture is based on a set of autonomous, intelligent and co-operative entities, designated by holons, to represent the automation factory components, aiming to support the distribution of skills and knowledge. These manufacturing components can be both physical resources (numerical control machines, robots, programmable controllers, etc) and logic entities (products, orders, etc), grouped in the following main types of holons: *product*, *task*, *operational* and *supervisor* [2]. The operational holon type represents the physical resources and comprise the physical manufacturing resource, capable of performing manufacturing operations and the Logical Control Device (LCD), which acts as an agent, and contains*: inter-holon interaction mechanisms* to support negotiation and coordination with other holons, *manufacturing control functions* that regulate the behavior of the holon aiming to pursuit its goals, and *intra-holon interaction mechanisms* to support the interaction between the physical manufacturing resources and the LCD.

The internal architecture for a generic LCD belonging to an ADACOR holon, Fig. 1, comprises a local knowledge base and three main components: *decision* (DeC), *communication* (ComC) and *physical interface* (PIC) components [3].

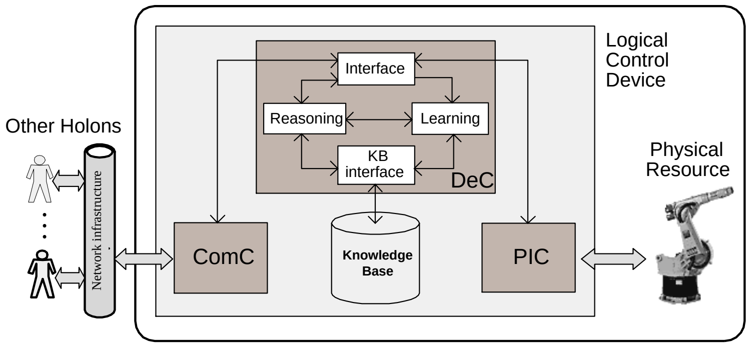

**Fig. 1.** Internal Architecture for an ADACOR Agent

The *physical interface component* provides the mechanisms to integrate the manufacturing resources. Since the manufacturing factory is a heterogeneous environment, with the distributed holons and the automation devices placed on a wide variety of interoperable control platforms, a crucial point when holonifying manufacturing components, such as robots and machine-tools, is the connection between the software part of the holon and the physical manufacturing resource. As the local resource controllers have mostly closed architectures it is necessary to develop wrappers to hide the details of each resource controller and supplies primitives that represent the functionality of the physical manufacturing resource [4]. Thus, the PIC component comprises the mechanisms for the interaction with the physical devices that makes transparent the access to manufacturing resources from the holonic control application, and independent the control application from the integration domain.

## 3 Manufacturing Resource Integration Technologies

As referred, the integration of manufacturing resources into holonic and agent-based applications assumes a crucial aspect, requiring mechanisms that make transparent and independent the control application from the details of local resource controllers. Moreover, FIPA (Foundation for Intelligent Physical Agents) [5], which aims to produce standards for the interoperation of heterogeneous software agents, does not present, at the moment, specifications to support the integration of physical resources, in spite of the effort to introduce new specifications that support manufacturing requirements, through the FIPA Product Design and Manufacturing working group.

MMS, which defines the application layer of the ancient MAP (Manufacturing Automation Protocol) protocol, provides a platform capable to interconnect various industrial devices supplied by different suppliers. MMS brought together many IT and manufacturing specialists to define a common framework for developing communication support between industrial computerized equipment, under the ISO 9506 international standard [6]. The basic concepts of the MMS protocol are a client-server mechanism and the VMD (Virtual Manufacturing Device) model. The VMD, associated with every real manufacturing device, is an abstract model of the server application, which maps the functionalities of the real device, and offers all services concerning itself and its related abstractions, mainly: *domains*, *variables*, *program invocations* and *events*. This set of objects and generic services can be applied to a large set of manufacturing devices, such as robots and numerical control machines [7]. However, the technology vendors do not closely follow the MMS standard, since certain functionalities have different implementations depending of the machine vendor and the underlying network. This missed adhesion by the vendors to the standardization associated to the high price of this technology retracted the expansion of the MMS technology in the market.

Some research teams introduced the idea to use the MMS concepts combined with a distributed object platform technology to integrate the manufacturing resources. The approaches presented in [8, 9] use the MMS concept over the CORBA (Common Object Request Broker Architecture) distributed object technology, with successful results. The real-time constraints in the industrial manufacturing world requires real-time response, being the CORBA technology and the classical TCP/IP not adequate. The ReTINA model has been used within the *Jonathan* distributed environment [10], in order to support applications subject to real-time functioning [11].

Another available technology is the OPC, which is based on Microsoft's OLE/COM technology. It allows software in the form of software components to interoperate regardless of where they are located [4]. The OPC servers, OLE/DCOM compliant, offer an automation interface, which allows to design PC-based clients that import real time automation data using standard Windows applications. The Windows proprietary scope remains an important limitation of this approach for the heterogeneous environments; however, some available tools, such as J-Integra [12] and Bridge2Java [13], allow to overcome this problem.

IEC 61499 standard [14, 15] is an approach for the easy and quickly integration of large re-configurable systems, defining a way to model the control and execution of algorithms in distributed control systems, being encapsulated, reusable software modules. Within this model, the ancient function block concept is re-introduced in order to make a clear distinction between the event and the triggered algorithms, making easier the verification of time properties [16]. A function block, which is the fundamental unit of software encapsulation and reuse in IEC 61499, encapsulates the control algorithm with physical interfaces, communications, human interfaces, monitoring and diagnostics, and information technology-based services.

At the moment some low level programmable controllers, such as PICs and PLCs already support the IEC 61499 standard, which makes adequate this approach for the resource integration by the direct communication between the entities. However, for the low level programmable controllers that do not support yet IEC61499, and essentially for the communication between high level programmable controllers such as robots, numerical control machines and PLCs, which is the main focus of the paper, IEC 61499 is not yet a solution for the resource integration.

From the preceding, it is clear there is a need for a low cost approach that could support transparent interfaces for physical manufacturing resources, and allows easy integration of these resources into holonic control applications. The use of light MMS concepts combined with distributed object paradigms seems a suitable approach to make transparent the resource integration from the agent-based or holonic control system.

## 4 Resource Integration Approach

Our ADACOR approach to transparent resource integration within holons is taking advantage of the OO-MMS mechanism for communication between any client and a Virtual Machine Device server [8]. The scheme is displayed in Fig.2.

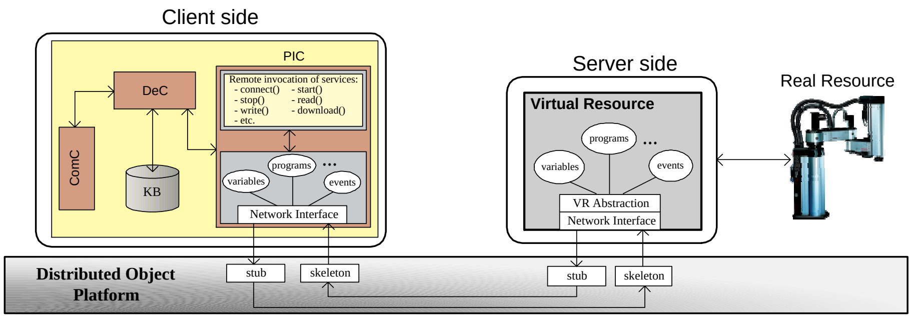

**Fig. 2.** Invocation of Remote Services using the Virtual Resource concept

Next, the main concepts of the schema, mainly the virtual resource and the client-server model, will be deeply analyzed.

### 4.1 Virtual Resource

The server part in the proposed mechanism is the virtual resource, inspired by the VMD concept from the MMS protocol [6]. It acts as an abstract machine that represent the functionality of the real manufacturing device and its local controller, supplying primitives to be invoked remotely by the client part. The virtual resource components can be re-used for additional and new applications, since the manufacturing resources are independent from the control application.

The virtual resource is developed for each physical device according the specifications of the machine vendors and comprises a set of objects that maps the services of manufacturing resources. The use of MMS specifications in the definition of services is important in order to standardize the approach, but due to the complexity of these specifications, a sub-group of services were defined, closest as possible to the MMS specifications, in order to make things easier and lighter. These services are grouped in re-use libraries, such as the *VR Support*, *Variable Handling*, *Program Handling* and *Events*, as represented in Fig. 3, which shows some services that provide the interaction with the physical manufacturing devices.

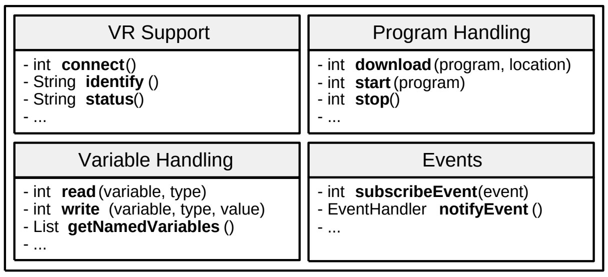
VR Support
- int **connect**()
- String **identify** ()
- String **status**()
- ...
Program Handling
- int **download** (program, location)
- int **start** (program)
- int **stop**()
- ...
Variable Handling
- int **read** (variable, type)
- int **write** (variable, type, value)
- List **getNamedVariables** ()
- ...
Events
- int **subscribeEvent**(event)
- EventHandler **notifyEvent** ()
- ...

**Fig. 3.** Re-use Libraries of Services Provided by the Virtual Resource

The objective, input parameters and return values of available services are always the same, making transparent the development of the holonic or agent-based manufacturing applications from the particular details of each resource, improving the ability to support the heterogeneity.

The interaction between the physical manufacturing resource and the virtual resource is also dependent of the communication platform, such as serial link, fieldbus networks and TCP/IP protocol or different connectivity's applications developed under OPC technology, ActiveX components, etc.

## 4.2 Client-Server Model

The second main concept in the proposed mechanism is the client-server model. The LCD device acts as a client part accessing to the real manufacturing resource by invoking remotely the primitives that represent services in physical resource.

The industrial manufacturing environments are characterized by its heterogeneity, with the distributed processing resources, i.e. computers, industrial controllers and automation devices, running in distinct platforms, such as Windows, Linux and AS400. This heterogeneity requires the use of distributed object platforms to support the interoperability between the clients (operational agents) and virtual resource components. The available technologies to support the distributed object platform are mainly the CORBA, DCOM (Distributed Common Object Model) and RMI (Remote Method Invocation) [4].

CORBA is based essentially in the Object Request Broker (ORB) concept (also designated as software bus or middleware), which allows a local client to invoke methods on a remote platform as if it were local. In order to mask remoteness and networking details, the middleware platform installs end points (stub and skeleton) on the client side and on the server side. The behavior is very close to the Remote Procedure Calls (RPC) mechanism; however, a RPC is offered by a dedicated server, while ORB methods are attached to a client and a server can handle many client. This mechanism requires an independent Interface Definition Language (IDL) for describing interfaces and generating stubs for various target languages, and an object registering mechanism and object locating schemes for unambiguous referencing and easy object access. Java IDL, which is part of Java 2 platform, allows IDL specifications to be compiled into Java interfaces so that java programs can work with a CORBA compliant ORB. Java IDL enables distributed Java applications to transparently invoke operations on remote network services using the industry standard OMG IDL (Object Management Group Interface Definition Language) and IIOP (Internet Inter-ORB Protocol) defined by the OMG consortium. The main advantage of CORBA is to allow object interaction independently of the source language and the execution platform.

Like CORBA, the Java Remote Method Invocation (RMI) is conceptually similar to the RPC, providing a means of communicating between Java applications using normal method calls, and offering the capability for applications to run on different computers. The RMI uses also skeletons to connect the server to the RMI framework, stubs that act as a proxy server in the client's environment, and a registring mechanism to store the location and name of the server object. The major advantages of RMI are its better performance and instead of using an *idl* file as the interface, it uses a normal java class as interface allowing to pass any java object as arguments.

IBM and Sun, with the cooperation of the OMG, jointly developed RMI over IIOP, so called RMI-IIOP, which joined together the interoperability of CORBA and the easy development of RMI. The implementation of the interface platform using RMI-IIOP was the easiest, being necessary to execute two main actions: compile the RMI code with the *-iiop* option, which generates the *stub* and *tie* components, and to start the naming service, using the same procedure as for the CORBA implementation.

The choice of the distributed object platform should take in consideration the easy mapping of MMS-based services, the easy integration with programming environment, the ability to support heterogeneity and the real-time constraints. In the experimental implementation section several platforms will be tested and a comparative analysis will be made.

## 5 Experimental Implementation

Our approach is validated through the integration of two different automation resources within a generic operational holon: a CPM1 PLC from Omron, which is accessed by a RS232 asynchronous line, and an industrial robot IRB1400 from ABB, accessed through TCP/IP using an ActiveX component.

In order to make easier the access to physical manufacturing resources, our goal is to have a common client, which is the operational holon, independent from the resource controller details and using the same generic methods to access to the automation resource services, such as the start, stop and read methods.

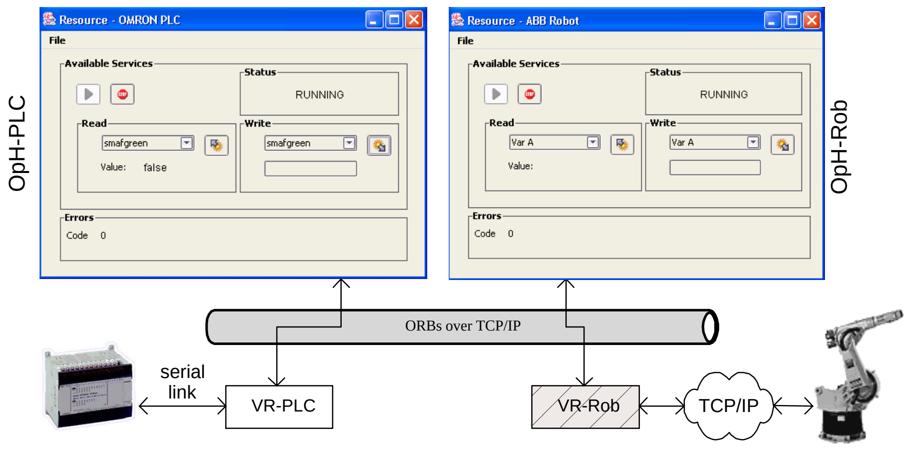

**Fig. 4.** Conceptual Architecture for the Prototype

The prototype, as illustrated in Fig. 4, uses a heterogeneous system environment, and comprises two virtual resources running in Windows XP platforms and customized for each physical resource, one client running in the Windows 2000 platform and another client running in a Linux platform. In order to test the interface between the client and virtual resource components were tested different approaches, namely using the CORBA, RMI and RMI-IIOP platforms.

### 5.1 Development of Virtual Resources

The development of virtual resources, a task for integration specialists, encompasses, for each manufacturing resource, the implementation of the methods available on the client side and described in section 4.1. The client ignores the details of this implementation and each developed virtual resource can be re-used by other similar resources or other holonic control applications.

Since it is intended to integrate two different automation resources, two virtual resources were developed, one for the PLC and another one for the industrial robot. These two servers implement the same services in a different way according the specificities of each resource. In order to illustrate the proposed approach, the implementation of the start service in both virtual resources will be described. In the client side, whatever the resource to be accessed, the invocation always conforms the same template and looks like this:

```
int ret=resource.start(programName);
```

where *resource* is the identifier of the virtual resource that represents the real automation device that is intend to access.

The virtual resource for the programmable logic controller CPM1 is developed according the communication protocol defined by the device [17] and using the serial link for the physical communication with the PLC. The implementation uses the *javax.comm* package, available at `http://java.sun.com`, to support the communication between a java application and an automation device. The code related to the implementation of the start service is illustrated bellow.

```
public int start(String progName){
   int returnCode=-1;
   ...
   returnCode=Write2PLC("@00SC03");
   return(returnCode);
}
```

The primitive essentially writes to the PLC a string containing the *run* command using the *Write2PLC* method, which comprises the message sending to the resource and the wait for the return code. The primitive returns the result of the command execution, returning null in case of success, or a positive number in case of an error.

The development of the virtual resource for the industrial robot was harder than in the previous case, mainly because of the manipulation of the ActiveX component [18]. Since the ActiveX components are adequate for Windows environments, it was necessary to convert the ActiveX component to a java package, using for this purpose the Bridge2Java tool [13]. The start service for the industrial robot is summarised bellow.

```
public int start(String progName){
   ...
   try {
      returnCode=h.s4Run ();
      returnCode=h.s4ProgramLoad(prgID,progName);
      returnCode=h.s4Start(prgID,procedure,nOfCycle,runMode);
   }
   catch(IOException ioe){returnCode=determineErrorType();}
   return (returnCode);
}
```

After the declaration of variables, three commands are executed: *s4Run* that turns on the robot motors, the *s4ProgramLoad* that loads a specified program to the robot controller and the *s4Start* that starts the execution of the loaded program. The primitive returns null in case of success or a positive integer in case of an error.

## 5.2 Analysis of Experimental Implementation

Based in the experimental implementation it is possible to extract some conclusions about the proposed approach for the resource integration. First of all, the resource integration problem become easier since the same holon can access to different manufacturing resources without the need of re-design and re-program, increasing the independency between the control and integration domains. The change or modification in a specific manufacturing resource environment does not affect the control application domain, which continues to invoke the services in the same way.

The second advantage is concerned to the easy development of the virtual resource by integrators and factory automation specialists, which only concentrates in the resource controller details and communication platform, without the need to know details about the control domain.

The third advantage is related to the ability to support heterogeneous environments due to the client-server model. During the experimental implementation, different distributed object platforms where tested. The results for the execution of the *read* service are summarized in Table 1. In Table 1 the VR parameter is concerned to the time spent by the virtual resource to execute the specified service, and the C-S parameter is related to the time spent in the interaction between the client and the server. From the experimental results it is

**Table 1.** Experimental Results

| | | CORBA(ms) | RMI(ms) | RMI-IIOP(ms) |
|---|---|---|---|---|
| OpH-PLC | *VR* | 41.18 | 41.03 | 41.86 |
| | *C-S* | 3.43 | 2.13 | 3.31 |
| OpH-Rob | *VR* | 12.11 | 11.75 | 12.5 |
| | *C-S* | 3.41 | 2.24 | 3.30 |

possible to extract some conclusions related to the behavior of the interface platforms. First, it is possible to verify the independency between the automation devices and the interface platforms: the execution of the service at the virtual resource is independent from the interface platform, and the interaction between the client and the server for each interface platform is independent from the type of automation device. Next, it is possible to compare the different interface platforms. CORBA presents better interoperability between ORB vendors than the RMI interface, since the latter is a Java-to-Java mechanism that limits the scope of its application. On the other hand, RMI presents the easiest interface development and better communication performance, illustrated by the C-S parameter in Table 1. RMI-IIOP presents an intermediate value of communication performance, but overcome the RMI interoperability problems.

## 6 Conclusions

The heterogeneity of industrial manufacturing environments is one of the most important challenge for the distributed manufacturing control systems. The integration of manufacturing resources with the holonic manufacturing control applications remains a problem, because no efficient standard allows an easy, transparent and essentially independent integration.

This paper has proposed an integration approach relying on a unified object-oriented model of the various manufacturing resources, in order to give independence to the holonic or agent-based control applications. The virtual resource concept and the client-server model, inspired by the MMS standard, have been used. In this way, the resource integration problem is reduced to the customization of the server side, where it is necessary to develop virtual resources according to each manufacturing resource specifications and details, which will supply primitives to be invoked remotely by the operational holons.

The proposed approach has been validated: connections between a holonic application and two industrial devices with very different specifications and communication protocols (a programmable logic controller and a robot) were realized and the performances evaluated.

In future work, guidelines for systematic development of virtual resources will be developed, particularly in cases with non standardized protocol.

## References

1. Van Brussel, H., Wyns, J., Valckenaers, P., Bongaerts, L. and Peeters, P.: Reference Architecture for Holonic Manufacturing Systems: PROSA. In: Computers In Industry, 37, (1998) 255-274.
2. Leitão, P. Restivo, F.: A Holonic Control Approach for Distributed Manufacturing. In: Marik, V., Camarinha-Matos, L., Afsarmanesh, H. (eds.): Knowledge and Technology Integration in Production and Services: Balancing Knowledge and Technology in Product and Service Life Cycle. Kluwer Academic Press, ISBN 1-4020-7211-1, (2002) 263-270.
3. Leitão, P. and Restivo, F.: Agent-based Holonic Production Control. In: Proceedings of 3$^{rd}$ International Workshop on Industrial Applications of Holonic and Multi-Agent Systems, Aix en Provence, France, 2-6 September (2002) 589-593.
4. Barata, J., Camarinha-Matos, L., Boissier, R., Leitão, P., Restivo, F. and Raddadi, M.: Integrated and Distributed Manufacturing, a Multi-agent Perspective. In: Proceedings of 3$^{rd}$ Workshop on European Scientific and Industrial Collaboration, Enschede, Netherlands, 27-29 June (2001) 145-156.
5. Foundation for Intelligent Physical Agents, http://www.fipa.org/, (May 2003).

6. ISO/IEC 9506-1: Industrial Automation Systems - Manufacturing Message Specification, Part 1 - Service Definition, (1992).
7. Nussbaumer, H.: Téléinformatique IV. Collection Informatique, Presses Polytechniques Romandes (1991).
8. Boissier, R., Gressier-Soudan, E., Laurent, A. and Seinturier, L.: Enhancing numerical controllers, using MMS concepts and a CORBA-based software bus. In: International Journal of Computer Integrated Manufacturing, **14**(6), (2001) 560-569.
9. Ariza, T., Fernández, F. and Rubio, F.: Implementing a Virtual Manufacturing Device for MMS/CORBA. In: Proceedings of $8^{th}$ IEEE International Conference on Emerging Technologies and Factory Automation, vol. 2, Nice, France 15-18 October (2001) 711-715.
10. Dumant, B., Horn, F., Tran, F. and Stéfani, J.-B.: Jonathan: An Open Distributed Processing Environment in Java.In: Davies, N., Raymond, K. and Seitz, J. (eds.): Midlleware'98: IFIP International Conference on Distributed Systems Platforms and Open Distributed Processing. Springer, ISBN 1-85-233-088-0, (1998) 175-190.
11. Boissier, R., Epivent, M., Gressier-Soudan, E., Horn, F. Laurent, A. and Razafindramary, D.: Providing Real-Time Object Oriented Industriel Messaging Services. In: Proceedings of European Conference on Object Oriented Programming, Brussels, July 1998.
12. Intrinsyc Software, J-Integra tool. Available on http://www.intrinsyc.com, (May 2003).
13. IBM alphaWorks: emerging technologies. Bridge2Java tool. Available on http://www.alphaworks.ibm.com/tech/bridge2java/, (May 2003).
14. Christensen J.H.: IEC 61499 Architecture : Engineering methodologies and software tools. In: Marik, V., Camarinha-Matos, L., Afsarmanesh, H. (eds.): Knowledge and Technology Integration in Production and Services: Balancing Knowledge and Technology in Product and Service Life Cycle. Kluwer Academic Press, ISBN 1-4020-7211-1, (2002) 221-228.
15. Balasubramanian S., Brennan R.W. and Norrie D.H.: An architecture for metamorphic control of holonic manufacturing systems. In: Computers in Industry, **46**(1), (2001) 13-31.
16. Vyatkin V. and Hanisch H.M.: Verification of Distributed Control Systems in Intelligent Manufacturing. In: Journal of Intelligent Manufacturing, **14**(1), (2003) 123-136.
17. Sysmac CQM1/CPM1 Programmable Controllers, Programming Manual. Omron, (1996).
18. RobComm User's Guide, version 3.0/3. ABB Flexible Automation, (1999).