ELSEVIER

Annual Reviews in Control 31 (2007) 81–92

Annual Reviews in Control

www.elsevier.com/locate/arcontrol

# Distributed real-time embedded systems: Recent advances, future trends and their impact on manufacturing plant control

Carlos Eduardo Pereira *, Luigi Carro

*Electrical Engineering Department, Universidade Federal do Rio Grande do Sul (UFRGS), Brazil*

Received 15 December 2006; accepted 16 February 2007

**Abstract**

Real-time and embedded systems have historically been small scale. However, advances in microelectronics and software now allow embedded systems to be composed of a large set of processing elements, and the trend is towards significant enhanced functionality, complexity, and scalability, since those systems are increasingly being connected by wired and wireless networks to create large-scale distributed real-time embedded systems (DRES). Such embedded computing and information technologies have become at the same time an enabler for future manufacturing enterprises as well as a transformer of organizations and markets. This paper discusses opportunities for using recent advances in the DRES area in the deployment of intelligent, adaptive, and reconfigurable manufacturing plant control architectures.


*Keywords:* Manufacturing systems; Real-time systems; Embedded systems

## 1. Introduction

Market demands for innovative, high quality products, aggressive competition at a global scale, increasing productivity through highly optimized production processes, and environmental/societal pressures are some of the challenges faced by the manufacturing industry today. Rapid changes in process technology demand production systems that are themselves easily upgradeable, and into which new technologies and new functions can be readily integrated (Mehrabi, Ulsoy, & Koren, 2000). This situation has created the need for novel manufacturing control systems that are able to manage production change and disturbances, both effectively and efficiently (Van Brussel, Wyns, Valckenaers, Bongaerts, & Peeters, 1998), and has lead to the creation of concepts such as ''flexible manufacturing'' (Draper, 1984), ''holonic manufacturing'' (Van Brussel, 1994), ''agile manufacturing'' (Goldman, Nagel, & Preiss, 1995), and reconfigurable manufacturing'' (Koren & Ulsoy, 1997; Mehrabi et al., 2000). All these approaches aim to incorporate increased levels of flexibility, reconfigurability, and intelligence into manufacturing systems in order to meet highly dynamically marked demands. Molina et al. (2005) presents a good overview on the historical perspective and on key research issues in developing next-generation manufacturing systems.

At the same time, an explosive growth in computer, communication, and information technologies has been experienced and manufacturing plants have also been affected by this ''pervasive and ubiquitous computing'' era. The manufacturing enterprise is intensively deploying a host of hardware/software automation/information technologies in order to face the changing societal environment pulled by the increasing customization of both goods and services as desired by customers (Morel et al., 2005). As the costs of embedding computing becomes negligible compared to the actual cost of goods, there is a trend of incorporating computing and communication capabilities in consumer products, and also in manufacturing equipments. The intelligent manufacturing field has been estimated to be larger than €300 million and growing rapidly (Filos, 2004).

Such embedded computing and information technologies have become at the same time an enabler for future manufacturing enterprises as well as a transformer of organizations and markets (Balakrishnan, Kumara, & Sundaresan, 1999). Almost every aspect of traditional manufacturing needs to be re-examined in view of the newly pervasive computing environment that has come to dominate the manufacturing floor (Johnson &

* Corresponding author.
*E-mail addresses:* cpereira@ece.ufrgs.br (C.E. Pereira), carro@ece.ufrgs.br (L. Carro).

1367-5788/$ – see front matter © 2007 Elsevier Ltd. All rights reserved.
doi:10.1016/j.arcontrol.2007.02.005

Dausch, 2006). A great variety of the so-called e-Work and e-Manufacturing activities (Nof, 2004) – such as virtual manufacturing, augmented reality, intelligent maintenance, intelligent supply, e-logistics – are now feasible and are being adopted by several companies.

However, as pointed out in Morel et al. (2005), ''only a form of technical intelligence that goes beyond simple data through information to knowledge and is embedded into manufacturing systems components and within the products themselves will play a prominent role as the pivotal technology that makes it possible to meet agility/reconfigurability in manufacturing over flexibility and reactivity''.

The INCOM Symposium series is the main technical event supported by the IFAC Technical Committee 5.1 on Manufacturing Plant Control. INCOM aims to discuss the application of automation, information and communication technologies in the control of the manufacturing plant and the entire supply chain within the e-enterprise. While the topics discussed at INCOM embrace all layers of the automation pyramid, from low-level (sensors/actuators/industrial controllers), through manufacturing execution systems to high-level e-enterprise operations (virtual enterprises, supply chain, etc.), this paper mainly focuses on the lower levels, the manufacturing plant control or shop-floor level. At this level, there is a clear trend towards distributed automation architectures, on which automation devices with local processing capabilities are interconnected through industrial communication protocols (Mahalik, 2003). This distributed manufacturing automation architecture heavily relies on an underling architecture composed by the so-called distributed real-time and embedded systems (DRES), ''distributed'' in the sense that devices/machines are physically dispersed, but usually have to exchange information in order to synchronize their operations. The real-time characteristic is due to the fact that the correctness of the system depends not only on the logical results, but also on the time at which these results are produced. While the areas of DRES and ''intelligent, adaptive and reconfigurable manufacturing'' have usually been developed apart, there is an increasing synergy among them, since recent advances in DRES are enabling technologies for the development of future intelligent manufacturing systems.

This paper discusses:

- Requirements imposed by industrial applications to the computational entities embedded in sensors, actuators, controllers, automated guided vehicles, industrial robots, etc.
- Recent advances on real-time distributed and embedded computing elements and how these shall impact the manufacturing plant control area.
- Development methodologies which aim to deploy DRES for industrial applications.

This paper is divided as follows: Section 2 describes some intelligent manufacturing concepts that are enabled by DRES. Section 3 lists requirements that are imposed on DRES to allow their use in industrial applications, while Section 4 gives an overview on recent advances and trends in DRES. In Section 5 some methodologies to develop DRES for industrial applications are discussed and also describes the SEEP methodology. In order to illustrate the benefits that can be achieved by design space exploration when deploying DRES for industrial applications, two case studies are presented in Section 6. Finally, Section 7 presents concluding remarks.

## 2. Examples of intelligent manufacturing areas enabled by embedded computing systems

This section gives an overview on examples of intelligent manufacturing concepts/methodologies which can benefit from advances in DRES.

### 2.1. Agent-based manufacturing systems

The main reason for considering the application of multi-agent systems to industrial applications is that other technologies have not been able to meet all requirements demanded by modern industrial automation systems: effective enterprise integration, handling of cooperative and agile processes, scalability, distribution, fault tolerance, interoperability, among others. In many industrial scenarios, conventional centralized and hierarchical approaches can be inadequate – especially under conditions of disruption and long-term change – to cope with the high degree of complexity and practical requirements for robustness, generality and reconfigurability in manufacturing plant control as well as in production management, planning and scheduling (Mařík & McFarlane, 2005).

These issues have led to the development of a new class of approaches to manufacturing and supply chain decision making. These approaches employ multi-agent systems (MAS), which consist of a set of autonomous, intelligent, and goal oriented units that efficiently cooperate and coordinate their decision making to reach a higher-level or global goal. In MAS groups of agents are organized according to specific, precisely defined principles of community organization and operation (such as messages and negotiation protocols), being supported by an adequate agent platform (Luck, McBurney, & Preist, 2001; Mařík & Lažanský, 2004). Industrial applications of agent-based technologies can be found mainly at three different levels (Mařík & McFarlane, 2005): (i) real-time manufacturing control; (ii) production management level (encompassing problems such as planning, scheduling, orders preprocessing, etc.); and (iii) virtual enterprises (VE) that will integrate manufacturing, sales networks, suppliers, distribution channels.

As long as completely different features and capacities are required for each level, different agent's concepts are applied. For instance, at the real-time manufacturing control, the use of agents with a simple reactive behaviour rather than deliberative behaviour based on complex models and strongly proactive strategies is prevalent. At this level, agents generally have one-to-one correspondence with individual resources (machines and other devices) and orders (raw materials, parts, and products). A more detailed discussion on the timing requirements imposed

on MAS at this level as well as on how existing approaches handle the almost contradictory goals of being adaptive and flexible, while also presenting a deterministic temporal behaviour is presented in (Pereira & Mitidieri, 1999).

As higher the level of the manufacturing automation pyramid, real-time requirements tend to become softer and knowledge intensive decision processes are performed by agents.

Examples of agent-based industrial applications can be found in Shen and Norrie (1999), Mařík and McFarlane (2005), Bussmann, Jennings, and Wooldridge (2004) or Parunak (1999).

### 2.2. Holonic Manufacturing Systems (HMS)

As originally introduced by Koestler (1989), the word ''holon'' aims to describe the hybrid nature of sub-wholes/parts in real-life systems: holons simultaneously are self-contained wholes to their subordinated parts and dependent parts when seen from the inverse direction. From this viewpoint, the whole factory can be considered as a holon being composed of other different kinds of holons such as those representing physical objects like machines, automatic guided vehicles, conveyor belts, pumps, valves, and even products as well as non-physical entities like customer orders, production plans and so on. Holonic Manufacturing Systems (HMS) are manufacturing systems structured as a set of holons, which are autonomous and cooperative units (see Deen, 2003; Van Brussel, 1994; Van Leeuwen & Norrie, 1997). While this definition is somewhat similar to the one presented in agent-based industrial applications, according to Mařík and Lažanský (2004) holons can be considered as specific reactive agents which are strongly connected with the physical level devices, operate in hard real-time and can be organized into a ''holarchical'' structures (hierarchical or heterachical). In Mařík and Pechoucek (2001), mutual impacts of holons and agent concepts are described.

In Van Brussel et al. (1998), a reference architecture for Holonic Manufacturing Systems named PROSA is presented. It consists of three types of basic holons: resource, product and order holons. PROSA makes use of object-oriented concepts such as aggregation and specialization to structure the holons. Holons, as cooperative units, exchange information about the manufacturing system. Product holons and resource holons communicate process knowledge, product holons and order holons exchange production knowledge, and resource holons and order holons share process execution knowledge.

The IEC 61499 standard for the application of function blocks in distributed industrial-process measurement and control systems has been developed by the Holonic Manufacturing Systems (HMS) consortium for the holonic real-time control (Christensen, 1994). Function blocks are used as a container of application algorithms and services wrapped with execution control, promoting reuse of existing algorithms and modular development of manufacturing systems (Fletcher & Brennan, 2001). Additionally, IEC 61499 also includes models for distribution, communication, and services.

As it is discussed later, distributed real-time embedded architectures provide a very interesting infrastructure to deploy such Holonic Manufacturing Systems.

### 2.3. Intelligent maintenance systems

Taking into consideration the fact that machines usually do not suddenly fail, but rather go through a measurable process of degradation before they fail, the basic idea of intelligent maintenance systems or intelligent prognostics systems is to use information provided by sensors and computerized components embedded on the equipments, and apply algorithms for health estimation and failure prediction.

The fundamental basis for such intelligent maintenance systems (see Fig. 1) are the so-called ''infotronic technologies'' (Lee, Qiu, Ni, & Ad Djurdjanovic, 2004), which ''transform the paradigm from precision machine to precise information through the use of intertwined embedded informatics and electronic intelligence in a networked and tether-free environment and

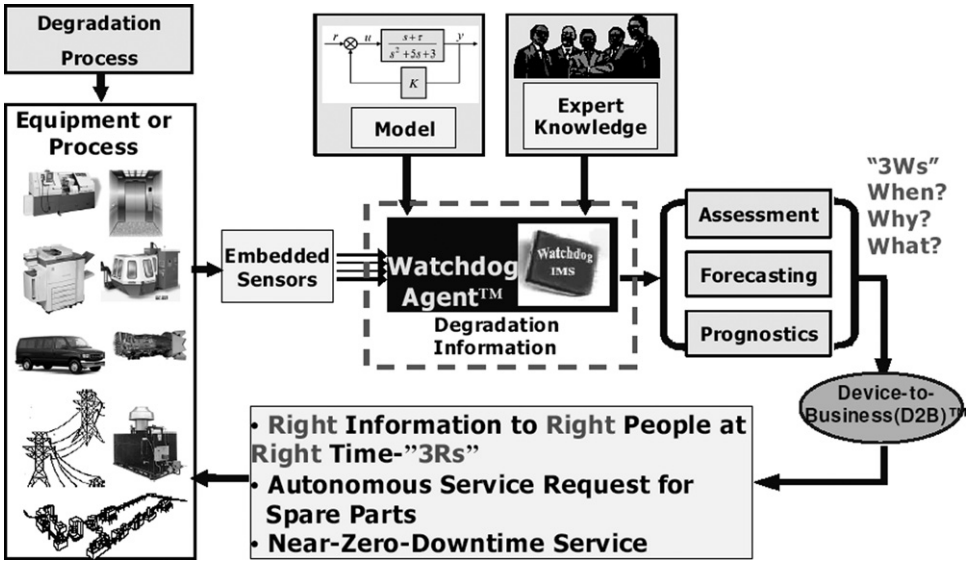


Fig. 1. Intelligent maintenance systems (Lee et al., 2004).

enables products and systems to intelligently monitor, predict and optimize their performance and ultimately to perform self-maintenance activities autonomously''.

Embedded computing components such as embedded sensors, intelligent actuators and processing elements for local smart decision devices play a fundamental role in the development of such intelligent maintenance systems. As it is presented later in the paper, recent advances in areas such Systems-on-Chip (SoC), reconfigurable hardware, etc., are enabling technologies for the development of embedded systems for intelligent prognostics.

### 2.4. Product lifecycle management using product embedded information devices

Product lifecycle management (PLM) can be described as a strategic business approach that applies a consistent set of business solutions in support of the collaborative creation, management, dissemination, and use of product definition information across the extended enterprise from concept to end-of-life—integrating people, processes, business systems, and information (Kiritsis, 2004). PLM consolidates diverse business activities that create, modify and use data to support all phases of a product's lifecycle from ''begin-of-life'' (design, production), middle-of-life (use, maintenance), and end-of-life (recycling, disposal).

A key component in modern PLM systems is the concept of ''smart items'', physical objects that are equipped with embedded computing units to enable close computing of the real world to backend information systems. Such embedded computing units are the so-called PEIDs (product embedded information devices), which usually contain RFID tags, sensor nodes, embedded PCs or similar devices (Anke & Neugebauer, 2006). This technology will allow producers to dramatically increase their capability and capacity to offer high-quality after-sales services while, at the same time, being able to demonstrate responsibility as producers of environmental friendly and sustainable products.

PROMISE is an international project on product lifecycle management and information tracking using smart embedded systems that is being carried on within the scope of the 6th Framework Program of the European Union and as an endorsed IMS project (http://www.promise.no/). PROMISE's main goal is to allow information flow management to go beyond the customer, to close the product lifecycle information loops, and to enable the seamless e-Transformation of Product Lifecycle Information to Knowledge (see Fig. 2).

As depicted in Fig. 2, PROMISE concepts rely on a DRES infrastructure to track objects' identity, status, and location and provide other decision-enabling information, such as embedded predictive e-services.

## 3. Requirements to embedded computing systems for manufacturing automation applications

The great majority of embedded systems currently being developed and deployed are for use in mass markets such as consumer electronics and their use in industrial applications are still a small but increasing percentage. Similarly, the rapid progress in COTS software for mainstream business systems has not yet become as broadly available for DRES.

However, as presented in the last sections, there are several opportunities to use embedded computing systems in advanced industrial applications. However, in order to be applicable to industrial applications, DRES have to meet following requirements:

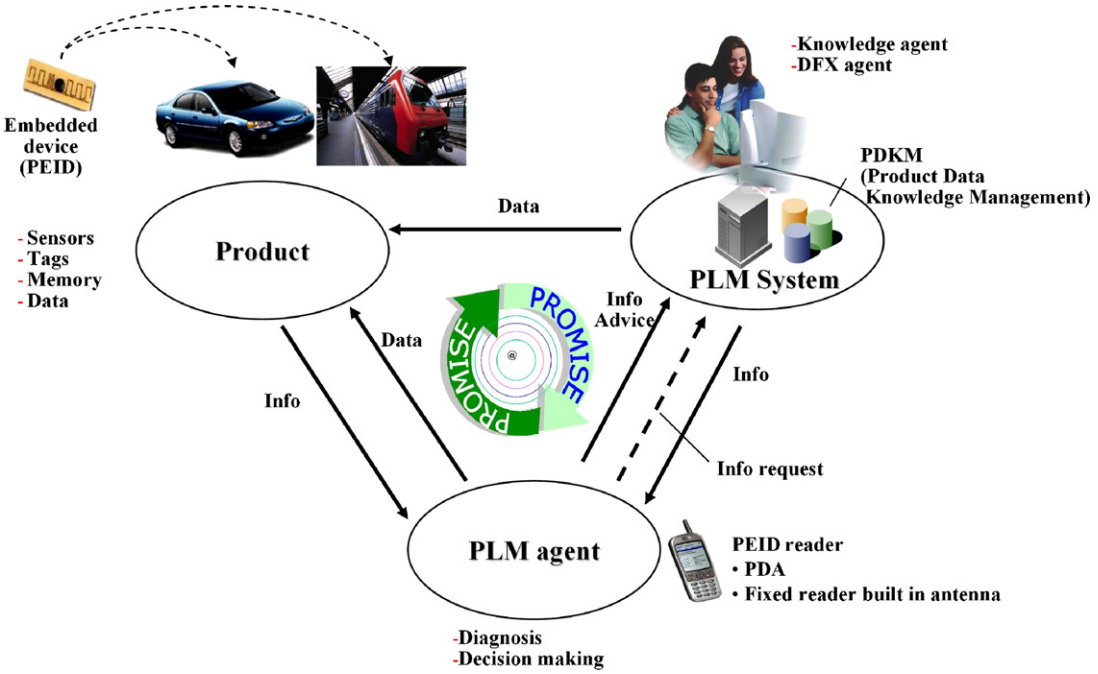


Fig. 2. PROMISE concept (from www.promise.no).

- *Dependability*: it is usually defined as that property of a computer system such that reliance can justifiably be placed on the service it delivers. Both for economical (for instance, high costs of breakdown time) as well as for safety reasons, dependability is a key concept that must be supported by DRES when applied to industrial applications.
- *Real-time communication*: most manufacturing systems are physically distributed over a plant site, so that that their embedded system components will also be physically apart. In order to be able to interact and synchronize while meeting stringent timing requirements, real-time industrial communication protocols must be employed (Mahalik, 2003; Neumann, 2007), so that a timely communication occurs.
- *Flexibility/reconfigurability/agility*: future manufacturing and industrial processes will exhibit much higher degrees of physical reconfigurability in order to accommodate frequent changes in product mix and volume, as well as due to frequent introduction of new product types and manufacturing technology. In addition, rapid reconfiguration will be used much more frequently to recover from machine and process faults with minimal loss of production. In all these cases, the control system (hardware and software) must be quickly reconfigured, and for the most part automatically so, in order not to become a bottleneck to agility.
- *Modularity*: in order to support the above-mentioned requirements of adaptability, DRES must be constructed in a modular way. Modularity also affects serviceability and recyclability in terms of disassembly, separation, repair, and reprocessing (Ishii, 1998).
- *Openness*: a system is defined as open when the implementations of its components conform to an (non-proprietary) interface specification such that upgrading and customization of the system as well as integration of new components is possible (Mehrabi et al., 2000).
- *Location transparency*: communication among different nodes should use names that are not dependent on user's or resource's location. This becomes particularly important when mobile devices/equipment, such as AGVs, are used.
- *Autonomous behaviour*: as discussed in Section 2, a key issue in intelligent manufacturing systems is the capability of manufacturing devices in autonomously making decisions and local data process capabilities.
- *Security*: embedded computing systems need to access, store, manipulate, or communicate sensitive information and frequently need to operate in physically insecure environments. In industrial applications it is mandatory to have a process to prevent and detect unauthorised use of services (Ravi, Raghunathan, Kocher, & Hattangady, 2004) presents a good overview on design challenges for deploying secure embedded systems.

## 4. Distributed real-time and embedded systems (DRES)

### 4.1. Embedded systems hardware and SOCs

Embedded systems are defined as computational resources that take part in bigger systems. They may vary from ABS (antilock braking systems) control in a car to the portable phone or home set-top box, and nowadays they include almost all systems that are not desktop computers. Helped by the technology advances allowed by Moore's law, that states that every 18 months the number of transistors on a single die doubles, embedded systems complexity has increased following the same pace. Semiconductor companies are now fabricating Systems-on-Chip, or SoCs (Bergamaschi et al., 2001), complex devices that include one or more processors, plus memories and communication resources in a single die, at a cost in the range of few dollars. Portable phones are classical examples of products that benefit from such technology.

This way, the availability of fast operating devices with embedded memories and other resources can be almost taken for granted. For example, the same seamless connection that today benefits persons while using a wireless link inside buildings or airports can be transported to the manufacturing scenario. The low cost of RF devices can provide new access means for plant observation and control, without wiring costs and accessibility problems.

Another key issue of current embedded devices is their adaptability. Thanks to the huge processing power available, and the reduced costs of embedded flash memories, these SOCs can be programmed using a regular C or C++ compiler, thus allowing fast adaptability to any task. Moreover, embedded operating systems are also available for these devices, making the porting of new applications an easier task, when compared to the way things should be done in case of dedicated hardware and software platforms.

The last technological evolution to enter the scenario is the possibility of hardware reconfiguration itself (Hartenstein, 2001). Reconfigurable hardware allows for another level of programmability that could boost performance at a fraction of the energy spent by a microprocessor.

Besides regular programmable devices like FPGAs (e.g. www.xilinx.com and www.altera.com are good examples), a new commercial trend is found nowadays, concerning the reconfigurations of processors themselves, either by changing the instruction set or by adding extra hardware tightly connected to the instruction set, like in the Coware and Tensilica approaches (www.coware.com and www.tensilica.com). The advantage of changing the processor concerns the huge amount of software reusability allowed, while still providing extra performance thanks to the dedicated instructions.

This technological scenario shows that embedded systems can be as complex as required, and thanks to their mass production, they can be available at a reasonable cost. From the manufacturing and plant control point of view, these advances might mean that the number of observation and control points might increase, without extra wiring required and without huge investments. Moreover, thanks to their high processing power, embedded SOCs allow for software maintenance at a higher level, with RTOS support and low software development cost, meaning that adaptability on the field can be deployed at a faster pace.

Hardware components in a SoC include one or several processors, even from different types (microcontrollers, DSP, RISC), memories, dedicated components for accelerating

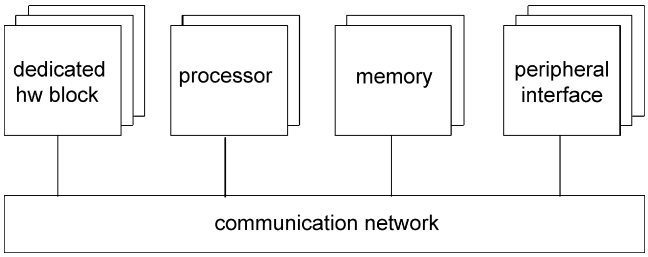


Fig. 3. Typical SoC hardware architecture.

critical tasks, and interfaces to various peripherals. Components are connected by arbitrary communication networks, which may range from a simple bus to hierarchical buses connected by bridges to a complex network-on-chip (De Micheli & Benini, 2002), as illustrated in Fig. 3.

The design of embedded systems is becoming largely software-dominated, and is tightly coupled to a platform (Densmore, Passerone, & Sangiovanni-Vincentelli, 2006). Market perspectives indicate that up to 90% of the embedded system design effort is now on the software part. Software components in a SoC include several application software tasks running on the processors, device drivers, and a real-time operating system (RTOS) (Burns & Wellings, 1997) for each available processor, to support basic services such as scheduling and communication. An RTOS, besides usual functions of an operating system, must also fulfill application temporal requirements, such as task deadlines and frequency of activation of periodic tasks. Temporal restrictions have impact on task scheduling algorithms and on response time of basic OS services such as interrupts and context switching.

Major guidelines for the SoC design are the clear separation between computation and communication (Rowson & Sangiovanni-Vincentelli, 1997) and between function and architecture (Keutzer, Malik, Newton, Rabaey, & Sangiovanni-Vincentelli, 2000; Sangiovanni-Vincentelli & Martin, 2001). These distinctions enforce modular design and promote the independent design and evolution of three aspects of system design: function, architecture, and communication.

As depicted in Fig. 4, the design of an embedded SoC starts with the definition and validation of a high-level, pure functional specification, which is not influenced by architectural choices and does not consider how design requirements (power, performance) may be fulfilled. Following a design space exploration step, which is nowadays mainly a manual task, an abstract macro-architecture is defined to implement this functionality, and a mapping assigns functional blocks to architectural ones. This mapping implements a hardware–software partitioning, where some functions are mapped to software tasks while other ones are mapped to dedicated hardware blocks. This high-level architectural model abstracts all low-level implementation details.

A performance evaluation of the system is then performed, by using estimates of the computation and communication costs for this macro-architecture. Results of this estimation will be used to guide the design space exploration, for instance requiring modifications in the chosen macro-architecture and/or functional mapping.

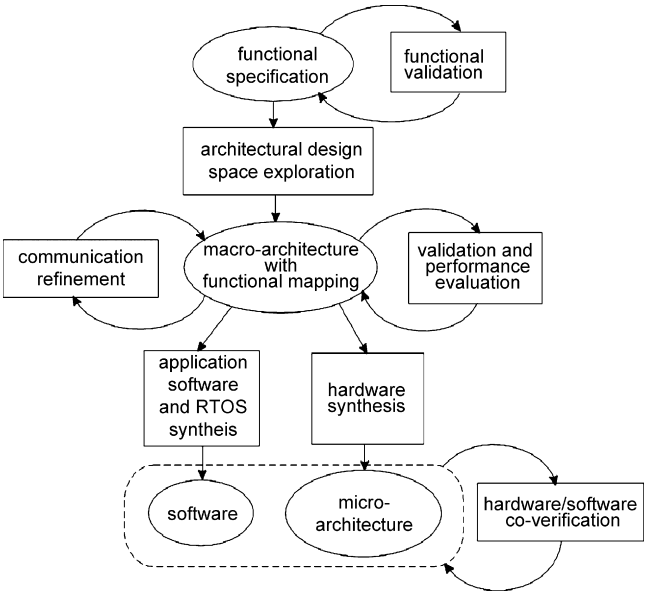


Fig. 4. A generic SoC design methodology.

Communication refinement is now possible, where high-level communications are mapped to particular mechanisms, protocols and channels, thus allowing a more precise performance evaluation. Depending on the chosen communication mechanisms, specialized components such as DMA and interrupt controllers and bus arbiters must be inserted. Some approaches implement the automatic generation of these components (Lyonnard, Yoo, Baghdadi, & Jerraya, 2001; O'Nils & Jantsch, 2001).

Hardware and software synthesis follow, usually resulting in C/C++ code for the software part and an HDL description of a cycle-and-pin accurate micro-architecture for the hardware part. This synthesis is largely automated, and the combined C-HDL description may be validated by conventional co-simulation tools, such as CoCentric System Studio (Synopsys, 2003), from Synopsys, and Seamless CVE (Mentor, 2004), from Mentor. As part of the software synthesis process, an operating system may be required, to implement for instance services for communication among tasks and for scheduling tasks that are mapped to the same processor.

### 4.2. Middleware and programming languages

Real-time and embedded systems have historically been relatively small scale. As discussed in the last section, recent advances in microelectronic and software now allow embedded systems to be composed of a large set of processing elements, and the trend is towards significant increased functionality, complexity, and scalability, since those systems are increasingly being connected by wired and wireless networks to create large-scale DRES. Additionally, the environment is generally non-static, and the whole system must be robust enough in order to operate under highly unpredictable and changeable conditions. An important and challenging problem for DRE systems is therefore adaptation of behaviour and reconfiguration of resources to maintain the best possible application

performance in the face of changes in system load and available resources (Schantz et al., 2006). Therefore, new techniques and software solutions are required in order to properly handle the increased system complexity.

Clearly, a changing environment requires extra adaptability from the software and hardware resources. Not only one must provide for these adaptations, but also one must take into account the possibilities of exploiting the architecture of the system itself. For example, as more intelligent nodes are available at low cost, distributed processing starts to make a lot of sense, since the cost of local processing is not only energetically efficient, but it is important to notice that bandwidth does not follow the advances of Moore's law.

The need for autonomous and time critical behaviour in manufacturing plant control demands a flexible distributed system substrate that adapt robustly to dynamic changes in application requirements and market/environment conditions. This substrate is usually called ''middleware'' because they sit ''in the middle'' in a layer above operating systems and networking SW/HW and below industry specific applications (Bernstein, 1996). Schantz and Schmidt (2001) define middleware as reusable systems software that functionally bridges the gap between: (i) the end-to-end functional requirements and mission doctrine of applications and (ii) the lower-level underlying operation system and network stack protocols. Middleware therefore provides capabilities whose quality and quality of service (QoS) are critical to DRE systems.

Similar to network protocols, middleware can also be decomposed in several layers (Schmidt, 2002): from ''host infrastructure middleware'' at lower level to ''domain specific middleware services'' at higher level (below the ''application level''). Each of these layers focuses on specific aspects, but all have in common the idea of allowing the implementation of an ''information utility'', to which components such as manufacturing devices can be connected and are able to interact with each other.

Some of the middleware provided functionalities are: (i) to encapsulate and enhance native OS communication and concurrency mechanisms to create portable and reusable network programming components (connection management, data transfer, parameter (de)marshalling, etc.), (ii) to minimize hardware and software infrastructure dependencies, and (iii) to allow management of processor, memory, and communication resources.

Domain-specific middleware services are tailored to the requirements of a particular domain and have the most potential to increase the quality and decrease the cycle-time and efforts that integrators require to develop a particular class of DRE systems (Schmidt, 2002). For instance, for manufacturing plant control applications, this middleware level could for instance, support all holon types defined by the PROSA reference architecture as discussed in Section 2. It is also important to note that in applications such as manufacturing plant control, middleware must allow functional and QoS-related properties to be modified dependably, i.e. without compromising the fulfillment of stringent timing requirements. That means, a key aspect is to achieve a good balance for the trade-off performance versus flexibility.

TAO (Schmidt, Levine, & Mungee, 1998) and QuO (Loyall et al., 1998) are examples of existing middleware that have been used for manufacturing plant control applications.

Language support for embedded software development efforts are currently centred on the C and C++ programming languages but Java is rapidly gaining momentum in this field. Devices with embedded Java such as cellular phones, PDAs and pagers have grown from 176 million in 2001 to nearly 800 million in 2005. It has been predicted that at least 80% of mobile phones will support Java by this year (Lawton, 2002). A key concept in Java is that Java byte codes can be run on any architecture for which a customized Java Virtual Machine (JVM) is available. The JVM encapsulates all platform-specific services, such as networking, file system operations, etc. The idea of using Java for embedded industrial applications is not new (see for instance (Atherton, 1998)) and with the advent of the so-called Real-Time Specification for Java (RTSJ) (Bollella, Gosling, & Benjamin, 2001), a very interesting alternative to the development of DRES became available. RTSJ incorporates several useful real-time constructs into Java, allowing periodic activation of concurrent processes, timed actions, handlers for asynchronous events, etc. Wehrmeister, Becker, and Pereira (2004) presents an approach to optimize the deployment of real-time embedded applications based on RTSJ.

### 4.3. Real-time communication protocols

Real-time communication protocols are an important component in DRES for industrial applications in order to ensure a safe and timely operation. Fiedlbus protocols such as Profibus, Foundation Fieldbus, DeviceNet and CAN are standardized (IEC, 2003), widely adopted and well established at field/shop floor level (see for instance Mahalik, 2003). Ongoing efforts in extensions to these industrial communication protocols have shifted from low level aspects (physical and data link layer standards) to the definition of higher-level automation objects, such as Profinet mechatronic objects (Profibus, 2002) or CIP application layer objects (www.odva.org).

At the same time, Ethernet has also been considered for use in real-time applications, either in the industrial domain or in large embedded systems. Attractive factors include wide availability, high bandwidth and low cost. The use of Ethernet-based communication protocols should also enable an easy integration to realise the access to data in various layers of an enterprise information system. As already mentioned, these different levels impose different requirements dictated by the nature and type of information being exchanged. Due to fact that Ethernet was not originally developed to meet real-time requirements and the medium access control protocol used – CSMA/CD – may cause unbounded network access delays, several proposals have been presented to adapt this protocol in order to achieve real-time behaviour (Neumann, 2007; Pedreiras & Almeida, 2005).

### 4.4. Future trends in DRES

The continuous trend towards smaller, more intelligent, and more numerous devices is continuous and very soon also embedded computing systems will be approaching limits of human capability to develop, operate and maintain these systems. It can be observed that objects are becoming intelligent ''things that think'' (Gershenfeld, 1999) and while this scenario will enable the realization of several high value e-manufacturing and e-services, completely new design and operation paradigms have to be developed. This has been the motivation to the creation of areas such as autonomic computing (Kephart & Chess, 2003) and organic computing (DFG, 2004; Schmek, 2005). The idea is to develop (embedded) computing systems that manage themselves given high-level objectives and are able to perform self-configuration, self-optimization, self-healing and self-protection (the so-called ''self-*x*'' properties). Those systems should have sufficient degrees of freedom to allow a self-organized behaviour which will adapt to dynamically changing requirements. The ability to deal with widely varying time and resources demands while still delivering dependable and adaptable services with guaranteed temporal qualities is a key aspect for future DRES (Stankovic, 1996). Some examples of autonomic and reconfigurable embedded real-time systems can be found in Brinkschulte et al. (2004), Rammig et al. (2006) and Götz, Rettberg, and Pereira (2005).

## 5. Methodologies

Clearly, the possibility of using low cost devices that can be configured, either in software or in hardware to adapt its characteristics to those better matching the underlying environment is beneficial to the whole design process. The major challenge, however, is how to provide engineers with effective and productive tools to allow them to make these complex systems in a timely manner. To develop the next generation of open, modular, reconfigurable, maintainable, and dependable manufacturing systems adequate methodologies must be available. When dealing with complex industrial automation applications, the definition of a good architecture is of utmost importance. Aspects such as modularity, cohesion, and coupling, which historically were relegated to a secondary plan due to an overemphasis on systems performance, have a major impact in installation, operation, maintenance, and engineering costs. Object-oriented systems have important and desirable architectural properties. They are composed of a number of communicating and well-defined objects. Objects with common characteristics and behaviours are organized into classes. Class hierarchies can be built using inheritance concepts. Objects also fit nicely with concurrence, since their logical autonomy makes them a natural unity for concurrent execution. That implies in a fruitful way of thinking, enabling concurrent processes present in the real world to be expressed in a natural and easily understandable way. Some examples of object-oriented technologies for industrial applications are described in the sequence.

SIMOO-RT (Becker & Pereira, 2002) is an object-oriented framework to the development of real-time computer-based systems (i.e. hardware and software) that are embedded in devices used in flexible and adaptive industrial automation systems. The approach is based on the concept of active objects, which are autonomous and concurrent processing units, having their own thread of control. Active objects are used to map the structure and the desired behaviour of technical plant components. The approach leads to a generic specification, which preserves the semantics of the physical plant under automation. The approach covers the whole lifecycle of industrial automation systems: from requirements engineering, through hardware and software design, and to implementation and validation.

DOME (Distributed Object Model Environment) is an event driven, object-oriented distributed architecture on which industrial are defined as a network of Automation Objects, which can be assigned to different contexts on several heterogenous nodes. The access to the process interface is also encapsulated inside DOME objects, which can be treated as a kind of service interface function block (SIFB) according to IEC 61499 or as a Proxy-object. A DOME case study is presented in Riedl, Diedrich, and Nauman (2006).

OONEIDA (Vyatkin, Christensen, & Lastra, 2005) is a research and development initiative in the domain of decentralized, agile industrial control and automation for both discrete manufacturing and continuous process systems. OONEIDA should enable all players in the automation value creation chain to encapsulate their intellectual property into the software components and to deploy these components into intelligent devices, machines, systems, and automated factories, respectively. This will enable time- and cost-effective specification, design, validation, realization, and deployment of intelligent mechatronic components in distributed industrial automation and control systems.

Ptolemy (http://ptolemy.eecs.berkeley.edu) is a design methodology based on the description of complex behaviour with the use of OO languages. The main goal of Ptolemy is to raise the abstraction level on the design process of systems composed of hardware and software components. The project goal is to cover the modelling, simulation and design of concurrent components. Several models of computation are supported, and basically the task of the designer is to use a heterogeneous mixture of several models to describe and simulate a complex system.

SEEP is a design and verification methodology that allows the development of DRES from high-level RT-UML models. The SEEP project (SEEP, 2006) concerns the development of embedded systems based on platforms. A platform is defined as a combination of hardware and software resources, where the customization of these resources is developed for a target application.

In the specification phase, the user might use UML diagrams, and a Simulink frontend is being currently adapted. From UML one can make the fist design exploration, by using the developed library plus a tool that evaluates the costs, in terms of processing speed, memory and power of the available solutions (Brisolara, Becker, Carro, Wagner, & Pereira).

After this first design exploration step, SEEP supports two different platforms, the Power-PC and a Java processor in several organizations (pipeline, VLIW and with a reconfigurable array). Each of these processor versions has a different trade-off regarding area, power dissipation and speed. Also, each one can be synthesized only with a dedicated instruction set, strongly tied to the problem it is trying to solve, so that maximum efficiency can be achieved with software compatibility. Examples of such architectures being deployed can be found in Krapf and Carro (2003), Beck and Carro (2004, 2005), and Silva et al. (2006).

## 6. Case studies

In order to better illustrate possibilities of design space exploration when synthesizing embedded systems for manufacturing automation applications, two case studies are presented.

### 6.1. Customizable RT-Java-based SOC for holonic/agent applications

The first case study deals with the deployment of a SOC for the holonic manufacturing case study described in McFarlane (2002), real-time manufacturing control tested which consists of a flexible production cell containing a robot arm, a screwing robot, a rotary table, a flipping unit, an item parts input and a storage unit to store the assembled items. Different product types are assembled according to product specification.

The proposed case study was implemented using the SEEP methodology described in previous section. A RT-UML model encompassing around 30 classes was created (due to restrictions in paper's length this diagram is not presented here) and a SOC containing a customized RT-FemtoJava, an API based on the Real-Time Specification for Java (RTSJ) (Wehrmeister et al., 2004), and a real-time communication API (Silva et al., 2006) was synthesized using the SASHIMI synthesis tool (Ito, Carro, & Jacobi, 2001). This case study is partially implemented, however all important functions are present in the current system version. Table 1 depicts the achieved optimization. The first column describes the required footprint for a single node application, i.e. when all system objects inhabit in one computing device. The optimized system requires 88.28% less hardware resources when compared with the same application running on a standard JVM and processor. Considering that most Java applications need to be executed using a virtual machine even when running on a single node, the reduction would be 96.27%, because most of the code included in the non-optimized application would not be used. In the distributed version, the application must include the real-time communication API and the RTSJ-base API in each system node. Even with this extra code the reduction remains about 88% to application code size and about 96% when compared to the same application running on a commercial RTSJ-compatible JVM.

### 6.2. Embedded prognostics SOC

As a second case study, let us consider the design of an embedded prognostics SoC to be used in intelligent prognostics or condition-based monitoring applications. The goal here is to use intelligent embedded prognostics algorithms in order to perform a continuous assessment and prediction of a product's degradation performance by extracting high-level information – the so-called features – from sensory signals. As discussed in Lee et al. (2004), examples of such algorithms include Kalman filters, time-frequency based, time-series based and wavelet-based system analysis, Autoregressive Moving-Average

Table 1
Flexible assembly—HW/SW optimization

| | Single node | Distributed nodes | | |
|---|---|---|---|---|
| | | Robot arm | Screwing robot | Assembly |
| Original code size | | | | |
| Application | 32.730 | 10.712 | 9.023 | 6.331 |
| RTSJ API | 28.428 | 28.428 | 28.428 | 28.428 |
| Comm. API | | 23.702 | 23.702 | 23.702 |
| Total | 61.158 | 62.842 | 61.153 | 58.461 |
| Total + RTSJ JVM[a] | 192.230 | 193.914 | 192.225 | 189.533 |
| Synthesized code size | | | | |
| Application | 4.050 | 1.059 | 815 | 2.932 |
| Used API classes | 3.116 | 3.116 | 3.116 | 3.116 |
| | | 3.247 | 3.247 | 3.247 |
| Total | 7.166 | 7.422 | 7.178 | 9.295 |
| Reduction (bytes) | | | | |
| App. code size | 53.992 | 55.420 | 53.975 | 49.166 |
| App. + RTSJ JVM | 185.064 | 186.492 | 185.047 | 180.238 |
| Reduction (%) | | | | |
| App. code size | 88.28 | 88.19 | 88.26 | 84.10 |
| App. + RTSJ JVM | 96.27 | 96.17 | 96.27 | 95.10 |

[a] The RTSJ JVM size was based on a commercial embedded JVM.

(ARMA) analysis, Hidden Markov Models, sensor fusion techniques, etc. For instance the IMS Center for Intelligent Maintenance Systems in U.S. (IMS Center, 2006) has developed a set of around 20 prognostic tools, based on different algorithms, which are used for feature extraction, performance assessment, diagnostics and prognostics and have been successfully applied to different industrial testbeds. More than one algorithm can be applied in a particular application and obtained results can be fused using some averaging technique in order to increase the robustness and quality of obtained confidence values.

An embedded system to implement such an embedded prognostic device includes: I/O modules for data acquisition of digital and analog signals, communication modules, both for industrial communication protocols as well as for Internet connectivity (for instance, with an embedded Web server to allow remote access and configuration), a database, an embedded processor, an embedded OS, and embedded software with executable codes for the prognostics algorithms.

Probably the most straightforward implementation of such embedded prognostic device with current technologies would be to use some commercial embedded processor, running an embedded RTOS with prognostic algorithms being implemented in programming languages like C/C++ and being executed as concurrent tasks under the RTOS. However, considering that prognostics algorithms are very distinct with regard to their processing models, an optimized embedded system synthesis can be achieved by exploring the design space evaluating different customizable processor architectures. For instance, as presented in (Beck & Carro, 2005) different usage of a customizable DSP-based processor could lead to performance improvements in the range of a factor of 4, with energy reduction by a factor of even 10, at the price of extra area. So, depending of the effective design goals (area, power dissipation, just performance or a combination of them all), the tuning of the processor to the specific system it must work in can provide a good balance within the design space, with a reasonable goal. Moreover, since all the different microprocessor descriptions are done in VHDL and synthesized of any FPGA available in the market, dynamically changing them whenever required by the upgrade of application allows for easy adaptability without performance loss.

## 7. Concluding remarks

As discussed in the paper, looking from a top-down perspective, modern manufacturing systems are challenged to incorporate increasing capabilities of reconfigurability, self-*x* and intelligence in order to be able to succeed in a very competitive and global market, on which product variety and complexity increase, product lifecycle shrinks, quality requirements increase, and profit margins decrease. Fortunately, when considering a bottom-up perspective, one can identify that considerable advances have been made in the last years in communication, computer and information technologies. These advances are allowing the deployment of distributed and real-time embedded computing architectures that can become key enablers to the development of reconfigurable/intelligent manufacturing machines.

The paper described recent advances in DRES, and presented some case studies that take benefit of some of these new technological conditions. As technology evolves, the immediate challenge is how to allow designers to deploy this technology in the field. The test cases showed that not only this is possible with available tools, but also that the research must continue to cover new aspects that are being leveraged by software and hardware technology advances.

## Acknowledgements

This work has been partly supported by the Brazilian research agencies CNPq, Fapergs, and FINEP.

## References

Anke, J., & Neugebauer, M. (2006). Early data processing in smart item environments using mobile services. In *Proceedings of the IFAC 12th INCOM 2006*.

Atherton, R. W. (1998). Moving Java to the factory. *IEEE Spectrum, 1998*, 18–23.

Balakrishnan, A., Kumara, S., & Sundaresan, S. (Jul 1999). Manufacturing in the digital age: Exploiting information technologies for product realization. *Information Systems Frontiers, 1*(1), 25–50.

Beck, A., & Carro, L. (2005). Dynamic reconfiguration with binary translation: Breaking the ILP barrier with software compatibility. In *Proceedings of the 42nd Design Automation Conference* (pp. 732–737).

Beck, A. C., & Carro, L. (2004). A VLIW low power Java processor for embedded applications. *ACM SBCCI 2004* (pp. 157–162)1-58113-947-0.

Becker, L. B., & Pereira, C. E. (2002). SIMOO-RT—an object-oriented framework for the development of realtime industrial automation systems. *IEEE Transactions of Robotics and Automation, 18*(4), 421–430.

Bergamaschi, R. A., Bhattacharya, S., Wagner, R., Fellenz, C., Muhlada, M., White, F., et al. (2001). Automating the Design of SOCs Using Cores. In *IEEE Design & Test of Computers* (p. 32–45).

Bernstein, P. A. (1996). Middleware: A model for distributed systems services. *Communications of the ACM, 39*(2), 86–98.

Bollella, G., Gosling, J., & Benjamin, B. (2001). The Real-Time Specification for Java, http://www.rtj.org/rtsj-V1.0.pdf.

Brinkschulte, U., Ungerer, T., & Becker, J. (2004). CARUSO: An approach towards low power autonomic SoCs for embedded RT applications. In *Proceedings of the 18th IEEE Parallel and Distributed Processing Symposium*.

Brisolara, L., Becker, L. B., Carro, L., Wagner, F. R., & Pereira, C. E. (2005). UML for SoC design. In Martin, G., & Muller, W. Eds. *A comparison between UML and function blocks for heterogeneous SoC design and ASIP generation*. Vol. 1 (pp.199–222). Springer. Chapter 9, ISBN 0-387-25744/6.

Burns, A., & Wellings, A. (1997). *Real-time systems and programming languages*. Addison-Wesley.

Bussmann, S., Jennings, N. R., & Wooldridge, M. (2004). *Multiagent systems for manufacturing control: A design methodology*. Springer-Verlag.

Christensen, J. H. (1994). Holonic manufacturing systems: Initial architecture and standards directions. In *Proceedings of the First European Conference on Holonic Manufacturing Systems* (pp. 1–20).

Deen, S. M. (Ed.). (2003). *Agent based manufacturing: Advances in the Holonic Approach*. Heidelberg: Springer Verlag.

De Micheli, G., & Benini, L. (2002). Networks-on-Chip: A new paradigm for systems-on-chip design. IEEE DATE'02—Design, Automation and Test in Europe, Paris, 2002.

Densmore, D., Passerone, R., & Sangiovanni-Vincentelli, A. (September–October, 2006). A platform-based taxonomy for ESL design. IEEE design and test of computers, p. 359–374.

DFG (2004). DFG SPP 1183. *Organic computing*. http://www.organic-computing.de/spp.

Draper, C. (1984). *Flexible Manufacturing Systems Handbook, Automation and Management Systems Division*. The Charles Stark Draper Laboratory Inc Noyes Publications.

Filos, E. (2004). European research and policies for knowledge-driven innovation. The example of industrial informatics. In *Proceedings of the second IEEE international conference on industrial informatics (INDIN'04)* (pp. 11–12).

Fletcher, M., & Brennan, R. W. (2001). Designing Holonic manufacturing systems using the IEC 61499 (function block) architecture. *IEICE Transactions* (Vol. E84-D, No. 10, p.1398–1401).

Gershenfeld, N. (1999). *When things start to think*. New York: Owl Books, Henry Holt.

Goldman, S. L., Nagel, R. N., & Preiss, K. (1995). *Agile competitors and virtual organizations: Strategies for enriching the customer*. New York: Van Nostrand Reinhold.

Götz, M., Rettberg, A., & Pereira, C. E. (2005). Towards run-time partitioning of a real time operating system for reconfigurable systems on chip. In *Proceedings of IESS*.

Hartenstein, R. (2001). A decade of reconfigurable computing: a visionary retrospective. IEEE DATE (p. 642–649).

IEC. (2003). IEC 61784-1. Digital data communications for measurement and control. Part 1. Profile sets for continuous and discrete manufacturing relative to fieldbus use in industrial control systems.

IMS. (2006). NSF I/UCRC Center for Intelligent Maintenance Systems, www.imscenter.net.

Ishii, K. (1998). Modularity: A key concept in product lifecycle engineering. In A. Molina & A. Kusiak (Eds.), *Handbook of life-cycle enterprise*. Kluwer.

Ito, S. A., Carro, L., & Jacobi, R. P. (2001). Making Java work for microcontroller applications. *IEEE Design and Test of Computers, 18*(5), 100–110.

Johnson, T., & Dausch, M. (2006). Sensor informatics for manufacturing. In *Proceedings of the IFAC 12th INCOM 2006*.

Kephart, J. O., & Chess, D. M. (2003). The vision of autonomic computing. *Computer, 36*(1), 41–50.

Keutzer, K., Malik, S., Newton, A. R., Rabaey, J., & Sangiovanni-Vincentelli, A. (December 2000). System-level design: orthogonalization of concerns and platform-based design. *IEEE Transactions on Computer-Aided Design of Integrated Circuits, 19*(12), 1523–1543.

Kiritsis, D. (2004). Ubiquitous product lifecycle management using product embedded information devices. In *Proceedings of 2004 intelligent maintenance systems (IMS) international conference*.

Koestler, A. (1989). *The ghost in the machine*. London: Arkana Books.

Koren, Y., & Ulsoy, A. G. (1997). Reconfigurable manufacturing systems. Engineering research center for reconfigurable machining systems. Report #1. University of Michigan, Ann Harbor.

Krapf, R., & Carro, L. (2003). Efficient Signal Processing in Embedded Java Systems. *ISCAS 2003* (p. IV-61–64). ISBN 0-7803-7762-1.

Lawton, G. (2002). Moving Java into mobile phones. *Computer, 35*(6), 17–20.

Lee, J., Qiu, H., Ni, J., & Ad Djurdjanovic, D. (2004). Infotronics technologies and predictive tools for next-generation maintenance systems. In *Proceedings of the 11th IFAC INCOM 2004*.

Loyall, J. P., Schantz, R. E., Zinky, J. A., & Bakken, D. E. (1998). Specifying and measuring quality of service in distributed object systems. In *Proceedings of IEEE ISORC '98*.

Luck, M., McBurney, P., & Preist, C. (2001). Agent technology: Enabling next generation computing. A roadmap for agent-based computing, AgentLink. www.agentlink.org/roadmap.

Lyonnard, D., Yoo, S., Baghdadi, A., & Jerraya, A. (2001). Automatic generation of application-specific architectures for heterogeneous multi-processor system-on-chip. *DAC'01—design automation conference*.

Mahalik, N. P. (Ed.). (July 2003). *Fieldbus technology: industrial network standards for real-time distributed control*. Springer Verlag. ISBN: 3540401830.

Mařík, V., & Lažanský, J. (2004). Industrial applications of agent technologies. In *Proceedings of the 11th IFAC INCOM 2004*.

Mařík, V., & McFarlane, D. (2005). Industrial adoption of agent-based technologies. In IEEE intelligent systems (Vol. 20, No. 1, p. 27–35). ISSN 1094-7167.

Mařík, V., & Pechoucek, (2001). Holons and agents: Recent development and mutual impacts. In *Proceedings of the 12th International Workshop on Database and Expert Systems Applications*.

McFarlane, D. (2002). Auto-ID based control—An overview. *Whitepaper*. http://www.autoidcenter.co.uk/research/CAM-AUTOID-WH-004.pdf.

Mehrabi, M. G., Ulsoy, A. G., & Koren, Y. (2000). Reconfigurable manufacturing systems: Key to future manufacturing. *Journal of Intelligent Manufacturing, 11*(4), 403–419.

Mentor. (2004). *Seamless CVE*. http://www.mentor.com/seamless.

Molina, A., Rodriguez, C., Ahuett, H., Cortés, J., Ramírez, M., Jiménez, G., et al. (2005). Next-generation manufacturing systems: Key research issues in developing and integrating reconfigurable and intelligent machines. *International Journal of Computer Integrated Manufacturing, 18*(7), 525–536.

Morel, G., Valckenaers, P., Faure, J. M., Pereira, C., & Diedrich, C. (2005). Manufacturing plant control: Challenges and open issues. In *Proceedings of the 16th IFAC Triennial World Congress*.

Neumann, P. (2007). Industrial communication: What is going on? *Control Engineering Practice* (special issue with INCOM 2004 selected papers), doi:10.1016/j.conengprac.2006.10.004.

Nof, S. (2004) Collaborative e-Work and e-Mfg.: The state of the art and challenges for production and logistics managers. *Keynote paper at 11th IFAC INCOM 2004*. Salvador, Brazil. (In C. Pereira, G. Morel, & P. Kopacek (Eds.). *Information Control Problems in Manufacturing 2004*. Elsevier Science. ISBN-13: 978-0-08-044249-5).

O'Nils, M., & Jantsch, A. (2001). *Device driver and DMA controller synthesis from HW/SW communication protocol specifications*. Design automation for embedded systems (Vol. 6, No. 2), Kluwer Academic Publisher.

Parunak, H. (1999). Industrial and practical applications of DAI. In G. Weiss (Ed.), *Multiagent systems: A modern approach to dist.artificial intelligence* (pp. 377–416). MIT Press.

Pedreiras, P., & Almeida, L. (2005). Approaches to enforce real-time behavior in Ethernet. In R. Zurawski (Ed.), *The industrial communication technology handbook*. Boca Raton, FL: CRC Press.

Pereira, C. E., & Mitidieri, C. (1999). Multi-agent systems from a real-time perspective. *Multi-agent-systems in production*. ISBN 0-08-043657-9.

PROFIBUS Guideline. (2002). *PROFInet architecture description and specification, Version 1.9*. Karlsruhe: PNO.

Rammig, F., Gotz, M., Heimfarth, T., Janacik, P., & Oberthur, S. (2006). Real-time operating systems for self-coordinating embedded systems. *IEEE ISORC*.

Ravi, S., Raghunathan, A., Kocher, P., & Hattangady, S. (2004). Security in embedded systems: Design challenges. *Transactions on Embedded Computing Systems, 3*, 461–491.

Riedl, M., Diedrich, C., & Nauman, F. (2006). Event driven applications for automation area. In *Proceedings of the IFAC 12th INCOM 2006*.

Rowson, J., & Sangiovanni-Vincentelli, A. (June 1997). Interface-based design. In *Proceedings of the DAC'97—Design Automation Conference*.

Sangiovanni-Vincentelli, A., & Martin, G. (2001). Platform-based design and software design methodology for embedded systems. *IEEE Design & Test of Computers*.

Schantz, R., & Schmidt, D. (2001). *Middleware for distributed systems: Evolving the common structure for network-centric applications. Encyclopedia of software engineering*. Wiley&Sons.

Schantz, R., Loyall, J., Rodrigues, C., & Schmidt, D. (2006). Controlling quality-of-service in distributed real-time and embedded systems via adaptive middleware. http://www.cs.wustl.edu/∼schmidt/PDF/AM.pdf.

Schmek, H. (2005). Organic computing—A new vision for distributed embedded systems. *IEEE ISORC*.

Schmidt, D. (2002). R&D advances in middleware for distributed, real-time, and embedded systems. *Communications of the ACM special issue on Middleware 45(6)*.

Schmidt, D., Levine, D., & Mungee, S. (1998). The design and performance of the TAO real-time object request broker. *Computer Communications Special Issue on Building Quality of Service into Distributed Systems 21(4)*.

SEEP. (2006). Sistemas Eletrônicos Embarcados baseados em Plataformas. *Platform-based Embedded Systems Development*. www.inf.ufrgs.br/~lse.

Shen, W., & Norrie, D. H. (1999). Agent-based systems for intelligent manufacturing: A state-of-the-art survey. *Knowledge and Information Systems, 1*(2), 129–156.

Silva, E., Jr., Freitas, E., Wagner, F., Carvalho, F., & Pereira, C. E. (2006). Java framework for distributed real-time embedded systems. In *Proceedings of 9th IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing (ISORC)* (pp. 85–92).

Stankovic, J. (1996). Strategic directions in real-time and embedded systems. *ACM Computing Surveys, 28*(4), 751–763.

Synopsys. (2003). *CoCentric System Studio*. http://www.synopsys.com.

Van Brussel, H. (1994). Holonic manufacturing systems, the vision matching the problem. In *Proceedings of the First Conference on Holonic Manufacturing Systems*.

Van Brussel, H., Wyns, J., Valckenaers, P., Bongaerts, L., & Peeters, P. (1998). Reference architecture for holonic manufacturing systems: PROSA. *Computers and Industry, 37*(3), 255–274.

Van Leeuwen, E. H., & Norrie, D. (1997). Intelligent manufacturing: Holons and Holarchies. *Manufacturing Engineering, 76*(2), 86–88.

Vyatkin, V., Christensen, J. H., & Lastra, J. L. M. (2005). OONEIDA: An open, object-oriented knowledge economy for intelligent industrial automation. *IEEE Transactions on Industrial Informatics, 1*(1), 4–17.

Wehrmeister, M. A., Becker, L. B., & Pereira, C. E. (2004). Optimizing real-time embedded systems development using a RTSJ-based API. *Workshop on Java Technologies for Real-Time and Embedded Systems—JTRES 2004, Proceedings Springer LNCS* (pp. 292–297).

**Carlos Eduardo Pereira** received the Dr.-Ing. degree in electrical engineering from the University of Stuttgart, Germany in 1995, the M.Sc. degree in computer science in 1990 and the B.S. degree in electrical engineering in 1987, both from the Federal University of Rio Grande do Sul (UFRGS) in Brazil. He is an associate professor of the Electrical Engineering Department at the Federal University of Rio Grande do Sul in Brazil, where he has served as Dean of the EE Department from 1996 to 1998 and Coordinator for the Graduate Research Program from 2002 to 2006. From 2000 to 2001 he was a visiting researcher at the United Technologies Research Center (UTRC) in Hartford, CT, USA, where he acted as Group Leader of the Embedded Information Devices Group and has coordinated a group of 15 research engineers involved with research projects for United Technologies companies, such as Carrier, Otis, Pratt and Whitney, Sikorsky and UT Fuel Cells. Since 2005 he is acting as technical director for CETA—an Applied Research Center, whose goal is to promote collaborative research work between academia and industry, focusing on the areas of industrial automation, information and communication technologies, and optimization of production processes. Prof. Pereira's research focuses on methodologies and tool support for the development of distributed real-time embedded systems, with special emphasis on industrial automation applications and the use of distributed objects over industrial communication protocols. He has worked on several research projects in collaboration with industry, mostly dealing with the development of real-time computer-based systems. He is Chair of the IFAC Technical Committee on Manufacturing Plant Control (TC 5.1). He is also an Associate Editor of the Journals ''Control Engineering Practice'' – Elsevier and AtP International, Oldenbourg. He has more than 150 technical publications on conferences and journals and has acted as member of International Program Committees for several conferences in the field of industrial automation, manufacturing, industrial protocols, and real-time distributed object computing. He has been the general chair of the 21st IFAC Workshop on Real-Time Programming, WRTP'96, the 5th IFAC Workshop on Intelligent Manufacturing Systems, IMS'98, the 2nd IFAC Workshop on Intelligent Assembly and Disassembly, IAD'01 and the 11th IFAC Symposium on Information Control Problems in Manufacturing, INCOM'04.

**Luigi Carro** was born in Porto Alegre, Brazil, in 1962. He received the electrical engineering and the M.Sc. degrees from Universidade Federal do Rio Grande do Sul (UFRGS), Brazil, in 1985 and 1989, respectively. From 1989 to 1991 he worked at ST-Microelectronics, Agrate, Italy, in the R&D group. In 1996 he received the Ph.D. degree in the area of computer science from Universidade Federal do Rio Grande do Sul (UFRGS), Brazil. Prof. Carro is presently at the Applied Informatics Department at the Informatics Institute of UFRGS, in charge of Computer Architecture and Organization disciplines at the graduate and undergraduate levels. He is also a member of the Graduation Program in Computer Science at UFRGS, where he is responsible courses on embedded systems, digital signal processing, and VLSI design. His primary research interests include embedded systems design, digital signal processing, mixed-signal and analog testing, and rapid system prototyping. He has published more than 120 technical papers on those topics and is the author of the books Digital systems Design and Prototyping (in Portuguese) and Fault-Tolerance Techniques for SRAM-based FPGAs. He has served as Technical Program Committee member of several conferences, like DATE, VTS, ETS, IESS + CODES, FPL, SAMOS and RAW.