The IEEE International Conference on
Industrial Informatics (INDIN 2008)
DCC, Daejeon, Korea July 13-16, 2008

Service-Oriented Control Architecture for Reconfigurable Production Systems

J. Marco Mendes
Faculty of Engineering - University of Porto, Rua Dr. Roberto Frias s/n, 4200-465 Porto, Portugal
marco.mendes@fe.up.pt

Paulo Leitão
Polytechnic Institute of Bragança, Quinta Sta Apolónia, Apartado 134, 5301-857 Bragança, Portugal
pleitao@ipb.pt

Armando W. Colombo
Schneider Electric GmbH, Steinheimer Str. 117, D-63500 Seligenstadt, Germany
armando.colombo@de.schneider-electric.com

Francisco Restivo
Faculty of Engineering - University of Porto, Rua Dr. Roberto Frias s/n, 4200-465 Porto, Portugal
fjr@fe.up.pt

***Abstract*-Evolvable and collaborative production systems are becoming an emergent paradigm towards flexibility and automatic re-configurability. The reconfiguration of those systems requires the existence of distributed and modular control components that interact in order to accomplish control activities. This paper focuses on service-oriented production systems, which behavior is regulated by the coordination of services that are provided and required by control components with different roles. Internally, these components are independent of the implementations, but an internal modular and event based structure is presented. Individual control and interaction is achieved by using embedded or inter-service control processes for which High-Level Petri Nets are proposed. Supporting the predefined control, decision support systems are used to provide conflict resolution and other decision-making functions.**

I. INTRODUCTION

Global competition, mass customization-influenced markets and rapidly changing customer requirements are forcing major changes in the production styles and configuration of manufacturing systems. Increasingly, traditional centralized and sequential manufacturing planning, scheduling, and control mechanisms are being found insufficiently flexible to respond to changing production styles and highly dynamic variations in product requirements [1]. Currently, to stay in business, a manufacturing enterprise should be able to change promptly and dynamically its product catalogue and react quickly to unexpected disturbances [2]. One of the most significant facts in this emergent environment is the need for reconfigurable systems, with re-configurability being a key enabler of competitiveness by providing the way to achieve rapid and adaptive response to change.

The ever growing needs for more flexible and reconfigurable production systems has led to the development and application of new production automation and control paradigms. One promising approach, which has the potential to surmount the technical, organizational and financial limitations inherent to the most current approaches, is to consider the production entities as a conglomerate of distributed, autonomous, intelligent and reusable units, which operate as a set of collaborating entities. From a functional point of view, each collaborative unit can, at each time, initiate collaborative actions and dynamically interact with each other in order to achieve both local and global objectives, when they are considered within a cross-layer infrastructure like a manufacturing enterprise [3]. Based on these principles, different approaches have been developed and analyzed to cover the requirements of reconfigurable production systems, such as Multi-Agent Systems (MAS) [4] and Holonic Manufacturing Systems (HMS) [2, 5].

Indeed, in spite of some agent and holonic approaches' success, a significant incursion in manufacturing plants in use today is still missing [6]. Additionally, some issues remain unanswered, namely those related to the standards of interaction and interoperability, common ways for resource exposition and access, and reconfiguration of the control components and integration with the business level.

The introduction of Service-oriented Architecture (SoA) [7-8] principles in production systems allows the development of distributed, reusable and agile systems, exhibiting more powerful reconfiguration mechanisms. In these systems, the behavior is regulated by the coordination of services, with each participant (such as manufacturing components, intelligent decision mechanisms and control devices) providing services that represent its functionalities, and that will be requested by other components. A major challenge in this type of service-oriented systems is related to how individual entities may interact, coordinating their activities by synchronizing the execution of services they provide. The aggregation of single services and all the interaction patterns between them is also a complex issue. Specifically for service-oriented based collaborative automation systems, there is a gap on mechanisms and engineering tools that provide interaction schemes, protocols and patterns applied for services and corresponding providers and requesters.

This paper addresses the challenge by proposing a modular control architecture for service-oriented production systems addressing the key requirements of flexibility and re-configurability. The control architecture is based on service-oriented principles to achieve distributed, modular, agile and interoperable systems, complemented with High-level Petri nets for the process control and some concepts of multi-agent and decision systems to achieve autonomy and intelligence supporting conflict resolution. The combination of service-oriented and multi-agent systems paradigms allows reconciling the principles of autonomy and interoperability.

978-1-4244-2171-8/08/$25.00 ©2008 IEEE

The paper is organized as follows: first, Section 2 describes the principles of service-oriented architectures and their adoption in automation. Section 3 introduces the foundations of the service-oriented control architecture for reconfigurable production systems, including the identification of control components and the coordination and synthesis of their services. Section 4 describes the modular structure for a generic control component of the proposed architecture. Finally, Section 5 rounds up the paper with conclusions.

II. Service-oriented Production Systems

Service-oriented Architecture (SoA) is the abstract concept of a software architecture, where the focus is the offer, search and use of services over the network [9], providing a communication platform by using open protocols which address the interoperability in heterogeneous systems.

In a service-oriented architecture, services are the primary organizing principle [8]. A service is a software module that encapsulates the control logic or resource functionality of a component that responds to a specific request (Fig. 1). The access to a service is described by its interface and the requester can thus access the service without bothering with the underlining implementation. To discover services, there must be a discovery mechanism to locate them, e.g, service directory facility, broadcasting announcement messages, etc. Services have the following main features [8]: autonomy, loosely-coupled, interoperable, composable and reusable.

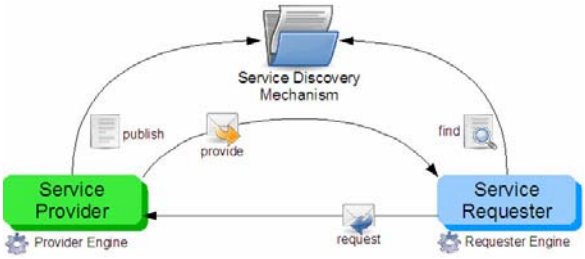

Fig. 1. Illustration of Service-oriented Architecture concepts.

In SOA, a pertinent question is about how to interact with services and how automated processes can use them [10]. Several words are used in the SoA community to describe more complex interactions, such as orchestration, choreography and composition (see [11]). As a new domain of software engineering, services engineering concerns every aspect from development, deployment, use, to evolution of services, such as analysis, architectures, development methodologies, descriptions, testing, development environments, management, and applications [12].

Standard protocols should handle the basis for these issues and thus specify technology rules that should be followed by all involved partners to successfully permit the conversation. From the technology point of view, Web services are the main force to implement these concepts using well established Web protocols, but also other types of implementations are possible. The use of the SoA paradigm implemented through Web services technologies enables the adoption of a unifying technology for all levels of the enterprise, from sensors and actuators to enterprise business processes [10].

In automation domain, the vision of using Service-oriented Architectures is to support the life-cycle needs in the context of agile and flexible manufacturing, addressing distributed, modular and reconfigurable automation systems which behavior is regulated by the coordination of services. The request for easy reconfigurable manufacturing systems composed of standard components that may be remotely supported by geographically distributed engineering partners to suit changing and unpredictable business [13]. The SIRENA Project [6] has contributed to potentate the SoA-based automation by providing Web Services at device level through the extension of the SoA paradigm into the realm of low-level embedded devices, such as sensors and actuators. The feasibility of this approach has been demonstrated through a proof-of-concept implementation based on the Devices Profile for Web Services (DPWS), a device-oriented subset of the Web Services protocols. Since then, significant research is going on, covering the engineering of such systems, including the modeling, semantic description and collaboration.

III. Foundations of a Service-oriented Control Architecture

The demand for production systems that exhibit high degree of re-configurability will obviously impose strong requirements on the way the systems are designed, installed, operated and even re-configured [6]. Being a system which requires diversified functionalities based on its nature and expected behavior, a structure that holds the components and defines the features and interactions has to be defined.

The essence of the proposed control architecture is the convincement that re-configurable production systems can be seen as aggregations of components that are (re)used whenever necessary, providing and requesting services. The proposed architecture is based on the following main foundations:

- Service-oriented systems, in which the access to the resources is via exposable services;
- High-level Petri nets to support the service process description, logical control and analysis;
- Multi-agent and decision support systems, introducing distributed intelligence for collaboration, negotiation and direct interaction over the process control when needed.

The suggested control architecture approach allows achieving powerful reconfigurable and evolvable systems. It is built upon modular and simple control components, illustrated in Fig. 2, each one providing a set of services that represent its internal functionalities. The idea behind the control components is similar to other approaches, such as holons or physical agents: these production entities are parts of a heterarchical organization (i.e. that may behave hierarchical and/or distributed) and contribute to the overall interest by collaborating. This organization is also characterized by its aggregation capabilities, i.e., simpler units might be aggregated in order to generate more complex structures.

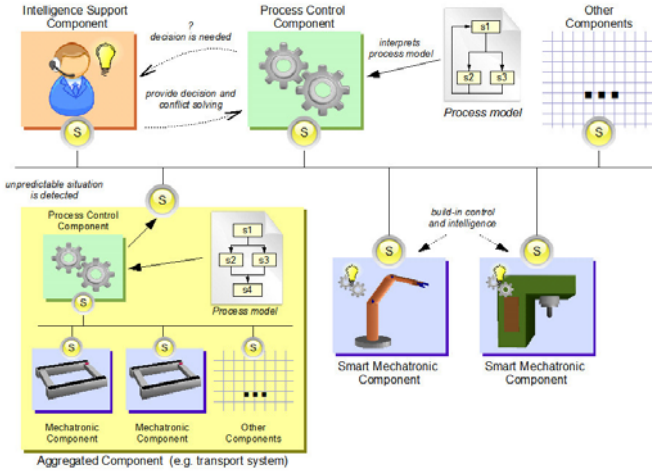

Fig. 2. Main components in a service-oriented control.

The control of such service-oriented production systems is related to the coordination of the distributed components of the system, i.e. by describing and managing processes of services they provide. All the interaction processes between the components are exclusively via the access to services that are connected by communication network.

A. Control Components

The architecture identifies three types of components that participate in the control: Mechatronic Components (MeC), Process Control Components (PCC) and Intelligence Support Components (ISC). They control their own behavior and can be easily combined to form more complex devices to build the desired production system in a "Russian doll" manner [6], providing different levels of granularity: local control exhibited by mechatronic devices, collaboration between components and aggregated control to create higher level services based on individual ones. The automatic re-configuration appears due to the easy re-organization of the components and the services they provide, reflected by the modification of the connections between the devices presented in the system.

The MeC component embraces the mechanical, electronic and software combination, allowing the local control of the physical device. It may have embedded a logic controller (that interprets e.g. Petri net process descriptions) to regulate its local behavior in cases where the decomposition of atomic operations is possible. In this case, it is also necessary to synchronize the services provided by the physical devices, such as reading inputs, writing outputs or invoking programs.

A MeC can also have embedded its own intelligence, being in this case used the abbreviation SMeC (Smart Mechatronic Component). The intelligent mechanisms provide support to the logic controller mechanism in case of decisions or conflict resolution. A SMeC may also represent composed components, in the sense that one component is an aggregation of several other ones (as the transport system illustrated in Fig. 2).

In more complex systems, built upon small mechatronic components, it is necessary to have components that provide global process coordination mechanisms. Process Control Components (PCC), which have their own logic controller, provide this kind of mechanisms, acting as clients that are able to use services provided by other components. The PCC support the complex process flow and interaction of services in the system, according to a process model. For this purpose, they implement the logic for the workflow-oriented execution and sequencing of atomic services, and provide a high-level interface for the aggregated process, made of individual ones. As an example, a robot can provide a transport service that incorporates several operations (such as pick, move and place). It may synchronize its activities with its connected neighbors and, if necessary, take orders from global control components.

Additionally to the static predicted control that mainly executes a specific process plan, the flexibility and indirection can be extended by decision mechanisms. Decision itself can be done locally by the component or using specific decision components for complex global behavior. For this purpose, the architecture considers Intelligence Support Components (ISC), that offer special services, e.g. to choose one of several available alternative actions. The idea is to have ISC components, which incorporate real-time decision capabilities, separating the system control layer from the decision-making layer. Being a dynamic system, sometimes there are unexpected circumstances that a PCC can not handle: operation's delay and canceling, synchronization among individual workflows, unexpected situations, unaccomplished operations, dynamically adding new operations, etc. In these situations, ISC components provide support to solve unexpected situations.

A PCC and an ISC component may be integrated as one component, being complementary to each other. In this paper, they are handled separately to highlight their roles.

B. Service Process Description and Control

In service-oriented production systems, built upon mechatronic components that provide and request distributed services, the desired production process is achieved by putting these control components working together to achieve global production objectives. Besides to exhibit individual pluggable and communication capabilities and to manifest intelligence to support self-organization and adaptation to evolution, a crucial challenge is related to how individual services may interact, coordinating their activities according to an established sequence of services. In fact, coordination and aggregation mechanisms may be of crucial importance to the emergent behaviors of individual control components. These coordination mechanisms must also consider interactions that combine the component level with higher-levels of supervision to achieve cohesive distributed intelligent control.

Fig. 3 illustrates the process description and control of services to address the execution of a model. In this example, different available resources in the system expose their services. The process model, representing the high-level service, is interpreted by the logic controller of the PCC, which is responsible to call the necessary services in the way described by the model.

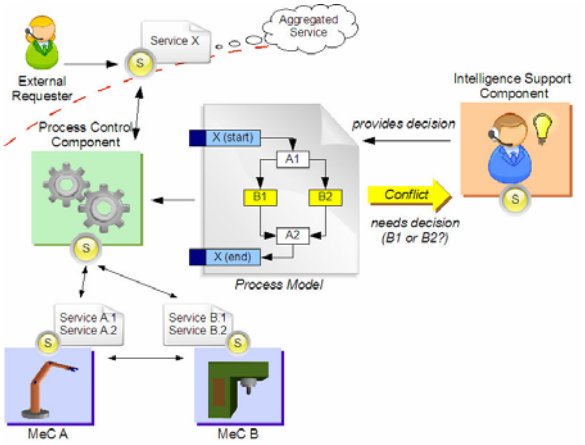

Fig. 3. Description and control in service-oriented production systems.

In this example the control is hierarchical and coordinated by the PCC because there is a requirement to provide an aggregate service. However, control can also be done by collaboration and coordination between MeC, since they may have their own logic controller. The task of the PCC is to follow the schema represented by a workflow model, synchronizing the associated activities (e.g. requesting the described services). In the model is also included a conflict that requires a decision to define which path of the process should be followed. In this case, an ISC can handle the special indirection over the control, by providing the necessary decision based on different parameters (e. g. product information, actual system status, etc.).

The process model in Fig. 3 is based on a generic workflow diagram, but - as indicated previously - a solution proposed by the authors is to use High Level Petri Nets. More details follow in section IV.

C. *Connection and Aggregation of Components and Services*

The control architecture proposes a Lego™-based approach: grouping elementary and interconnectable components in a particular way it is possible to build bigger and more complex systems. Thus, the development of modular, re-configurable and complex production systems requires the aggregation of control components in a dynamic and automatic way.

The connection between mechatronic devices is a way to achieve the union of components. Connections are established via the ports of the control models, that can be directly done when models are in the same component (e.g. running on a PCC) or using the service ports for distributed components (e.g. between MeC). The question is if it is simply a matter of overlapping port's logic or if it requires more complex "connection logics". The connection between control components must comprehend several rules. In a first balance, each model should have a compatible interface to be connected to; not only a set of matching operations but also a complementary functionality. After that, a connection can only be established after a successfully agreement between the involved partners. In a more complex scenario it may involve some kind of negotiation. The connection itself needs a specific "glue" that corresponds to some logic restrictions, which are dependent on the type of the connection. This can be seen as a protocol for the communication that, beside others, establishes a message synchronization pattern for the interaction. The direct connection of modules inside the same component is more restricted in terms of requirements, but that can be handled in a similar way to its distributed counterpart.

Connections, aggregations and general service usage can be quite complex when the procedure should occur automatically. Semantically-rich descriptions combined with machine reasoning allow the automatic matchmaking of required and offered services using logical inference, rather than the hard-coded one-to-one mappings [14]. This type of matchmaking enables utilizing services that did not exist or were not known when the requester side was programmed. Automatic matchmaking enables the reconfiguration of the system by dynamically selecting services without reprogramming the integration logic.

IV. Modular Structure of Control Components

Each control component of the proposed architecture may be implemented independently and differently. The only requirement is that it should share its functions as services and obey to the protocols of communication and processes. This section proposes a modular and event-driven design methodology for the control components.

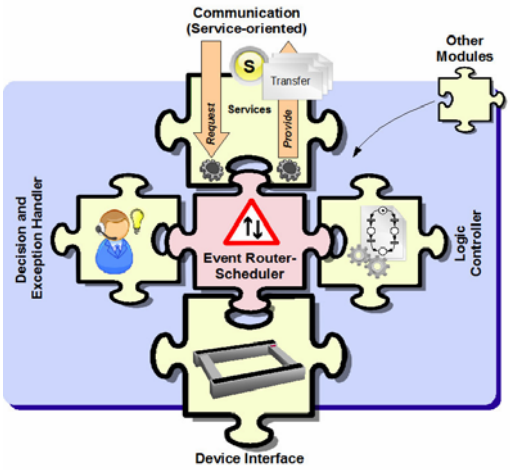

Fig. 4. Modular conceptual structure of a control component.

A general control component is structured in a puzzle of adaptable and reusable modules, as illustrated in Fig. 4. The main modules considered here are: Logical Control, Decision and Exception Handler, Communication, Device Interface and Event Router-Scheduler. These modules are included in the control component according to its needs and possibly implemented using different technologies depending on the host platform. The proposed modules are not strictly necessary to deploy the components (except Event Router-Scheduler as a backbone for event handling and adaptation of other modules), thus it is up to the developer to setup the structure according to the components objectives.

A. *Logic Controller Module*

The Logic Controller module regulates the behavior of the component (i.e. the physical device if embedded in a MeC or a larger system if embedded in a PCC component) by coordinating a set of services described by a logical model. The Logic Controller module may be integrated in more than one component, building a non-hierarchical architecture, which means that the supervisory control is really distributed and not centrally coordinated. In this case, the processes encapsulated by the coordination services have to collaborate with each other in order to reach the same level of synchronization as a centralized PCC would provide.

The logic embedded in the control component is dependent on the system topology, but should be independent from the process plan of each product. The proposed approach to develop logic controllers (i.e. design, validation, simulation and execution) is to use a kind of High-level Petri nets tailored for service-oriented systems [15], taking the advantage of their powerful mathematical foundation to represent discrete, dynamic, and distributed systems. High-level Petri nets are particularly well-suited for systems in which concurrency and parallelism, synchronization, resource sharing and mutual exclusion are important (see [16]).

Logic controllers have to interpret and execute the process model expressed in HLPN. In real-time execution, enabled transition must be detected, services associated with the transition must be called and, after that, the process model has to be updated to reflect the actual state of the system. Logic controllers synchronize and control the whole process until it reaches the goal, based on the elaborated model.

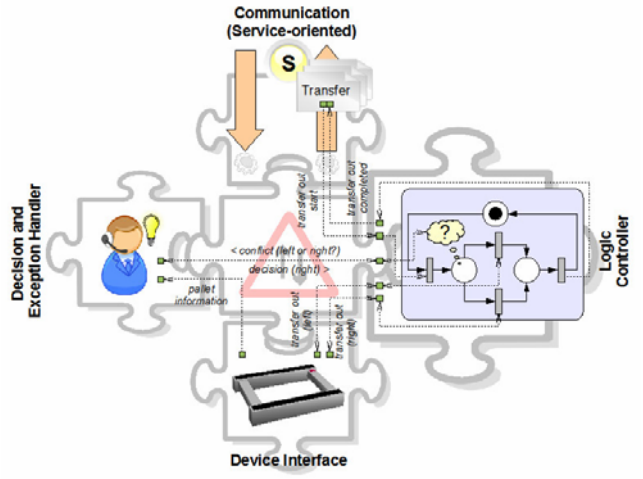

Fig. 5. High-level Petri net control and event based interaction inside the control component.

The logic control model is represented by a Petri net model according to the behavior of the process. The represented transitions are used to model time-consuming activities, such as the execution of an operation. These transitions can also be exploded (i.e. detailed) allowing the step wise refinement of the model, such as the *transfer out* (right/left) operation that is actually a sequence of different steps. Fig. 5 illustrates the coordination of a process model that presents two alternative outputs for the transfer (right and left). The execution of the process model is behind the transfer service that is coordinated by the transfer in start/completed operations.

The designed Petri nets models including the information about the system operation (e.g. process plans, resources, layout and control laws) will constitute a computational model practical for analytical validation, and a simulation model, which allows experiments to be performed in the system model. This model can be easily analyzed and validated in the design phase, and proceeding into the implementation phase only after the verification of the correctness of system design.

B. *Communication Module*

The communication among the control components is done via the invocation of component's services hosted and provided by the communication module. As an example, a conveyor may provide the Transfer service to handle the movement of pallets. This service may be used by other components, but it can also call external services when needed (e.g. to be connected to other conveyor it needs to request the Transfer service of another conveyor).

A suitable technological solution to implement the service-oriented communication module is to use Web technology, and most specifically Web services. At its core, Web services technology is quite simple and it is designed to move XML (eXtended Markup Language) documents between service processes using standard Internet protocols. This simplicity helps Web services to achieve the primary goal of interoperability and also means that it is necessary to add other technologies to build complex distributed applications. A profile has been specified for adopting Web services at the device level known as Device Profile for Web Services (DPWS) [9].

C. *Decision and Exception Handler Module*

The coordination of the services' execution, depending on the flexibility that the system reveals, requires the decision-making and conflict resolution at runtime, because a system logic model does not describe a fixed sequence of actions, but rather all possible combinations thereof. For instance, if a system is flexible through redundancy of service providers, the concrete mapping of services to devices has to be decided at runtime. The existence of conflicts does not strictly mean that there are design problems in the system, but should be also understood as an opportunity of applying decision to a more flexible system [1]. Other aspects are the handling of undocumented situations by the process model, generating a exception to the normal control.

Considering the Fig. 5, the model presents a decision point (marked with a '?') to allow different transfer outputs (in this case left and right). This decision is translated into the Petri net model as a conflict and requires that someone resolves the conflict according to a specific criterion. In the example, the information of the pallet that is to be transferred (e.g its

identification and process description) is considered by the Decision and Exception Handler to provide a specific direction (left or right) to the control and thus, activating one of the corresponding transition.

The degree of complexity associated to the decision-making mechanism can range from simple algorithms to complex computational systems, such as multi-agent systems, neural networks and genetic algorithms. Agents are technologically suitable to provide autonomy and intelligence, supporting indirection, control decision and conflict resolution. Their support can be extended into the reconfiguration of the system by providing new generated possibilities to the behavioral model through the use of learning mechanisms.

D. Device Interface Module

The Device Interface provides the mechanisms to integrate the physical device, such as robots or sensors, within the control of the MeC. As local device controllers usually have closed architectures it is necessary to develop wrappers to hide the details of each device controller and to supply primitives that represent the functionality of the physical devices. The Device Interface module aims to fulfill this objective, by providing mechanisms to support the easy and transparent integration of physical devices within control applications.

E. Event Router-Scheduler Module

Being the control component constituted from several modules that may operate asynchronously and using different processes and/or threads, a special module is required to connect all modules together and protect data concurrency. For this purpose, the heart of the puzzle is the Event Router-Scheduler module that provides the following main features:

- Internal event-based mechanisms to pass events from one module to another;
- Prioritized buffers for events in case of high event traffic;
- Thread management and data protection facilities;
- General interface to plug-in different type of modules.

As an example, the request for a *transfer out start* operation received by the Communication module is translated into an internal event and routed by the Event Router-Scheduler module to the Logic Controller module. After processing the logic, a new event is sent to the Device Interface module that may put the conveyor's motor output signal ON.

V. CONCLUSION

This paper introduces a service-oriented production control architecture addressing the requirements of flexibility and re-configurability. The present work suggests reconfigurable production systems build upon the concept of distributed control components that combines the features of Service-oriented Architectures directed to automation and production systems. On the other hand, the flexible architecture permits the application of different control strategies. As process control methodology, HLPN was introduced to design, validate, simulate and execute processes.

Several key points were achieved, namely a stable basis describing the environment and role of different components in the control architecture. Other details were explained about the modular structure of control components, based on different functionality modules that are tied together by the central Event Router-Scheduler Module.

As future work, the further specification of behavioral, process control and decision mechanisms is required, as well as their application to case studies.

ACKNOWLEDGMENT

The authors would like to thank the partners of the Innovative Production Machines and Systems (I*PROMS) Network of Excellence (http://www.iproms.org) and the SOCRADES project (http://www.socrades.eu), for their support.

REFERENCES

[1] J. Mendes, P. Leitão, F. Restivo, A.W. Colombo and A. Bepperling, "Engineering of Service-oriented Automation Systems: a Survey", I*PROMS Conference on Innovative Production Machines and Systems, 2007.

[2] A.W. Colombo and F. Jammes, "Collaborative Automation and Service-oriented Architectures in the Industry", Tutorial at the IEEE International Conference on Industrial Electronics, France, 2006.

[3] M. Wooldridge, "An Introduction to Multi-Agent Systems", John Wiley & Sons, 2002.

[4] S. Deen. "Agent-based Manufacturing: Advances in the Holonic Approach", Springer Verlag Berlin Heidelberg, 2003.

[5] P. Leitão and F. Restivo, "ADACOR: A Holonic Architecture for Agile and Adaptive Manufacturing Control", Computers in Industry, 57(2), 2006, pp. 121-130.

[6] F. Jammes and H. Smit. Service-oriented Paradigms in Industrial Automation. IEEE Transactions on Industrial Informatics, 1(1), 2005, pp. 62–70.

[7] E. Marks and M. Bell, "Service-oriented Architecture: A Planning and Implementation Guide for Business and Technology", John Wiley & Sons, 2006.

[8] I. Melzer et al., "Service-orientierte Architecturen mit Web Services", 2. Aufl., Elsevier, Spektrum Akademischer Verlag.

[9] F. Jammes, H. Smit, J. Lastra and I. Delamer, "Orchestration of Service-Oriented Manufacturing Processes", Proceedings of the 10th IEEE International Conference on Emerging Technologies and Factory Automation, Vol. 1, 2005, pp. 617-624.

[10] A. Bepperling, J. Mendes, A.W. Colombo, R. Schoop and A. Aspragathos, "A Framework for Development and Implementation of Web Service-based Intelligent Autonomous Mechatronics Components", Proceedings of the IEEE International Conference on Industrial Informatics, 2006, pp. 341-347.

[11] C. Peltz, "Web Services Orchestration", Hewlett Packard, Co., 2003.

[12] M. Aoyama, S. Weerawarana, H. Maruyama, C. Szyperski, K. Sullivan, and D. Lea, "Web services engineering: promises and challenges", Proceedings of the 24th International Conference on Software Engineering, ACM Press, 2002, pp. 647-648.

[13] A. Colombo, F. Jammes, H. Smit, R. Harrison, J. Lastra and I. Delamer, "Service-oriented architectures for collaborative automation", 32nd Annual Conference of IEEE Industrial Electronics Society, 2005, 6pp.

[14] I. Delamer and J. Lastra, "Ontology Modeling of Assembly Processes and Systems using Semantic Web Services", Proceedings of the IEEE International Conference on Industrial Informatics, 2006, pp. 611–617.

[15] P. Leitão and A.W. Colombo, "Petri net based Methodology for the Development of Collaborative Production Systems", Proceedings of the 11th IEEE International Conference on Emerging Technologies and Factory Automation, 2006, pp. 819-826.

[16] A. W. Colombo, "Development and implementation of hierarchical control structures of flexible production systems using high-level Petri nets", PhD Thesis, Fertigungstechnik, Erlangen, Nuernberg, 1998.