RECONFIGURABLE PRODUCTION CONTROL SYSTEMS: BEYOND ADACOR

Paulo Leitão [1], João Mendes[2], Armando W. Colombo[3], Francisco Restivo[2]

[1] *Polytechnic Institute of Bragança, Quinta Sta Apolónia, Apartado 1134, 5301-857 Bragança, Portugal, pleitao@ipb.pt*

[2] *Faculty of Engineering of University of Porto, Rua Dr. Roberto Frias, 4200-465 Porto, Portugal, {joao.mendes,fjr}@fe.up.pt*

[3] *Schneider Electric - HUB & Globalization of Technology, Steinheimer Str. 117, 63500 Seligenstadt, Germany, armando.colombo@ de.schneider-electric.com*

Abstract: In the recent evolution of production control systems, the emergence of decentralized systems capable of dealing with the rapid changes in the production environment better than the traditional centralized architectures has been one of the most significant developments. The agent-based and holonic paradigms symbolize this approach, and ADACOR holonic control architecture is a successful example of such a system. In this paper, authors discusses the current challenges and the way to go in the direction of new, reconfigurable, evolvable and ubiquitous systems, able to respond to current production environment demands and variability.

1. INTRODUCTION

Production control systems have evolved dramatically during the last few years. One of the most significant facts is the emergence of decentralized systems capable of dealing with the rapid changes in the production environment better than the traditional centralized architectures. The quest for agility and re-configurability requires a new class of production control systems, characterized by:

- A community of distributed and intelligent building blocks, designated by control units.
- Each control unit is autonomous, having its own objectives, knowledge and skills, and encapsulating intelligent functions; however, none of them has a global view of the system.
- Global decisions (e.g. scheduling and diagnosis) are obtained by more than one control unit, i.e. control units need to work together to reach a production decision.
- Control units should exhibit some emergent behavior, such as to adapt to changes without external intervention.
- Control units representing mechatronic devices, such as sensors or robots, are part of the production control system architecture.

The agent-based and holonic paradigms symbolize this new approach, and ADACOR (ADAptive holonic COntrol aRchitecture for distributed manufacturing systems) [1], as others (see e.g. PROSA [2], HCBA [3] and Bussmann [4]), is a successful example of such a system. ADACOR deals with the re-configurability in manufacturing systems by introducing an adaptive production control system that evolves dynamically between a more hierarchical and a more heterarchical control architecture, based in self-organization and learning capabilities embedded in individual holons.

Re-configurability, that is the ability of the system to dynamically change its configuration, usually to respond to dynamic changes in its environment, e.g. a new production scenario, assumes a key role in the new generation of production control systems, providing the way to achieve a rapid and adaptive response to change, which is a key enabler of competitiveness.

Starting with an overview of how re-configurability is supported by ADACOR architecture, this paper presents the current challenges and the way to go in the direction of new, reconfigurable and ubiquitous systems, able to integrate networked production resources to respond to the variability of production scenarios beyond those that were envisaged at design time. For this purpose, the paper introduces the guidelines for the new generation of re-configurable production systems and discusses how to implement these systems, pointing out the benefits of combining multi-agent systems with service-oriented architectures.

2. RE-CONFIGURABILITY IN ADACOR ARCHITECTURE

ADACOR is built upon a society of autonomous and cooperative holons, each one representing a manufacturing component. ADACOR defines four holon classes [1]: product (PH), task (TH), operational (OH) and supervisor (SH). The product holons represent the available products in the factory catalogue, the task holons represent the production orders launched to execute the requested products and the operational holons represent the physical resources available at shop floor. Supervisor holons bring hierarchy to these otherwise decentralized systems by providing co-ordination to the holons under their supervision.

ADACOR addresses the reaction of the control system to unexpected disturbances, specifically those that have impact at planning and scheduling level, such as machine failures and rush orders. In these situations, local information is more important than global information to assure a fast reaction to the problem, and decentralized architectures behave better than centralized ones in terms of short term production loss. ADACOR solves this problem through a simple reconfiguration mechanism that balances between a more hierarchical and a more flat approach, combining the global production optimization with the agile reaction to unpredictable disturbances [1]. In other words, the designed adaptive mechanism intends to be as centralized as possible and as decentralized as necessary, i.e. using a hierarchical approach when the objective is the optimization, and a more heterarchical approach in presence of unexpected scenarios.

The autonomy, self-organization and learning capabilities exhibited by ADACOR holons, and the role of supervisor holons are the keys to achieve this adaptive production control approach, which supports the dynamic evolution and re-configuration of the organizational control structure [1]. Briefly, in normal operation, the holons are organized in a hierarchical structure, with the presence of supervisor holons, acting as coordinating entities. Supervisor holons elaborate, periodically, optimized schedules that are proposed to the operational holons, which follow the suggestions issued by the central entity.

When an unexpected disturbance is detected, the system is forced to evolve to a heterarchical structure, operating without the presence of coordination levels, with each holon assuming the control of its own activity. This re-organization is supported by the self-organization capability of each holon, mapped with the increase of its level of autonomy and the propagation of the disturbance to the neighbor holons using ant-based techniques [5]. In this turbulent stage, the manufacturing re-scheduling is achieved in a distributed manner, resulting from the direct interaction between task and operational holons.

After the disturbance recovery, the system evolves to a new control structure. Learning mechanisms embedded in each holon are responsible for the distinction between abnormal situations and normal evolution of the system environment.

3. TOWARDS NEW RECONFIGURABLE PRODUCTION SYSTEMS

The demand for intelligent and distributed control systems that exhibit high degree of re-configurability will obviously impose strong requirements on the way the systems are designed, installed and operated. Current approaches being applied within the reconfiguration domain will not suffice. Indeed, in spite of the success of some agent-based and holonic approaches, of which ADACOR is one example, a significant incursion in manufacturing plants in use today is still missing. The reasons for this situation are many, from the lack of answer to several basic questions in terms of development and performance of these systems (e.g. distributed thinking, interoperability, re-configurability, robustness and scalability) to the unavailability of methodologies for manufacturing control software development and verification, before the final implementation, including methods for effective reuse and/or reconfiguration of control solutions.

If there is still a long way to go in the direction of new, reconfigurable and ubiquitous systems, able to integrate networked production resources to respond to the variability of production scenarios beyond those that were envisaged at design time, for sure that self-organization and emergent behavior will be key issues to support the new generation of reconfigurable production control systems. Two types of architectures, with different granularity, must be considered:

- The *individual architecture* of each autonomous entity, which must exhibit intelligence, learning, self-organization and pluggable capabilities.
- The *overall manufacturing control system architecture* emerged from the interaction among these autonomous entities, which must be able to support efficiently the adaptation to the new unexpected scenarios and to respond to new business opportunities.

This dynamic and evolvable reconfiguration is one step ahead of traditional re-configurability, considering also the evolution of the system and its components during its life-cycle, e.g. by offering new services or learning to differentiate normal from abnormal situations.

3.1 Building Smart Control Components

Intelligent and distributed manufacturing control systems can be seen as compositions of smart manufacturing components that can be reused whenever necessary. The composition of components is achieved by introducing electronics, intelligence and communication in each mechanic device (e.g. a sensor, a gripper, an actuator or a PLC), leading to the concept of smart control component, Fig. 1.

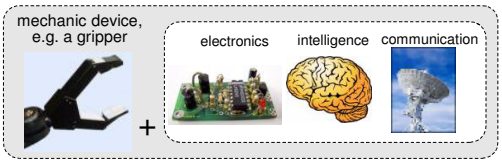

Fig. 1 – Smart Control Component

The smart control components have embedded control and intelligence in processing units (for example microcontrollers) with possibly more computational power than personal computers had some years ago. Each smart control component must encapsulate functions and services that the physical device can perform, e.g. open or close the gripper. These services, that can be modified, added or removed (e.g. a new piece can be handled by a robot after the aggregation of a new gripper), are then exposed to be invoked by other smart control components that want to use them.

Self-organization (i.e. the capability to dynamically re-organize itself in presence of disturbances) and learning (i.e. the capability to acquire new knowledge supporting the dynamic behavior evolution) mechanisms should be considered to provide each component with capability to dynamically evolve during its life-cycle. Additionally, re-configurability is only completely achieved if smart control components can enter or exit the system on the fly, i.e. without the need to re-initialize and re-program the other control components of the system.

The integration of mechatronic devices in such smart control components still presents some problems, mainly due to the heterogeneity of these devices. In fact, the majority of agent-based laboratorial control applications use software agents without the need to integrate physical devices (for example in the supply chain case) or emulators when they are needed (for example, in manufacturing control systems). But in the real situations, industrial applications require the integration of physical mechatronic devices, normally tens or hundreds. Methodologies to support an easy, fast, transparent and re-usable integration of mechatronic devices is then required.

The design of a system based in distributed smart control components raise a pertinent question related to the granularity of the system. In fact, fine-granularity, considering simple local units, implies a high complexity of the system due to the high degree of interactions involved. On the other hand, coarse-granularity, considering large units, implies a lower complexity of the system but a higher difficulty to evolve and to adapt to change. The best solution depends on the envisaged scenarios, the short or long term perspectives, and may require a careful balance of the weak and strong points of each view.

3.2 Smart Control Components Working Together

The smart control components, as parts of a complex and distributed real-time system, are distributed autonomous entities which only have local knowledge and act to fulfill their own goals. The desired production process is achieved by putting these smart control components working together to achieve global production objectives.

Orchestration mechanisms may be of crucial importance to coordinate the complex and emergent behaviors of individual smart control components. These coordination mechanisms, that include orchestration engines for service composition, coordination and collaboration, must also consider interaction mechanisms that combine the component level with higher-levels of supervision to achieve cohesive distributed intelligent control.

In distributed manufacturing environments it is important to guarantee the interoperability between the distributed entities or applications and to verify that the semantic content is preserved during the exchange of messages between them. In fact, a study commissioned by NIST (National Institute of Standards and Technology) reported that the U.S. automotive sector alone expends one billion dollars per year to solve interoperability problems [6]. The solution to those problems requires the use of standard platforms that support transparent communication between distributed smart control components or applications. Ontologies play a decisive role to support interoperability problems. Their ultimate goal is the description, possibly without ambiguity, of a certain “reality” of interest, in this case related to production systems. The development of an ontology may take from a few hours up to months or even years depending on the choice of the language, the covered topics, and the level of formality and precision [7].

3.3 Dynamic and Evolvable Reconfiguration

The needs for re-configuration can appear in several manufacturing situations, from the virtual organization level, where reconfiguration of partners is frequent, to the shop floor level, for example in the re-configuration of the part transportation flow, passing by the machine level, for example by changing the set of grippers aggregated to a robot.

The application of multi-agent systems and holonic architectures by themselves does not solve the current manufacturing problems, being necessary to

combine them with mechanisms to support the dynamic structure re-configuration, thus dealing more effectively with unexpected disturbances and minimizing their effects. In other words, questions like how the global production optimization is achieved in decentralized systems, how temporary hierarchies are dynamically formed, evolved and removed, how individual smart components self-organize and evolve to support evolution and emergency, and how to adapt their emergent behavior using learning algorithms, are yet far from being answered.

The achievement of dynamic evolution and reconfiguration requires the introduction of self-organization mechanisms associated to emergent behaviors that support the evolution and reconfiguration of the system based in the self-organization of each individual smart control component. Ideally, re-configuration should appear to users like "drag-and-drop" applications where complexity and details are handled by background services. The reconfiguration of any smart control component should be done on the fly, maintaining unchanged the behavior of the entire system which should continue to run smoothly after the change.

Another requirement to support the dynamic evolution of the system is the introduction of learning mechanisms, allowing the evolution of the functionalities and behavior of individual smart control components and consequently the evolution of entire system. Indeed, learning mechanisms strongly influence the performance of the self-organization mechanism, being critical to support the identification of re-configuration opportunities.

Local and global mechanisms are required to identify the reconfiguration and evolution opportunities, while maintaining system behavior predictable and stable. During the reconfiguration process, some instability can appear as the result of not properly synchronized evolution processes. This implies the need to build up the reconfigurable production system from simple to self-organized and emergent reconfiguration.

4. A NEW PERSPECTIVE TO IMPLEMENT RE-CONFIGURABLE PRODUCTION SYSTEMS

Multi-agent systems [8] is a suitable approach to develop the new class of reconfigurable production systems since they already support the idea of interaction within a society of individual agents, fitting well with the idea of a community of smart control components. Additionally, emergence can be mapped to the evolution of the society of agents when identifying re-configuration opportunities and defining new complex functionalities and behavior. The great advantage of this approach is the possibility to create complex functionalities and behaviors based on the interaction among distributed smart control components.

However, some unanswered problems in multi-agent systems, namely those related to interoperability, how knowledge is shared during the interaction processes and reconfiguration of the control components, by removing, adding or modifying the services they provide, still constitute a barrier to the easy development of such kind of systems.

A current challenge in production control, in the opinion of the authors, is to approach multi-agent systems with new emergent technologies, such as Service-Oriented Architectures (SOA). The best features of these technologies will hopefully support the development of more powerful re-configuration mechanisms, using more complex self-organization and learning techniques. In fact, web services technologies have potential to solve problems concerning heterogeneity of knowledge representation between distributed agents; on the other hand, agent technologies brings solutions in issues that are now becoming important in SOA, such as coordination, negotiation and agreement [11].

In this chapter, the basic concepts about SOAs will be reviewed and the benefits of combining these concepts to support re-configurability and interoperability will be discussed.

4.1 Basic Concepts about SOA

A SOA [9-10] is a middleware that faces the problems of interoperability in autonomous, heterogeneous and distributed systems in the form of service requester and service provider mechanisms, see Fig. 2. A provider hides its internal structure and shows only the necessary functionalities to the outside world, in the form of services. The list of provided services must be published, so they can be discovered by the service requester. A service discovery facility acts like a directory in which services can be added, removed and located.

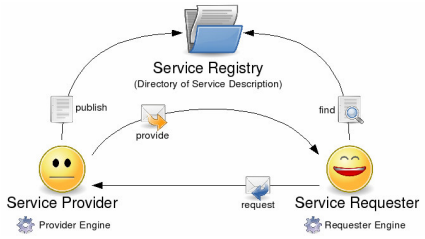

Fig. 2 – Illustration of SOA's Concepts

Commonly SOAs are based on web services (WS), using standard and open protocols to provide a communication platform between distributed and heterogeneous systems and applications. Most of the web service platforms are made of SOAP (Simple Object Access Protocol), WSDL (Web Services

Description Language) and UDDI (Universal Description, Discovery and Integration) that use the combination of HTTP (Hypertext Transfer Protocol) and XML (Extensible Markup Language) as the basic foundation of WS (Fig. 3).

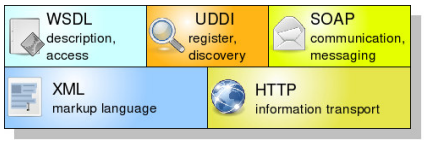

Fig. 3 – Web Service's Building Blocks

One of the challenges of SOA is to reconcile the opposing principles of autonomy and interoperability [12]. Autonomous units have an independent condition, each of them regarding its own structure and conditions. By bridging them together, there is a path that allows the interoperability in a mutual service offer and request.

4.2 Service Orchestration and Choreography

In SOAs, a pertinent question is about how services interact. Service composition [13] is the combination of single services and all the interaction patterns between them. Commonly, terms as service orchestration and choreography are used for this purpose. Orchestration is the practice of sequencing and synchronizing the execution of services, which encapsulate business or manufacturing processes [12-13]. An orchestration engine implements the logic for workflow-oriented execution and sequencing of atomic services, and provides a high-level interface for the composed process. Service choreography is a complementary concept, which considers the rules that define the messages and interaction sequences that must occur to execute a given process through a particular service interface. Additionally, choreography can be used independently in a collaborative system without a centralized approach. Fig. 4 illustrates the service orchestration and choreography concepts.

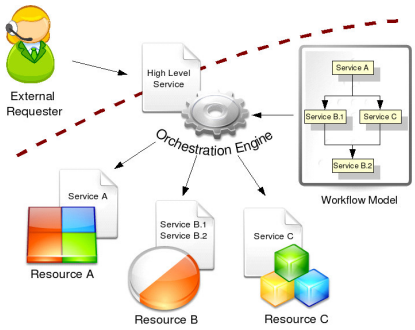

Fig. 4 – Example of Service Orchestration and Choreography

In this example, different available resources in the system expose their services. The workflow model, representing the high-level service, is interpreted by the orchestration engine, which is responsible to call the necessary services in the way described by the model. The choreographed sequences are used for the interaction between the elements and therefore the correct execution of individual services when required. When the workflow model reaches the final stage, the high level service is concluded and the external requester notified.

The Petri nets formalism is a mathematical and graphical oriented language for the design, specification, simulation and validation of systems, particularly those in which concurrency and parallelism, synchronization, resource sharing and mutual exclusion are important. Petri nets can be used to design and validate the process model, i.e. the workflow model, which translates the system goal. Orchestration engines have to interpret the workflow model expressed in Petri nets and execute it. In real-time execution, the enabled transition must be detected, services associated with the enabled transition must be called and, after that, the workflow model has to be updated to reflect the actual state of the system. Orchestration engines synchronize and control the whole process until it reaches the goal, based on the elaborated model, i.e. they have to orchestrate the production system.

As shown in the Fig. 5, the workflow modeled using Petri nets has two parallel processes, e.g. two different products are produced independently, but for reaching the goal, the two products must be concluded successfully. The production is behind the two services (one for each type of product) that only exposes the required operations to the outside. The transition *produce1()* should be invoked to begin the production of product #1 and the transition *finish1()* doesn't fire until that production is concluded.

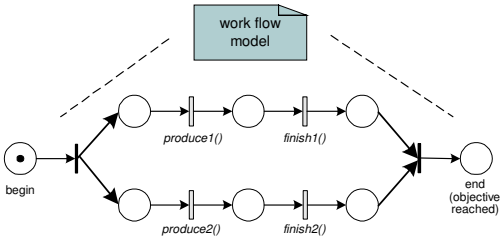

Fig. 5 – Service Orchestration by using the Petri Nets Formalism

The same is also valid for the production of the product #2, but asynchronously from the product #1. The process is terminated only if the last place (*end*) is marked with a token, i.e. after firing the transitions *finish1()* and *finish2()*.

4.3 Multi-agent Systems using Web Services

The integration of web services with software agents will bring benefits from both technologies. From the perspective of agents, web services are a way to

communicate and interoperate using web-specific protocols. To web services, agents can form a powerful means of indirection and control decision by masking the web service. With these aspects in mind, the control architecture may be improved in terms of software flexibility and self-configuration.

The adoption of web services by multi-agent systems satisfies the following requirements:

- Resources (e.g. physical devices, software modules, intelligent units, sub-systems) can be encapsulated with a service provider that acts like a bridge between the internal structure and the exposed interface to the outside world. The access to this kind of resource is via the invocation of the services described by its interface.
- Some services can be composed by other services, creating a leveled structure of services.
- Interoperability in a heterogeneous environment can be addressed by using common communication semantics based on the use of open protocols, namely web technologies, e.g. web services. The distributed nature of the architectures suggests the definition of interoperability functionalities based on service-oriented architecture and the realization of efficient, flexible and robust overall plant control.
- Fault-tolerant systems can consider anomalies that may occur during the production process, identifying in advance possible future occurrence of disturbances.

The re-configurability and evolution of the system is facilitated using multi-agent systems supported by web services technology since it is possible to add, remove and modify dynamically resources and services without interrupting the processes. This allows changing on fly the agents in the system, the services provided by each one of them and the way they are organized.

5. CONCLUSIONS

The objective of this paper was to discuss the main guidelines of the next generation of reconfigurable production systems, based in the experience gained with the ADACOR holonic control architecture, which in its time proved to be an innovative and successful approach to address the re-configurability at shop floor level.

Reconfigurable production systems will be built upon the concept of smart control components, whose individual self-organization and emergent behavior will contribute for the re-configurability and evolution of entire production system. For this purpose, concepts like self-organization, learning and emergent theory should be considered. Multi-agent systems technology appears to be a good solution to develop these systems, which supported by web service technology may deal completely with interoperability and re-configurability needs.

The development of orchestration and choreography mechanisms and tools, including orchestration engines, for service composition, coordination and collaboration, will play a crucial role to support intelligent, re-configurable and modular production control systems.

An important aspect to be taken in the future for the success of this new generation of production control systems, is to proof its real applicability and merits, since industry has afraid to introduce, in its production processes, emergent technologies that are not yet proved, and particularly from the usage of emergent terms like ontologies, self-organization, emergence, distributed thinking and learning. The solution is to demonstrate that reconfigurable production systems based in those concepts, work better than the existing ones.

REFERENCES

[1] P. Leitão and F. Restivo, "ADACOR: A Holonic Architecture for Agile and Adaptive Manufacturing Control", Computers in Industry, vol. 57, nº 2, 2006, pp. 121-130.

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

[3] J.-L. Chirn and D. McFarlane, "A Holonic Component-Based Approach to Reconfigurable Manufacturing Control Architecture", Proc. of the International Workshop on HoloMAS, 2000, pp. 219-223.

[4] S. Bussmann and K. Schild, "An Agent-based Approach to the Control of Flexible Production Systems", Proc. of 8th IEEE International Conference on Emerging Technologies and Factory Automation, vol. 2, 2000, pp. 481-488.

[5] P. Leitão, A. W. Colombo and F. Restivo, "ADACOR, A Collaborative Production Automation and Control Architecture", IEEE Intelligent Systems, vol. 20, nº 1, 2005, pp. 58-66.

[6] S. B. Brunnermeier and S. A. Martin, "Interoperability Cost Analysis of the U.S. Automotive Supply Chain" (Planning Report #99-1), Technical report, NIST, 1999.

[7] S. Borgo and P. Leitão, "Foundations for a Core Ontology of Manufacturing", in "Ontologies: A Handbook of Principles, Concepts and Applications in Information Systems", Springer, 2006, pp. 751-776.

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

[9] F. Jammes and H. Smit, "Service-oriented Architectures for Devices: the SIRENA View", Proceedings of the 3rd IEEE International Conference on Industrial Informatics, 2005, pp. 140-147.

[10] J. Lastra and I. Delamer, "Semantic Web Services in Factory Automation: Fundamental Insights and Research Roadmap", IEEE Transactions on Industrial Informatics, 2 (1), 2006, pp. 1-11.

[11] T. Payne, "AgentLink News Editorial", AgentLink Issue 17, April 2005.

[12] F. Jammes, H. Smit, J. L. Martinez Lastra and I. Delamer, "Orchestration of Service-Oriented Manufacturing Processes", Proceedings of the 10th IEEE International Conference ETFA, Vol. 1, 2005, pp. 617-624.

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