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

Service-oriented Process Control using High-Level Petri Nets

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*- Service-oriented systems constitute a suitable approach for the development of modular, flexible and reconfigurable production systems, addressing the current requirements imposed by global markets. This paper focuses on the process control of Service-oriented production systems, whose behavior is regulated by the coordination of services that are available from distributed modular and collaborative control components. The proposed control is based on a kind of High-level Petri Nets tailored for the description, connection and synchronization of different concurrent processes. Several features can be distinguished by this approach, since its modularity, control reusability, flexibility and other valuable inherited characteristics from the proposed High-Level Petri Nets.**

I. INTRODUCTION

Traditionally, the control in automation systems is done in a centralized/hierarchical manner, e.g. using a PLC (Programmable Logic Controller) that communicates and synchronizes the operation of individual devices, providing limited reconfiguration capabilities. However, production systems should be modular and easily reconfigurable to address current requirements imposed by global markets and customers demands. Additionally, reconfigurable production systems, instead of incorporating all the flexibility once at the beginning of their life cycle, should incorporate basic process models - both hardware and software - that can be rearranged or replaced quickly and reliably [1].

New revolutionary manufacturing concepts and emerging technologies, which take advantage of the newest mechatronics, information and communication technologies are being researched to increase the flexibility and re-configurability of automation systems. Service-oriented Architectures (SoA) is an example of these emergent paradigms that is suitable to address this challenge. Since industrial automation and production systems domains present distinct technical requirements from the originally application in business levels, SoA must be proved not only at its most basic form (see the SIRENA project [2]), but also to permit complex engineering steps that are required in modern distributed systems (see the SOCRADES project [3]).

However, an important question remains unanswered: how to efficiently handle distributed systems based on services and consequently their processes? In other words, the challenge is how to describe the processes that regulates the system behavior and how to synchronize and coordinate the execution of the services offered by distributed entities to achieve the desired behavior. Besides the individual control, interaction must occur along individual components of the system. This can be done by defining processes between them that agglomerates together pieces of single process models controlled by the components. Whatever is the strategy chosen by the system engineer, i.e. a centralized control by gluing control models together or peer-to-peer synchronization between components' control models, it should be possible to design the control with a minimal effort.

The present solution for the control of service-oriented systems is based on modular process description of intra- and inter-control activities. The goal is to apply a kind of High-Level Petri Nets (HLPN) to define the predictable and modular control of distributed devices and other components that offer their control capabilities as services. Complex control emerged from individual ones can be tied together from the modular, collaborative and event-based nature of these systems. Unpredicted behavior that interferes in the control may also happen, requiring the support of composable intelligence systems. Semantically rich interactions may also contribute for the definition of control models and description of existing and new available resources and events [4].

After the introductory notes, Section 2 resumes the service-oriented control architecture and its control components. Section 3 describes the High-level Petri Nets based approach for the design and coordination of service-oriented systems and Section 4 refers to the engineering of modular control and the connection of service-oriented control components. At last, Section 5 gives the final conclusions.

II. SERVICE-ORIENTED CONTROL ARCHITECTURE PRINCIPLES

The proposed architecture for reconfigurable automation systems is based in Service-oriented Architecture (SoA) principles, which introduces distributed control structures that exhibits agility, modularity and interoperability [5]. The achieved reconfigurable automation systems are based on modular and simple mechatronic devices, each one providing a set of services that represent its internal functionalities. All the interaction processes among the components are via the access to services that are connected by a communication network.

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

The control of such systems is mainly related to the coordination of the services provided by these distributed mechatronic devices and other components in the system.

Our approach considers several levels of control, namely local control embedded in the mechatronic components, collaborative control among mechatronic components and aggregated control where a specific control service is used, normally to create higher level services based on individual ones. For this purpose, four classes of functional control components were defined, as illustrated in Fig. 1, that can be extended to fulfill the requirements and also combined together to provide an integrated solution [5]: Mechatronic Components (MeC), Smart Mechatronic Component (SMeC), Process Control Components (PCC) and Intelligence Support Component (ISC).

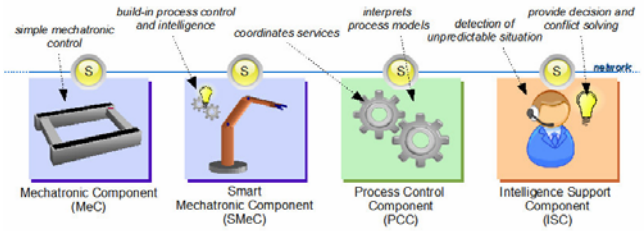

Fig. 1. Overview of the control components classes.

Mechatronic Components are the mechatronic devices, including the device interface and the service-oriented communication module. They can be easily combined to form more complex devices to build the desired automation system in a “Russian doll” manner [2]. The Smart Mechatronic Components are mechatronic components that additionally have embedded their own “smart” control, being able to participate in collaborative activities. Process Control Components (PCC) provide logic process control services based on descriptions able to coordinate different services (e.g. offered by MeC) to create aggregate services of higher value. Intelligence Support Components (ISC) have decision services to support flexibility, indirection and conflict resolution in logic processes, e.g. deciding which alternative should be chosen. It can also force undocumented procedures to support the treatment of unexpected situations.

The proposed architecture is open to permit the introduction of other components, even not related to the direct control, such as components that incorporate and manage workflow databases, ontology/description and product flow. Furthermore, the architecture is flexible enough to support different control solutions: in a more centralized manner by using PCC components to introduce control over the services offered by simple MeC or in a more distributed manner by using SMeC working together to coordinate their services.

The generic internal organization of control components may correspond to functional organs that are responsible for specific tasks, providing the "vital" properties to be able to fulfill their requirements. Since the internal structure is independent from the communication technology and can be differently implemented, a structural pattern is described in [5]. It deploys a set of functional, pluggable and reusable modules and an event-based framework to permit the intercommunication of modules. The main modules are the Service-oriented Communication, Logical Control, Decision Support and Exception Handler, Device Interface and Event Router-Scheduler.

The Service-oriented Communication is responsible to handle the communication between the control component and the other components, i.e. requesting and providing services. The Logical Control is an engine that manages the process model that describes the component behavior and synchronizes it taking in consideration the internal activities and the external events (e.g. services and I/O signals). The Decision Support and Exception Handler module provides intelligence over the control and manages the occurrence of unexpected events. The Device Interface module provides mechanisms to access the physical device, such as setting outputs or reading inputs. At last, the Event Router-Scheduler module provides mechanisms to connect all internal modules and regulate the internal operation of the control component.

These modules are included in the control component according to its needs and possibly implemented using different technologies. For example, the inter-component communication and description can be implemented using SOA4D implementation of Device Profile for Web Services (DPWS). The reader can consult Mendes et *al.* [5] for more information on the architecture principles and how the components are internally structured.

III. A High-Level Petri Nets Approach for the Process Control

In the previous section, it was described the components of the control architecture for a service-oriented automation system. However, the specification of process description and control is needed to assemble part of the internal and external behavior of control components, aiming to achieve the coordination of service-oriented systems.

It is then clear that a flexible and powerful enough approach is necessary to support a low-cost and detailed design-implementation process of the control of service-oriented automation systems, covering the specification, implementation, operation and reconfiguration phases. In service-oriented automation systems, the research on coordination models (service-oriented engineering in general) is a relatively new area, since the proven concepts of SoA at the device level by the SIRENA project [2]. From the side of process modeling in SoA, research is mainly directed to e-business/e-commerce (see [6-8]) and even to Petri Nets formalism to describe service-oriented systems [9-10].

This section introduces a formal approach to develop logic controllers that defines and synchronizes process models in service-oriented systems, covering the requirements of the modular control activities. Please note that these logic controllers could be embedded in SMeC or PCC components.

A. *High-Level Petri Nets Solution for the Development of Logic Controllers*

The proposed approach to develop logic controllers, embedded in control components, is to use a kind of HLPN tailored for service-oriented systems, taking the advantage of their powerful mathematical foundation to represent and validate certain typical relationships, such as concurrency and parallelism, synchronization, resource sharing, mutual exclusion, memorizing, monitoring, supervising, which are typical specifications of automation systems [11-12]. For sake of simplicity, the tokens here represented have no associated data structure, which represent the status of pallets and message exchange in the subsequent models.

HLPN-based logic controllers have to interpret and execute the process model expressed in HLPN, as illustrated in Fig. 2, synchronizing and coordinating the whole process until it reaches the goal. The process model is elaborated according to the behavior of the process, using normal transitions to define structural and logic behavior and three-phase firing transitions to model time-consuming activities, such as the execution of a robot program or a transfer movement.

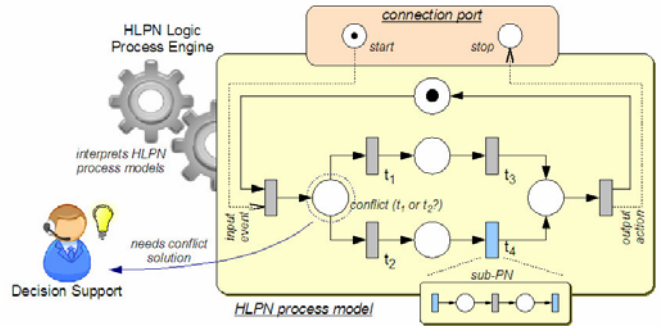

Fig. 2. Example model with the characteristics of the proposed High-Level Petri Nets and supporting structures.

Input events (e.g. a signal indicating the status of a sensor or a service request) and output actions (e.g. a signal to an actuator or a notification of a service execution) can be connected to transitions. As an example, the execution of the HLPN process model, illustrated in Fig. 2, is behind the transfer service that is triggered when the start transfer service is requested. This feature of HLPN allows adapting interfaces to I/Os and services, via the description of transitions.

The real-time execution of the developed HLPN models, aiming to achieve the control of service-oriented systems, requires an interpreter that can evolve their state and control the associated ports (services and I/Os), by setting actions and respond to external events. For this purpose, the engine should detect the enabled transitions, services and actions associated with the enabled transition must be called and, after that, the process model has to be updated to reflect the actual state of the system. Transitions may only be activated by enabling the rule of the HLPN and through the activation of all input events connected to the transition. The transition firing corresponds to the firing rule of the HLPN and the setting of output actions.

The three-phase firing transitions may be exploded (detailed) into a sub-Petri net, creating a step-wise and hierarchical refinement of the model, allowing reaching the control at physical level, i.e. sensing and actuating hardware devices. In Fig. 2, transition *t4* represents the transfer out (right/left) operation that is actually a sequence of different steps, modeled by its explosion in a sub-Petri net. By enabling such transition, the associated detailed Petri net is executed, being fired when the sub-HLPN has reached its terminate state.

Since HLPN models describe processes based on predicted behavior by the system engineer, there are several questions to the occurrence of unexpected events and error manipulation that somehow change the system's behavior and/or should be considered. The process model illustrated in Fig. 2 presents a conflict with transitions *t1* and *t2* being alternatives to the evolution of the model, i.e. two alternative outputs for the transfer execution (right and left). Resolving this conflict means that only one transition in the conflict may be fired. These situations should be handled by the embedded decision support and exception handler module or a special dedicated external component, but there may also be required a modification in the process model to reflect the new state.

The HLPN-based logic controller module, that interprets HLPN models, should be portable in sense of being included in software applications, embedded in micro-controllers devices and primarily in the proposed modular structure of a component, performing the logic control behavior of the automation systems. The module can be integrated in an independent PCC or directly in the SMeC for collaboration purposes.

B. *Building High-Level Petri Net Control Models for Devices*

Aiming to illustrate the development of HLPN-based logic controllers, let us consider the Unidirectional Transfer Unit and Cross Transfer Unit from the FlexLink® Dynamic Assembly System (DAS) 30.

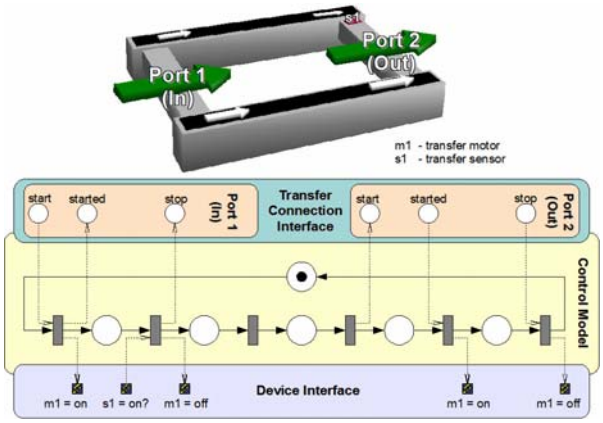

Fig. 3. HLPN control model for the unidirectional transfer unit.

The Unidirectional Transfer Unit, illustrated in Fig. 3, provides two ports (*In* and *Out*) to be connected to other devices, such as similar transfer units, and a device interface to set and read the inputs/outputs of the device. The logic that

controls the three ports is done by a HLPN model, as also illustrated in Fig. 3. The expected behavior is basically related to set ON or OFF the motor *m1* according to the external requests (e.g. start transfer service) and the status of the sensor (which indicates that a pallet is available after a transfer in operation). The two transfer ports can also be used to synchronize the transfer in and out of pallets.

A more complex device is the Cross Transfer Unit, depicted in Fig. 4, which can be seen as a composition of two individual devices: a Unidirectional Transfer Unit and a Cross Lifter Unit. The cross unit allows that pallets be transferred not only in the longitudinal but also in transversal axis, presenting a distinct behavior those presented by the individual transfer unit (note that now the conveyor possesses 6 transfer ports).

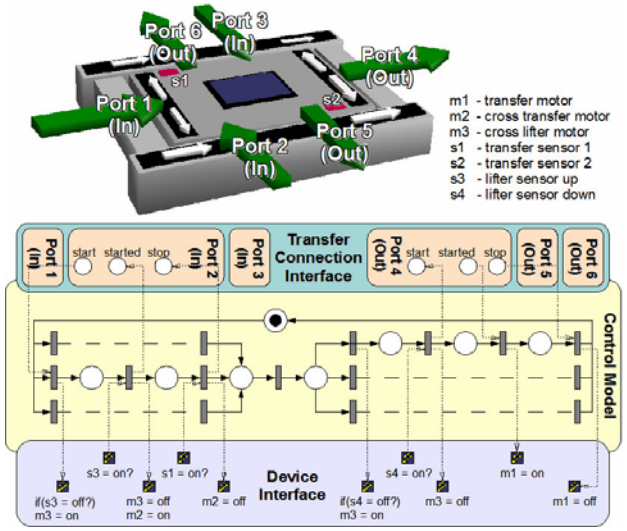

Fig. 4. HLPN control model for the cross transfer unit.

With the lifter unit down and using the motor *m1* it is possible to transfer from port 1 to port 4. When the lifter is up, the transfer from port 2 to port 6 is done using the motor *m2* and the transfer from port 3 to port 5 is done by setting the motor *m2* with reverse polarity. The movement of the lifter is done via the motor *m3*, using two sensors (s3 and s4) to indicate if the lifter is up or down.

The embedded control model has now different options to receive and route a pallet. For simplification, only the logic for the transfer from the port 2 to port 4 is present in Fig. 4, being the others done in a similar way. In this model, a decision is required to choose one among several options that are described and detected by the control module. A special decision support module (or external ISC component) may provide necessary decision information based e.g. on the identifier of the pallet (given by the RFID device in the middle of the unit).

The designed control models for the Unidirectional Transfer Unit and the Cross Transfer Unit have some design flaws and may not respond adequately to several events. These situations were not considered here to reduce the complexity of the models and also because they are out of scope. For example, there are minimal stops between a transfer in and a transfer out operation in the unit illustrated in Fig. 3, even if the movement should be continuous. Other characteristic is related to disturbances in the normal control, such as a pallet that falls down or get stucked on the transport system. Some of these situations can be handled by the indirect control of ISC or by the embedded exception handling module [5].

The logic control model, using High-level Petri nets, is designed according to the type of physical device it represents: the behavior associated to a conveyor is certainly different from the behavior associated to a robot. Also, the logic control model is designed according to the type of operation the physical device performs. For example, an industrial robot can perform different operations, such as handling, welding or painting, which corresponds to different logic models. In this way, some work should be devoted to the identification of the patterns associated to which type of operations the different usual automation devices perform. Having these patterns it is possible to build a library of HLPN models to represent each one of these identified operations, simplifying the development of modular automation systems.

IV. Engineering of Modular HLPN-based Control of Service-oriented Systems

An important piece of the puzzle in the HLPN-based approach for the control of service-oriented systems is to have mechanisms that automatically support the design and validation of automation systems, contributing for an easy and fast reconfiguration. This section focuses on engineering aspects about the control possibilities, the connection and synthesis of HLPN control models and the analysis, validation and simulation of service-oriented systems using HLPN-based control.

A. Interface Mapping and Control Strategies

The proposed control approach does not fix the control layout and topology to be used based on the control modules. Independently from where the control is done, the described interfaces of the models (e.g. transfer connection interface and device interface) must be associated to a type of technology, e.g. access the inputs/outputs of a device, permit the conversation of distributed models via service-orientation and connect two local control models.

Fig. 5 describes two different control strategies that use the same control model for the Unidirectional Transfer Unit, represented in Fig. 3: the embedded control and the centralized control.

The embedded control is used whenever it is possible to include and/or define the logic inside the device. The internal control engine, that runs a HLPN process model, has to synchronize the activity and to evolve the state based on the events received from the Transfer Service and the Device Interface (directly embedded in the device). External access is only done via the service invocation (that can be used by service requestors) and its two ports.

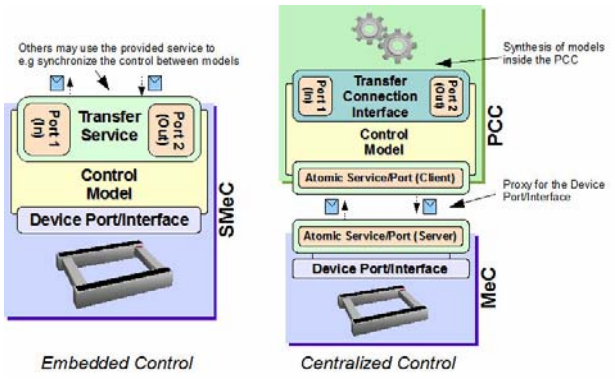

Fig. 5. Embedded and centralized control strategies.

In the case of using a centralized control approach, i.e. when the HLPN-based logic controller runs in a PCC component that does not directly access the device's I/O, the distribution can be handled by including an intermediate proxy mechanism (service-oriented) so that the client (i.e. the central control) can easily access the atomic operations provided by the server (i.e. a (S)MeC). This case is frequent when device vendors may only provide the access to the interface and/or do not enable the ability to program the device itself. In theses cases, or the device is prepared to be integrated in a distributed environment or a PCC component is required to synchronize the control with the devices.

B. Connection and Synthesis of HLPN Control Models

The development of modular service-oriented automation systems requires that (S)MeC components (i.e. the equipment itself) be connected together working in a decentralized manner or working under the supervision of a PCC. 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 SMeCs). The question is if it is simply a matter of overlapping ports or if it requires more complex "connection logics".

To be able to provide a reliable activity between control models the logical and/or service-oriented connection of them must comprehend several rules. In a first instance, each model should have a compatible port interface to be connected to; not only a set of matching operations but also a complementary functionality (see the example of transfer in and transfer out services). 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 of specific "glue" that corresponds to a logic that is dependent on the type of the connection. This can be seen as a protocol for the communication that establishes, among others, a message synchronization pattern for the communication.

Using HLPN to represent the control models of each component, the connection task is simplified through the use of interfaces and ports and a semi-automatic matching between them. Fig. 6 illustrates the application of the previous appointments to perform the connection of two Transfer Units, each one described using a HLPN control module, similar to the logic represented in Fig. 3.

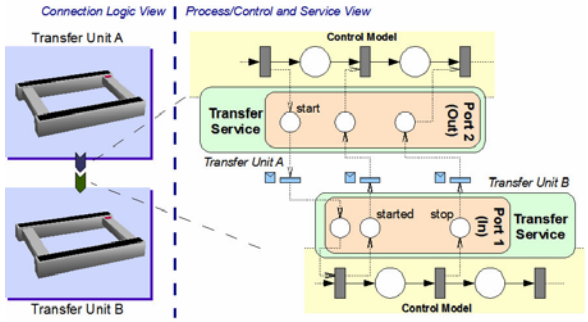

Fig. 6. Connection of two transfer units using HLPN models.

In this example, the interface of port 2 (Out) from the Transfer Unit A is connected to the interface port 1 (In) from the Transfer Unit B. Additionally a logic connection is used to group together these two processes, mapping them to integrate both in one bigger control process.

The design of more complex systems require that some processes should be coordinated hierarchically, for example, to aggregate services based on individual ones. The PCC components provide coordination and aggregation services and support the complex process flow and interaction of services in the system, according to the process model or its synthesis based on smaller ones. For this purpose, it implements the logic for the process-oriented execution and sequencing of atomic services (from its point of view), and provides a high-level interface for the aggregated process.

Fig. 7 illustrates an adaptation of the example of Fig. 6, considering the inclusion of a PCC as coordinator and synthesizer of processes, representing a centralized control approach to the service-oriented approach.

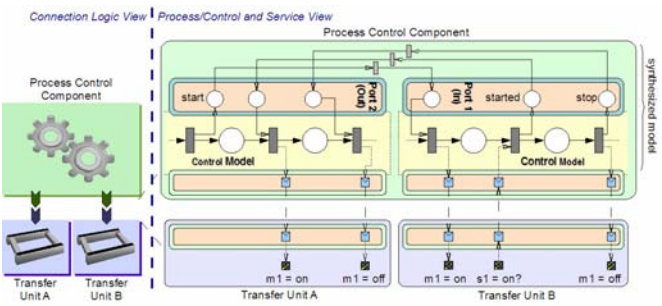

Fig. 7. Connection and synthesis of two transfer units' models using a PCC.

Several significant differences can be identified when comparing this control strategy with the distributed one. First of all, control models of each device run independently and are synchronized by a central entity, i.e. a PCC component, in opposite to the distributed approach where each SMeC has its own coordination mechanism. This supposes that the communication between the PCC and (S)MeCs components is done in a simplistic way, using atomic operations instead of complex connections built in transfer ports. In the centralized

approach, all the control process is performed by the PCC component, including the interaction between the local control models of the (S)MeCs. In spite of these differences, it is important to note that the control models of each device remain the same in both control strategies, reducing the engineering efforts of the system designer.

The examples given here are based on pallet transfer mechanisms, but the same control solution can be extended to other control purposes and modular automation processes, in sense of building more complex systems. As an example, the transfer units can be connected to other transfer units or compatible devices, such as cross transfer units and robots with transfer capabilities.

C. *Analysis and Simulation of High-Level Petri Nets Control Models*

The designed HLPN 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 into the system model. Combining modeling and validation methods and using the powerful theoretical foundations of HLPN formalism, which are completely based on the functional analysis theory, a practical procedure of formally designing service-oriented automation systems is applied in this work. In fact, HLPN models 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.

Basically, two types of analysis can be made: the qualitative and the quantitative analysis. The first one verifies the compliance of certain desirable specifications of the system components and system behavior such as absence of deadlocks, reversibility, finite number of system states, boundedness of resources and possible control sequences. The quantitative analysis, often called performance evaluation, takes into account system specifications, therewith checking the system compliance with specified performance indexes, such as production period for a product, throughput of the system, percentual use of a resource and manufactured parts per time units.

The goal of the analysis is to give a complete and correct description of the behavior of a control component. This task is crucial, because if only one part of the model is erroneous, the validity of the complete system is jeopardized and correctness of validation, simulation and analysis cannot be guaranteed.

V. CONCLUSION

This paper introduces a High-Level Petri nets approach for the modular control of service-oriented automation and production systems. The modularity associated to the use of HLPN for the control of service-oriented systems allows the easy development of complex automation systems based on the arrangement of modular and components and leaves different design choices to the developer. Also the re-configuration of the automation system is simplified due to the modularity and adaptability provided. In fact, the proposed architecture can be compared to the Lego™ concept: as Lego, the proposed service-oriented architecture does not propose rigid automation systems, but a set of interconnectable modules that may be grouped to form a desired evolvable automation system. Using elementary components it is possible to build systems, which grouped in a particular way will constitute bigger and more complex systems.

The use of HLPN-based logic controllers to coordinate the services provided by distributed entities, according to the logic behavior model, contributes to achieve this demanding for modularity, flexibility and re-configurability.

Further research is related to the more detailed specification of a unified methodology for the design, implementation and reconfiguration of HLPN logic controllers, and integration of decision support mechanisms for conflict detection and resolution.

ACKNOWLEDGMENT

The authors would like to thank the partners of 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] M. G. Mehrabi, A.G. Ulsoy, Y. Koren, “Reconfigurable Manufacturing Systems and their Enabling Technologies”, *International Journal of Manufacturing Technology and Management*, 1(1), 2000.

[2] 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.

[3] M. Taisch, “Service-Orientated Cross-layer Infrastructure for Distributed Smart Embedded Devices”, 2nd World Congress on Engineering Asset Management and the 4th International Conference on Condition Monitoring, presentation, 2007.

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

[5] J. M. Mendes, P. Leitão, A. W. Colombo, F. Restivo, “Service-oriented Control Architecture for Reconfigurable Production Systems”, submitted to the 6th IEEE International Conference on Industrial Informatics, 2008.

[6] I. J. G. Santos, M. Fluegge, N. P. Tizzo and E. R. M. Madeira, “Challenges and Techniques on the Road to Dynamically Compose Web Services”, Proceedings of the 6th International Conference on Web engineering, 2006, pp. 40-47, ACM Press.

[7] E. Karakoc, K. Kardas and P. Senkul, “A Workflow-Based Web Service Composition System”, Proceedings of the 2006 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology, IEEE Computer Society, 2006, pp. 113-116.

[8] W. T. Tsai, C. Fan, Y. Chen and R. Paul “DDSOS: A Dynamic Distributed Service-Oriented Simulation Framework”, Proceedings of the 39th annual Symposium on Simulation, IEEE Computer Society, 2006, pp. 160-167.

[9] J. Thomas, M. Thomas and G. Ghinea, “Modeling of Web Services Flow”, IEEE International Conference on E-Commerce, 2003, pp. 391-398.

[10] Y. Fu, Z. Dong and X. He, “An Approach to Web Services Oriented Modeling and Validation”, Proceedings of the 2006 International Workshop on Service-oriented Software Engineering, ACM Press, 2006, pp. 81-87.

[11] T. Murata, "Petri nets: Properties, Analysis and Applications", *IEEE*, vol. 77, 1989, pp. 541-580.

[12] J. M. Couvreur and J. Martinez. “Linear Invariants in Commutative High Level Petri Nets”, Lecture notes in Computer Science, Vol. 483, pp. 146-165, Springer Verlag, 1990.