International Journal of Production Research
Publication details, including instructions for authors and subscription information:
http://www.tandfonline.com/loi/tprs20

High-level Petri nets for the process description and control in service-oriented manufacturing systems

J. Marco Mendes [a], Paulo Leitão [b c], Armando W. Colombo [d e] & Francisco Restivo [a c]

[a] Faculty of Engineering of University of Porto, Porto, Portugal

[b] Department of Electrical Engineering, Politechnic Institute of Braganca, Braganca, Portugal

[c] LIACC - Artificial Intelligence and Computer Science Laboratory, Porto, Portugal

[d] Schneider Electric Automation GmbH, Seligenstadt, Germany

[e] University of Applied Sciences Emden/Leer, Emden, Germany

Available online: 04 Aug 2011

To cite this article: J. Marco Mendes, Paulo Leitão, Armando W. Colombo & Francisco Restivo (2012): High-level Petri nets for the process description and control in service-oriented manufacturing systems, International Journal of Production Research, 50:6, 1650-1665

To link to this article: http://dx.doi.org/10.1080/00207543.2011.575892

International Journal of Production Research
Vol. 50, No. 6, 15 March 2012, 1650–1665

Taylor & Francis
Taylor & Francis Group

High-level Petri nets for the process description and control in service-oriented manufacturing systems

J. Marco Mendes[a]*, Paulo Leitão[bc], Armando W. Colombo[de] and Francisco Restivo[ac]

[a]*Faculty of Engineering of University of Porto, Porto, Portugal;* [b]*Department of Electrical Engineering, Politechnic Institute of Braganca, Braganca, Portugal;* [c]*LIACC - Artificial Intelligence and Computer Science Laboratory, Porto, Portugal;* [d]*Schneider Electric Automation GmbH, Seligenstadt, Germany;* [e]*University of Applied Sciences Emden/Leer, Emden, Germany*

(*Received 18 June 2010; final version received 15 March 2011*)

The use of service-orientation principles in manufacturing systems is a promising solution to achieve modularity, flexibility, re-configurability and interoperability. Crucial issues in these service-oriented systems are the description and co-ordination of the execution of the services offered by the distributed entities. This paper introduces an integrated approach for the design, analysis, validation, simulation and process execution of service-oriented manufacturing systems, using the High-level Petri net formalism as the formal language to describe the system behaviour. The use of the proposed approach contributes to achieving an easier and faster development of these solutions and provides the basis to support modularity and re-configurability.

Keywords: flexible manufacturing systems; reconfigurability; service-oriented systems; High-level Petri nets

1. Introduction

The current tendency in manufacturing is to achieve the requirements imposed by global markets that demand customised and high quality products at lower prices with very short delivery times through modularity, flexibility and re-configurability. This new class of manufacturing systems requires the existence of distributed and modular control entities that collaborate to accomplish distributed control activities, while being able to self-organise to drive evolvable organisation structures. In fact, reconfigurable systems, instead of incorporating all the flexibility once at the beginning of their life cycle, incorporate basic process models that can be rearranged or replaced quickly and reliably (Mehrabi *et al*. 2000).

Traditionally, design and manufacturing activities have taken place sequentially rather than simultaneously leading to inefficient and time consuming iterations between design and manufacturing stages (Shukor and Axinte 2009). Moreover, since the introduction of the programmable logic controllers (PLC) in the early 1970s, significant efforts have been made to overcome the original PLC's limitations in terms of decentralised usage, aiming to address the topics of modularity and re-configurability. New manufacturing paradigms and emergent technologies, which take advantage of the newest mechatronics, information and communication technologies are being researched and applied to increase the modularity, flexibility and re-configurability, such as multi-agent systems (MAS) (Wooldridge 2002), holonic manufacturing systems (HMS) (Deen 2003) and recently service-oriented architectures (SoA) (Jammes and Smit 2005a).

MAS are characterised by decentralisation and parallel execution of activities based on autonomous entities, called agents. These systems have the capability to respond promptly and correctly to change, and differ from the conventional approaches due to their inherent capabilities to adapt to emergencies without external intervention (Wooldridge 2002). In a similar way, HMS are pyramidal systems based on the concept of holon. A holon is a part of a manufacturing system that is made up of sub-ordinate parts and in turn is part of a larger whole. SoA are centred in the notion of service-orientation, i.e. the entities provide their functionalities and skills in the form of services that may be searched, requested and used by other entities (Melzer 2007).

The work on MAS, HMS and SoA provides a good framework for the new generation of control systems in the sense that they support re-configurability and agility quite naturally. In this work, SoA systems are used to develop

*Corresponding author. Email: marco.mendes@fe.up.pt

ISSN 0020–7543 print/ISSN 1366–588X online

http://dx.doi.org/10.1080/00207543.2011.575892
http://www.tandfonline.com

modular and reconfigurable manufacturing control systems, taking advantage of its capabilities of encapsulation, modularisation and interoperability. The resulting solutions are made of control entities that work together and use service-orientation as their main co-ordination mechanism. In such distributed systems based on services, aggregation and co-ordination methods are required, from the basic specification to the more complex engineering. Particularly, the challenge is how to describe the processes that regulate the system behaviour and how to synchronise and co-ordinate the execution of the services offered by distributed entities to achieve the desired behaviour. A possible solution is to use Web Services Business Process Execution Language (WS-BPEL) (OASIS 2007), but it only specifies interactions among Web services and does not consider the internal logic of software components and services. Additionally, it is heavily based on Web services technology, business-oriented and with weak validation features. Other solutions that have been applied are based on 61131-3 languages (Bonfatti *et al*. 1995, Erickson 1996, IEC 2003) and IEC 61499 function blocks (Lewis 2001) with the objective of adapting industrial standards to SoA. For the flexibility and expandability of the event-driven system software, programming paradigms such as a Petri net are useful, excelling in the expression and analysis of dynamic system behaviour (Kurihara *et al*. 2002). Similar to Petri nets, TNCES (timed net condition/event systems) are used for modelling interaction-aware services (Popescu and Lastra 2008).

This paper considers a kind of High-level Petri net for the description and co-ordination of process behaviour in service-oriented automation systems, taking advantage of the powerful theoretical foundations of this formalism, which are completely based on the functional analysis theory. The proposed approach contributes to the development of modular, collaborative and re-configurable service-oriented manufacturing systems by providing important features, namely the formal specification and configuration of control models, the easy composition of individual devices to build more complex systems and the integration of decision-making to support the conflict resolution. It also supports the design, analysis, validation and simulation of the system behaviour during the design phase and before its deployment into the operation phase.

The remainder of the paper is organised as follows: Section 2 overviews the concept of service-oriented automation and particularly the main foundations of service-oriented control architecture for reconfigurable production systems. Section 3 presents the kind of High-level Petri net used in this work and discusses its advantages as a formal language for the process description and co-ordination in service-oriented systems. Section 4 introduces a methodology to develop High-level Petri net based service-oriented control systems. Section 5 describes the case study scenario used to illustrate the applicability of the proposed concepts and Section 6 describes the application of the proposed High-level Petri net based service oriented control to the experimental case study. Finally, Section 7 rounds up the paper with the final conclusions.

2. Service-orientation for reconfigurable production systems

Service-orientation paradigm is a promising approach to introduce modularity and re-configurability in the development of manufacturing control systems. This section briefly presents the basic concepts of service-oriented systems and their promising application to the automation field, and overviews the basic architectural concepts of a service-oriented production control system that will be later used to host the High-level Petri net process controllers.

2.1 *Service-oriented systems*

The SoA paradigm is an abstract concept of a software architecture based on the idea of encapsulating resource functionalities as services that can be offered, searched and used by other entities (the service requesters) (Melzer 2007), without knowing their underlining implementation. For this purpose, providers publish the services they want to offer in a service registry, and requesters search the services they want to use.

In these systems, a pertinent question is about how services interact. Service composition (Peltz 2003) is the combination of single services and all the interaction patterns between them. Commonly, terms such as service orchestration and choreography are used for this purpose. Orchestration is the practice of sequencing and synchronising the execution of services, which encapsulate business or manufacturing processes (Jammes *et al*. 2005, Jammes and Smit 2005b). 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. Some technologies used in orchestration and choreography are Web Services Description Language (WSDL) for the abstract description of the services interfaces, Simple Object Access Protocol (SOAP) for messaging and communication, and Universal Description, Discovery and Integration (UDDI) for registry and discovery of services.

The SoA paradigm was originally applied in electronic commerce and business systems, using Web services technology, but is being progressively adopted by other fields. The SoA principles fit well with collaborative automation, in the sense of autonomous, re-usable and loosely-coupled distributed components, supporting the vertical enterprise integration, from sensors and actuators level to the strategic level. The SIRENA Project (Jammes and Smit 2005b) has contributed for the visibility of 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 has been going on covering the engineering of such systems, including the modelling, semantic description and collaboration. Some research projects, such as the EU FP6 SOCRADES (http://www.socrades.org/) contributed to the applicability of service-orientation principles in industrial automation. The current EU FP7 IMC-AESOP project (http://www.imc-aesop.eu) is expected to envision, design, implement and demonstrate an SoA approach for monitoring and control of process control applications.

2.2 *Reconfigurable service-oriented manufacturing control*

The demand for manufacturing systems that exhibit a high degree of re-configurability imposes strong requirements on the way the systems are designed, installed, operated and re-configured (Mendes *et al*. 2008). A service-oriented manufacturing control system was designed to address this challenge, built upon modular, distributed and simple mechatronic control entities, working under the service-orientation principles, i.e. providing a set of services that represent their internal functionalities. The architecture identifies several types of entities that participate in the control, as illustrated in Figure 1 (Mendes *et al*. 2008): Mechatronic, Process Manager, Intelligence Support and Product Manager entities.

Mechatronic entities combine the mechanical, electronic and software components, allowing the local control of the physical device according to a process model that describes a sequence of actions, for instance reading the status of a sensor or starting the execution of a robot program. Smart Mechatronic entities are extensions of Mechatronic entities that have embedded their own intelligent capabilities to support decision-making and exception handling. In more complex systems, it is necessary to have entities that provide global process co-ordination mechanisms. The Process Manager entities provide this kind of mechanisms, being able to co-ordinate the execution and sequencing of atomic services based on a process description. A Process Manager entity can also offer aggregated services of higher value, made of individual ones. The Intelligence Support entities provide decision services to support flexibility, indirection, conflict resolution and unexpected situations in the logic process control, for instance deciding which alternative resources services should be chosen. Product Manager entities are responsible for managing the products execution, providing information about how to execute the products, based on the product and process models.

In analogy to the Lego^{TM} concept, grouping these elementary and inter-connectable entities allows for building very complex systems, which are modular, reconfigurable and evolvable. In fact, the reconfiguration is achieved due to the easy re-organisation of the entities and the services they provide, reflected by the modification of the connections between the devices presented in the system.

The service-oriented entities use an anatomical-like structure, comprising several ‘organs’ (i.e. functional modules), to provide a modular and event-driven design and deployment. The designed functional modules, illustrated in the bottom right of Figure 1, are: Communication, Logic Controller, Decision and Exception Handler, Device Interface and Event Router-Scheduler.

The Communication module is responsible for handling the interaction between the control entity and the other entities, i.e. requesting and providing services. The Logical Controller module is responsible for managing the process model that describes the entity’s behavior and for synchronising the specified actions, taking in consideration the internal activities and the external events (e.g. services and I/O signals). The Decision and Exception Handler module provides decision support for conflict resolution and unexpected situations. The Device Interface module provides mechanisms to access the physical device, such as setting outputs or reading inputs. The

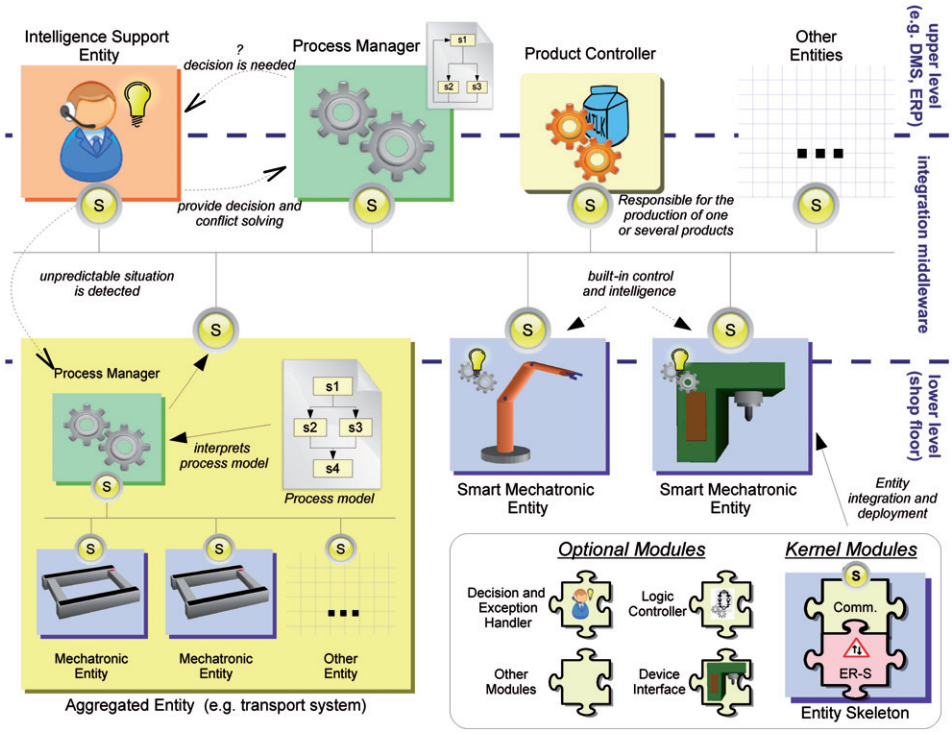

Figure 1. Reconfigurable service-oriented manufacturing control architecture.

Event Router-Scheduler module, which can be compared with the nervous system of living beings in the sense of carrying impulses from and to different organs, provides mechanisms to connect internal modules and to regulate the operation of the control entity. Other modules can be included to perform other functionalities, it being only necessary that they respect the rules specified by the Event Router-Scheduler module.

These modules are included in the control entity according to its needs and are possibly implemented using different technologies. As an example, the inter-entity communication can be implemented using the SoA for Devices (SOA4D) implementation of Device Profile for Web Services (DPWS) (OASIS 2009) or using any other communication technology such as Common Object Requesting Broker Architecture (CORBA). For more information on the architecture principles and how the entities are internally structured, please consult Mendes *et al*. (2008).

3. High-level Petri nets approach to process description and control

In the previous section, the service-oriented manufacturing architecture and the internal structure of architectural entities were described. However, the specification of the process description and control is needed, aiming to achieve the co-ordination of service-oriented systems. Traditionally, the process control of such systems can be defined in several languages, such as IEC 61131-3, IEC 61499 or WS-BPEL. However, it is clear that a more flexible and powerful approach is necessary to support a low-cost and detailed design-implementation process of the service-oriented manufacturing control systems, covering the specification, implementation, operation and reconfiguration phases in an integrated manner. In the proposed control approach, a kind of High-level Petri net (ISO 2000), tailored for service-oriented systems, is used as the kernel for the design, modelling, analysis, validation, simulation and execution of process control in service-oriented systems. This section presents the main characteristics of this kind of High-level Petri net and discusses its advantages over other approaches.

3.1 *High-level Petri net formalism*

Petri nets have recently emerged as a promising approach for modelling, simulating and analysing flexible and automated manufacturing systems (Moore and Gupta 1996). The Petri net formalism, based on a well-founded

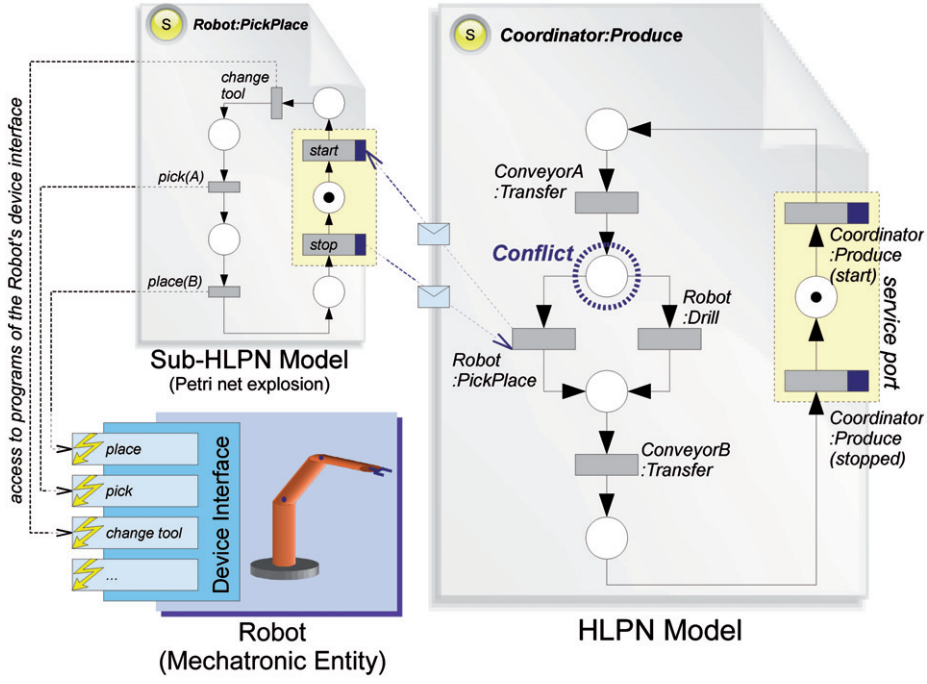

Figure 2. Characteristics of the proposed high-level Petri net approach.

mathematical theory, has a very good capacity to graphically and formally represent and validate certain typical relationships, such as concurrency and parallelism, synchronisation, resource sharing, mutual exclusion, monitoring and supervision, which are typical specifications of manufacturing systems (Murata 1989, Zurawski and Zhou 1994). A Petri net is a directed, bipartite graph, formally defined according to the following definition (Murata 1989).

Definition 1: A Petri net is a 5-tuple, $PN=(P,T,F,W,M)$, where $P=\{p_1,p_2,\ldots,p_m\}$ is a finite set of places, $T=\{t_1,t_2,\ldots,t_n\}$ is a finite set of transitions, $F\,(P\times T)\cup(T\times P)$ is a set of arcs connecting places and transitions (flow relation), $W:F\rightarrow\{1,2,3,\ldots\}$ is a weight function associated to each arc, and $M:P\rightarrow\{1,2,3,\ldots\}$ is the marking of the Petri nets; note that $P\cap T=0$ and $P\cup T\neq 0$.

In a Petri net each node is either a place or a transition. Tokens occupy places representing resources or states of the system. The dynamic evolution of the Petri net, i.e. the change of a state or marking in the model, is described through the enabling and firing rules, stated in the following definitions.

Definition 2: Considering $^\bullet t$ as the set of input places for a transition t, this transition is enabled if and only if $\forall pi\in{}^\bullet t$, $m(p_i)\geq w(t,pi)$, i.e. if each input place p_i of the transition t is marked with at least $w(p_i,t)$ tokens, where $w(p_i,t)$ is the weight of the arc from place pi to the transition t.

Definition 3: An enabled transition t fires reaching a new marking m by considering the rule, $m'(p)=m(p)+W(t,p)-W(p,t)$, i.e. removing $w(p,t)$ tokens from each input place p of the transition t, and adding $w(t,p)$ tokens to each output place p of the transition t.

The High-level Petri nets used in this work extend the ordinary Petri net formalism with additional features for a more powerful representation of complex discrete event systems, notably considering timed transitions, stepwise refinement, coloured representation, and associating transitions to service operations and I/Os, as illustrated in Figure 2.

A modelled manufacturing system can comprise activities, represented by transitions, which take place at a much faster (or slower) pace than others. Additionally, there may be a requirement for the introduction of transitions that correspond to purely logical aspects of the system behaviour, which have no associated time (Colombo *et al.* 1997). In these circumstances, two distinct types of transitions are considered:

- *Immediate transitions*, which fire in zero time, are used to model atomic activities, such as sending a message.

- *Timed transitions*, which have associated a delay time that specifies the amount of time that must elapse before the transition fires, are used to represent time consuming activities, e.g. a robot operation; these timed transitions have associated the notion of three-phase firing (Colombo *et al*. 1997).

A top-down methodology is used to achieve a formal specification of the logic control structure, refining the model to include more system operation details, and consequently reaching the control at physical level. For this purpose, timed transitions are exploded into a detailed sub-Petri net, so that a large Petri net can be obtained. By enabling the timed transition, the associated sub-Petri net is executed; the transition fires when the sub-Petri net has reached the terminate state. In Figure 2, the transition *Robot:PickPlace* represents the pick and place operation that is actually a sequence of different steps, modelled by its explosion into a sub-Petri net.

In industrial manufacturing applications, Petri net models may become very complex and difficult to handle, due to a set of Petri nets' weak points (Vyatkin *et al*. 2001, Leitão *et al*. 2003). This is particularly true when the system presents many instances of the same element (e.g. resources), being the model increased in terms of structure and components, in a complex manner. Moreover, a process-based model does not usually fully correspond to the control programming behaviour (Peng and Zhou 2003), due to the complexity that has normally to be represented. The use of coloured Petri net (Jensen 1992) principles in the proposed High-level Petri net specification, adds another dimension to the control and especially to the flux of information, compressing the representation of the system elements and supporting the modelling of more complex and bigger systems (Feldmann *et al*. 1996, Holloway *et al*. 1997, Colombo *et al*. 2001). For this purpose, function guards are associated to the transitions, representing restrictions to the type of data value, i.e. coloured marks that a transition can move during its firing.

Aiming to support the connection of the model to the 'real world', input events (e.g. a signal indicating the status of a sensor or a service request to start the execution of a robot operation) or output actions (e.g. a signal to an actuator or a notification of a service execution) can be connected to transitions. Special ports are then defined, to link the control model represented by a Petri net to several communication standards (e.g. I/O interface, service interface and other control models). In other words, the ports are specific gates to synchronise control models, being in some cases also related to the physical connectivity of the entities. Transitions may only be activated by the enabling rule of the Petri net and the input conditions guided by specific I/Os or messages (if any) of the port. As an example, the execution of the Petri net process model illustrated in Figure 2 is behind a connection port that is triggered when the input event is received and of course the Petri net is in a favourable state. The existence of ports in the Petri net models also allows connecting several models together (since their ports match together). This permits building bigger and more complex systems in a modular way, and providing interoperability between control modules.

3.2 *Advantages of using High-level Petri nets*

The use of the High-level Petri net formalism over other methodologies, such as the referred WS-BPEL language, can be discussed and compared. The main advantages of using High-level Petri nets as language for the process description and control logic for service-oriented production systems are:

- Makes easier the system modelling and understanding by using a graphical and mathematical notation.
- Aggregates, composes and co-ordinates behaviour models in a modular way.
- Simplifies the validation, analysis and simulation of the control system during the design phase.
- Makes easier the detection of conflicts and unexpected situations; significant information extracted from the Petri net structure, e.g. the set of T-invariants, can be used to support the decision-making.
- Drives and synchronises the run-time behaviour of the entities, achieving a powerful and effective control mechanism, since the Petri nets are represented and manipulated internally as a set of matrices.
- Controllers are easy to design, implement, maintain and re-configure, reducing substantially the development time when compared with the traditional approaches.
- Adapts interfaces to the real physical I/Os and services, via the description of the transitions.

In this work, one of the major requirements for the development of service-oriented automation systems was the compliance and implementation based on the DPWS specification and framework. The choice of Petri nets over other forms that are more standard in automation and/or SoA (e.g. IEC 61131-3 and WS-BPEL), besides the

referred inherent advantages, is related to the lower complexity of the nets in terms of implementation and the hypothetical better performance for device integration. Another advantage was the flexibility of the Petri nets' implementation and availability of the source code to introduce new features that would not be possible when limits are given by standards.

One of the first documented implementations of Petri net based sequence controllers was developed in the early 1980s by Hitachi Ltd (Murata 1986). It was successfully used in real applications to control parts on an assembly system and to control an industrial robot. However, the use of High-level Petri net formalism is principally recommended at the process control level, mainly due to the real-time constraints imposed by the lower device levels, which normally use PLCs running IEC61131-3 programs. The main focus of this work is not the lower-level control of physical devices (i.e. controllers to be embedded in Mechatronic entities) but essentially the description and execution of the process control (i.e. to be embedded in the Process Manager entities). Meanwhile, envisaging supporting the vertical integration, the proposed engineering approach supports the implementation of High-level Petri net engines for the entire range of enterprise control levels, from business levels to the hardware device level.

4. Methodology to develop High-level Petri net based service-oriented control

In the proposed approach, High-level Petri net formalism is used as formal language to describe and execute the behaviour of automation systems. Concretely, logic controller modules, embedded in (Smart) Mechatronic and Process Manager entities, are engines that interpret and execute behavioural process models expressed in High-level Petri nets. They are responsible for co-ordinating and synchronising the services' execution of the whole process until it reaches the desired goal.

The engineering methodology used to develop such High-level Petri net service-oriented controllers introduces several innovation aspects, such as the integration of modelling, analysis, validation and execution tasks. This leads to a reduction of the development time and cost, supporting also the capability to model the system behaviour from the high-level control abstraction to the hardware control level. The proposed methodology comprises three different phases or stages, notably the modelling, the analysis and validation, and the deployment and execution, as illustrated in Figure 3.

The following sections describe the methodology phases.

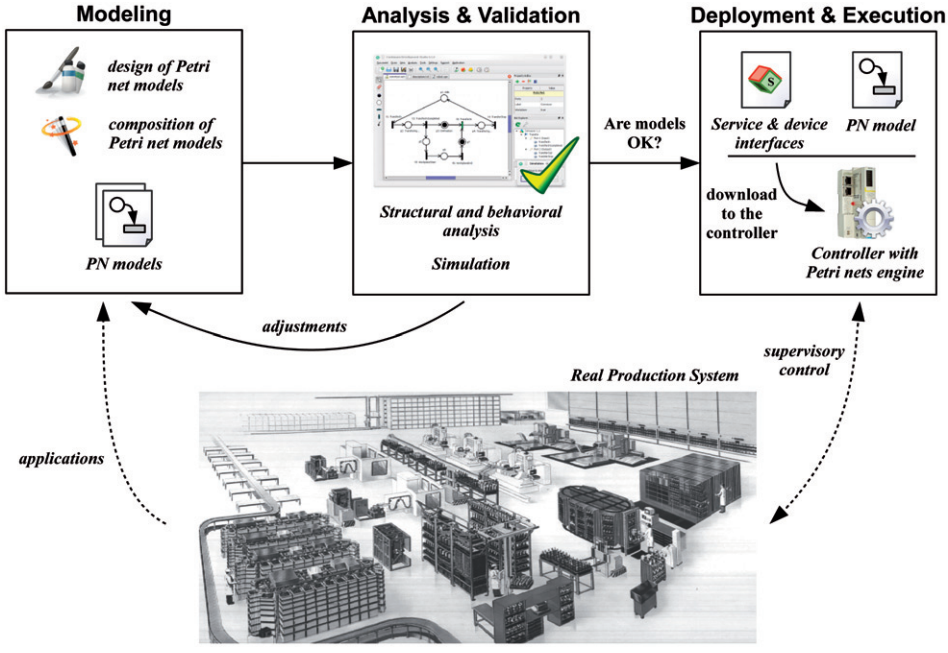

Figure 3. Methodology for the development of high-level Petri net based service-oriented manufacturing systems.

4.1 *Modelling*

An important key issue in the development of High-level Petri net based service-oriented control systems is the specification and configuration of the High-level Petri net control models, namely their description, co-ordination, composition and synthesis. In this article the modelling of Petri nets is based on the fundaments expressed in Mendes *et al*. (2009a) and Mendes *et al*. (2010).

The logic control model using High-level Petri nets is designed in a bottom-up manner according to the process behaviour that is intended to describe and control, such as robots and conveyors. Each model represents all possible discrete states of such a resource and also all the functions that this resource is able to expose as services, for instance move-piece, pick-part and transfer-pallet. The High-level Petri net model is dependent on the physical resource it represents (e.g. the behaviour associated to a conveyor is certainly different from the behaviour associated to a robot), and the type of operation the physical resource performs (e.g. an industrial robot can perform different operations, such as handling, welding or painting). The identification of the patterns associated to usual operations allows building a library of High-level Petri net models that can be re-used later, simplifying the development of modular solutions. In these models, the services representing time-consuming processes are associated to transitions of Petri net models. Note that the concept of ports is used here to define a logical endpoint for the service and to associate it to the connection points of physical devices.

The development of modular service-oriented manufacturing systems requires the composition of individual High-level Petri net models into a co-ordination model, and consequently the composition of services. This task follows the same rules of configuring a required resource's layout, i.e. taking into account, among others, the competition, concurrency and shared resources behavioural relationships. The result is a High-level Petri net model of the whole system. The aggregation and composition by connecting control modules using the High-level Petri net formalism is simplified through the use of interfaces, ports and a semi-automatic matching between them (with ontologies and semantics playing a crucial role). The connection of control models must obey to a set of reliable rules:

- Each model should have a compatible port interface to be connected to; not only a set of matching operations but also a complementary functionality (e.g. transfer in and transfer out services).
- A connection can only be established after an agreement between the involved partners; in a more complex scenario it may involve some negotiation.
- The connection itself needs 'glue' that corresponds to the communication that establishes, among others, a message synchronisation pattern for the communication.

Figure 4 represents an example where two Petri nets (PN_1 and PN_2) are composed via the addition of composition logic, resulting in the composed Petri net $PN_{1\cup 2}$. The generated new Petri net model is the composition of several smaller ones, where individual models are maintained in their distributed units and synchronised together via the network.

When the composition is done, new inter-logic is generated. In this example, two transitions from different models are matched by connecting them via a place (and corresponding arcs). Additionally, the direction is also needed (e.g. t_2 from PN_1 is the input of t_1 from PN_2). This approach, in addition to simplifying the development of greater models, also facilitates the synchronisation of models viewed as singular entities.

The design of more complex systems requires that some processes should be co-ordinated hierarchically, for example, to aggregate services based on individual ones. The Process Manager entities provide co-ordination and aggregation of atomic 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.

4.2 *Analysis and validation*

Analysis and validation of systems is crucial, not only to prove that they are free of errors, but also to test if they perform the desired specifications. Using the powerful theoretical foundations of High-level Petri net formalism, a formal analysis and validation of the designed service-oriented automation system can be performed. In fact, the designed High-level Petri net models, including the information about the system operation (e.g. process plans, resources, layout and control laws), will constitute a computational model that can be easily analysed, validated and simulated in the design phase. This allows performing different experiments under distinct scenarios and proceeding into the implementation phase only after the verification of the system correctness.

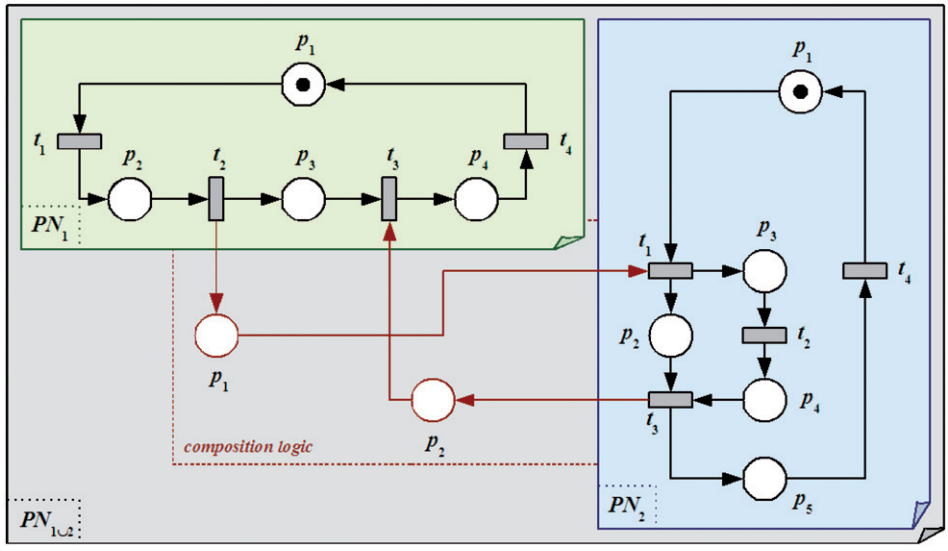

Figure 4. Composition of Petri nets PN_1 and PN_2 resulting in $PN_{1\cup2}$.

Basically, two types of analyses can be made using the Petri net theory: the qualitative and the quantitative analysis.

The qualitative analysis, based on the structural analysis of the model, allows verifying its structural and behavioural properties, extracting conclusions about the functionality of the system, such as the existence of deadlocks, the bounded capacity of resources, the conservativeness, the possible control sequences and the existence of structural and behavioural conflicts in the system (Feldmann *et al.* 1996). These desirable specifications can be verified by analysing the reachability tree or applying linear algebra methods (see Murata (1989) and Zurawski and Zhou (1994) to get detailed information about how to extract properties from Petri net models). Additional analysis can be performed by obtaining the *T*- and *P*-invariants from the incidence matrix. The analysis of *P*-invariants allows confirming mutual exclusion relationships among places, functions and resources involved in the model structure, and the analysis of the *T*-invariants allows the identification of work cycles. The analysis and validation of coloured Petri nets is mainly performed through the analysis of *P*- and *T*-invariants analysis (Jensen 1992).

Since this work considers the hierarchical refinement of models, it is important to guarantee that after replacing a timed transition by a more refined sub-net, the large Petri net preserves its live and bounded properties. Vallete has proven that using stepwise refinement, all the properties of a large Petri net can be deduced from the analysis of the initial Petri net and each one of the sub Petri nets (Vallete 1979).

The quantitative analysis allows simulating the system behaviour, therewith checking the system compliance with specified performance indexes, such as the lead time to produce a product, the throughput of the system and the percentual use of the resources. The information extracted from the temporal evolution of the High-level Petri net control model reflects the temporal sequence of the system operation, being easy to discover cyclic evolution and the existence of bottlenecks. This analysis allows to reproduce abnormal conditions (or at least, conditions that cannot be easily created in real world) to debug the system behaviour.

The analysis of High-level Petri net control models, both quantitative and qualitative, allows validating the specifications of the system's behaviour, verifying the correctness of the models and verifying if the models fulfil the desired specifications. It is also possible to refine strategies or specifications of the system, detecting errors and mistakes before implementing the real production system. Only if the control model verifies all necessary properties (structural and behavioural), is it possible to be sure that the model is correct from the functional point of view and can be seen as a virtual representation of the system. In this case, the control model is ready to be used in the execution phase, i.e. interpreted and executed by a High-level Petri net based engine.

4.3 *Deployment and execution*

The High-level Petri net control models, designed and validated in previous phases, should at this stage be deployed into a controller platform to be executed.

The real-time execution of High-level Petri net control models, aiming to achieve the control of service-oriented systems, requires that an interpreter, i.e. the Logic Controller embedded in Smart Mechatronic and Process Manager entities, can evolve their state and control the associated ports (services and I/Os), by setting actions and responding to external events. For this purpose, the engine should detect the enabled transitions, which may only be activated according to the enabling rule of Petri nets, and especially considering all input events connected to the transition. The transition firing corresponds to the firing rule of the High-level Petri nets and the setting of the actions associated with the fired transition (e.g. setting I/Os or notifying the execution of a service). After that, the process model has to be updated to reflect the current state of the system.

Being able to define the services in the control entities, thoughts have to be carried about how and where the control models should run. The High-level Petri net based Logic Controller module should be portable in the sense of being included in software applications, embedded into PC or micro-controller devices and primarily in the proposed modular structure of a service-oriented entity, performing the logic control behaviour of the automation system. Simpler devices may embed already pre-compiled High-level Petri nets instead of having an interpreter, since the logic and communication themselves don't require to be changed at runtime. It is important to mention that other controlling mechanisms, such as using IEC 61131-3 programming languages, can be used in parallel by different devices. The only requirement is that the communication should guarantee the interoperability and synchronisation between the different and distributed control mechanisms with the employment of Web services.

5. Description of the case study scenario

The specification of the High-level Petri net approach to drive the process control in service-oriented systems will be illustrated using an experimental case study scenario. It is based on the FlexLink® Dynamic Assembly System 30, depictured in Figure 5, which combines flow-oriented production control and modular automation with ergonomic manual assembly solutions, providing flexibility and versatility.

The case study only considers the upper part of the transfer system, made of nine transfer units (conveyors) of the unidirectional and cross types. The unidirectional transfer unit provides an input and an output port, and the cross transfer unit provides transfers not only in the longitudinal but also in transversal axis. Moreover, the cross transfer unit may be seen as a composition of two devices, namely a unidirectional transfer unit and a lifter with

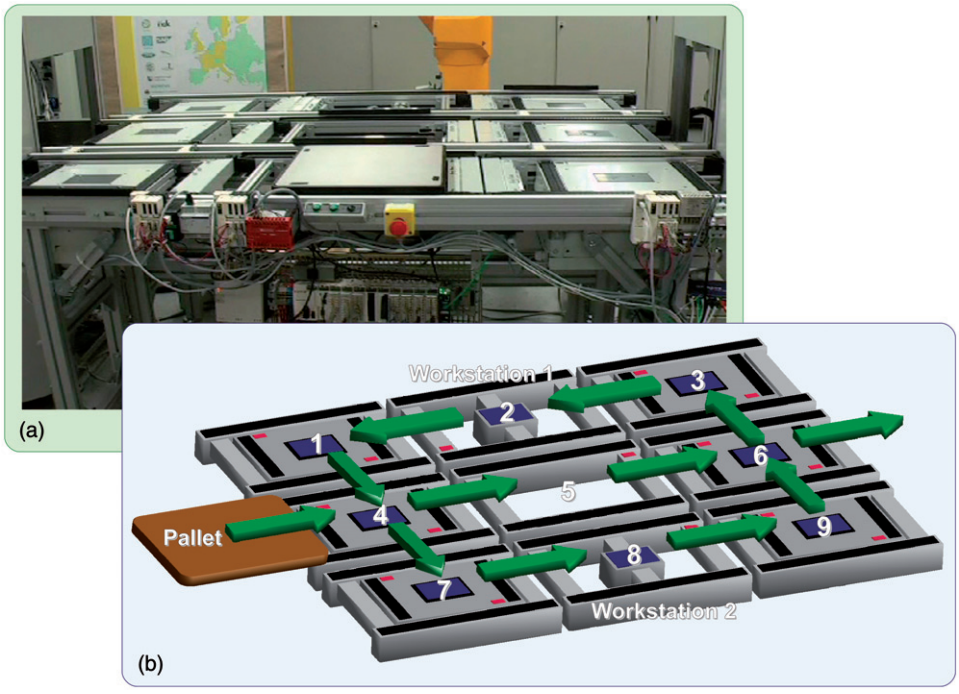

Figure 5. Layout of the transfer system: (a) real system, (b) virtual representation.

directional transfer capabilities. Each transfer unit has a RFID (radio-frequency identification) reader/writer for identifying the pallets and transmitting information to them.

The pallets enter in the system through the transfer unit 4 and are conveyed using alternative paths to achieve the two workstations associated to the transfer units 2 and 8. These transfer units have the possibility to halt the pallet during the required amount of time for its processing. Lastly, the pallets are routed outside through the transfer unit 6.

This case study scenario will be used to accommodate a service-oriented control system, serving to illustrate its specification and configuration, where the control is based on single High-level Petri net models that can be set together to balance the overall operation of the transfer system.

6. Experimental validation

Aiming to illustrate the applicability of the proposed methodology for the development of High-level Petri net based service-oriented control system, a control solution was designed for the described experimental case study.

6.1 *Description and synthesis of the control model*

The identification of the entities that are part of the system's behaviour is crucial. Figure 6 represents the mapping of the transfer units into service-oriented control entities, resulting in nine Mechatronic entities. In case of using Mechatronic entities without the Logic Controller, additional Process Manager and Intelligent Support entities must be used as controlling supervisors.

The expected behaviour of each unidirectional and cross transfer unit, constituting Mechatronic entities, needs to be specified, using High-level Petri net control models.

The unidirectional transfer unit provides two ports (*In* and *Out*) to be connected to other devices, such as similar transfer units, and a port to set and read the outputs/inputs of the device interface. The logic that controls the three ports is done by a High-level Petri net model represented in Figure 7.

The expected behaviour is basically related to set ON or OFF the motor m_1 according to the external requests (e.g. start the *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 synchronise the *transfer in* and *transfer out* of pallets. The High-level Petri net control model for units 2 and 8 considers a special transition that can be used to represent the time execution during the workstation's operation.

A more complex device is the cross transfer unit, depicted in Figure 8. It has six different transfer ports (of *In* and *Out* types) and one device port. With the lifter unit down and using the motor m1 it is possible to transfer pallets from port 1 to port 4. When the lifter is up, the transfer from port 2 to port 6 is done using the motor m_2 and the

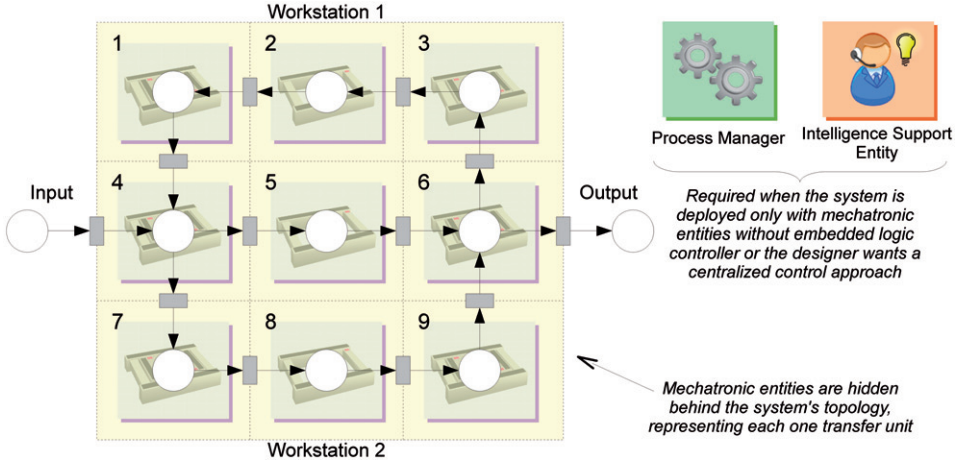

Figure 6. Control entities and global behaviour description.

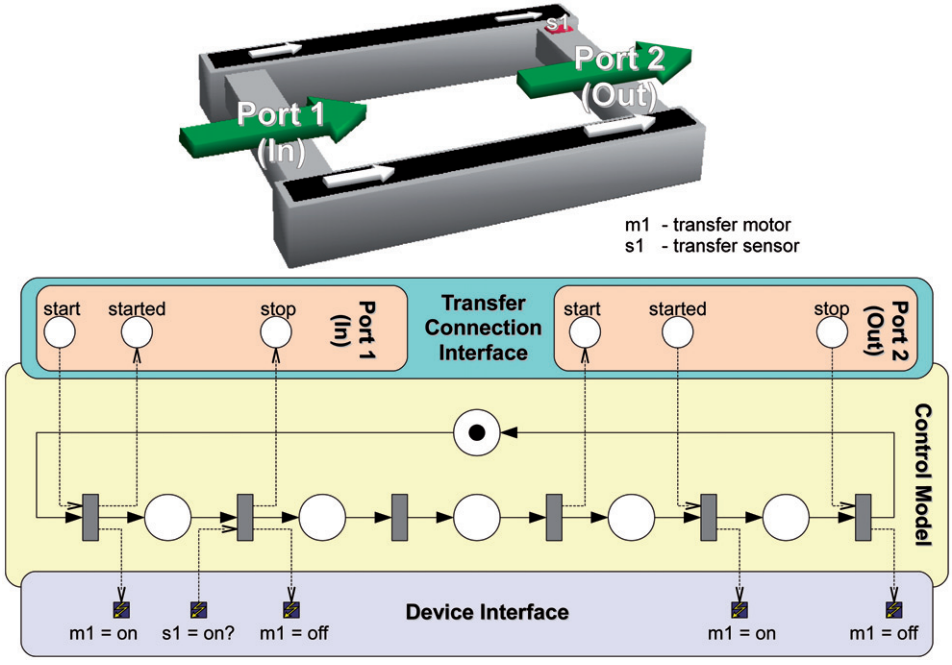

Figure 7. High-level Petri net control model for the unidirectional transfer unit.

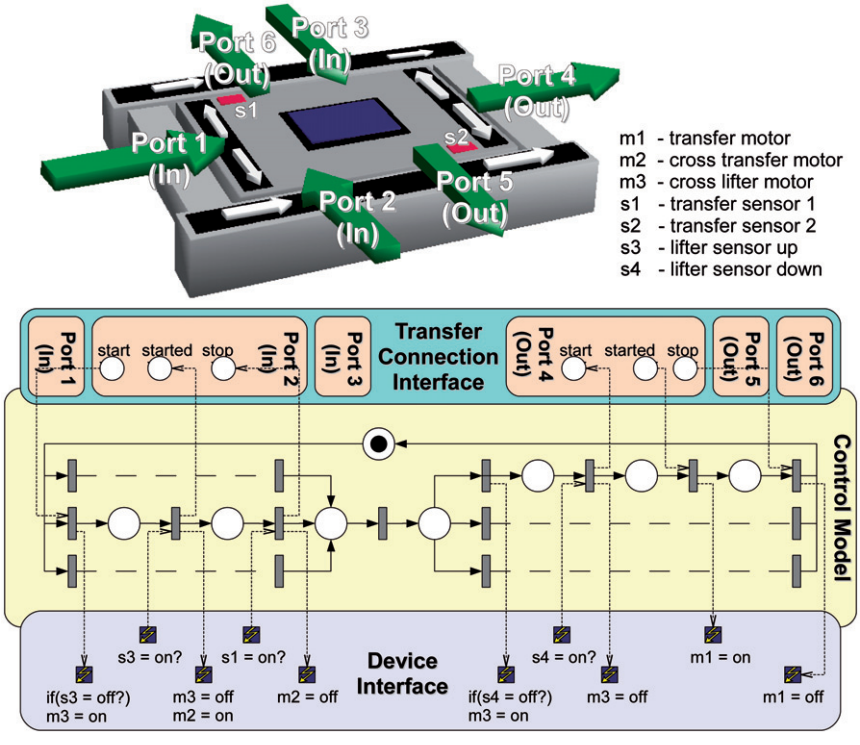

Figure 8. High-level Petri net control model for the cross transfer unit.

transfer from port 3 to port 5 is done by setting the motor m_2 with reverse polarity. The movement of the lifter is done via the motor m_3, using two sensors (s_3 and s_4) to indicate if the lifter is up or down.

Transfer units 1, 3, 4, 6, 7 and 9 are deployed using the control logic represented in Figure 8. Most of them (1, 3, 7 and 9) only require one path of the several available, the others being deactivated. On the other side, units 4 and 6

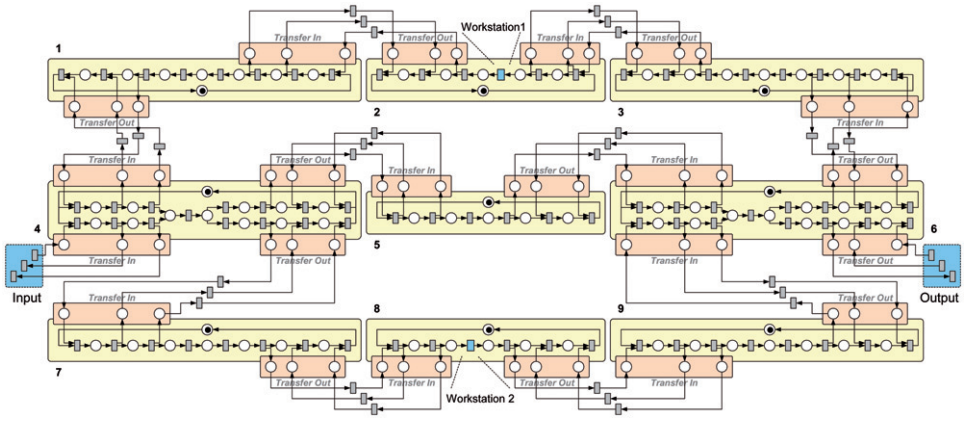

Figure 9. Synthesis of the complete control for the case scenario.

are made of several routing possibilities and the control model has now different options to receive and route a pallet (for simplification, only the logic of ports 2 and 4 is present in Figure 8, the others being done in a similar way). At this stage, a decision is required to choose one among different options that are described in the High-level Petri net control model. For instance, a special Decision and Exception Handler module (or in alternative an external Intelligent Support entity) may provide the necessary decision support, for example based on the identification of the pallet (given by the RFID reader in the middle of the transfer unit) and in the production plans.

A synthesised control model can be easily elaborated to represent the complete case study behaviour by composing the individual High-level Petri net models designed for the transfer units, as illustrated in Figure 9. As previously described, the solution uses the models as building blocks and connects them together according to the external visible ports.

This transport system has different routing options for the pallets that can populate it. The main options are transporting pallets to the desired workstations and passing through the system to another one that is connected (e.g. when it is not necessary to execute any workstation service over the pallet). This involves the presence of conflicts in the control, namely by supplying different paths for the pallets. The monitor places, presented in each individual High-level Petri net model, are responsible for regulating the shared resources, forbidding the access of a pallet to an occupied transfer unit. A further remark is the deactivation of some control branches of the cross transfer units in the scenario, namely the ones in the corners, i.e. 1, 3, 7 and 9.

6.2 *Analysis and validation of the control model*

The modelling of the process behaviour for the individual devices, and posterior composition, was performed by using the Continuum Development Studio (CDS) software tool (Mendes *et al*. 2009b). CDS, which is based on an extensible Document/View framework, provides an engineering tool for service-oriented automation entities and devices, for example, supporting the visual description, analysis, simulation and deployment. The analysis and validation of the synthesised High-level Petri net service-oriented control model was also performed with CDS tool. Figure 10 illustrates the structural and behavioural analysis of the synthesised control model.

This structural analysis shows that the designed High-level Petri net model is live, which guarantees the deadlock freeness, i.e. the execution of a sequence of movements of a pallet in the transport system is done without stopping in an undetermined intermediate state. It is also possible to verify that the Petri net model is:

- Reversible, which means that the model returns to the initial state, through well defined work cycles in the execution of a sequence of pallet movements.
- Conservative, which means that once in the transfer system, the tokens representing pallets do not disappear nor are new tokens created.
- Bounded, which means that the number of pallets in the system is limited to the maximum value of m, due to the existence of a monitor place that regulates the available pallets in the system.

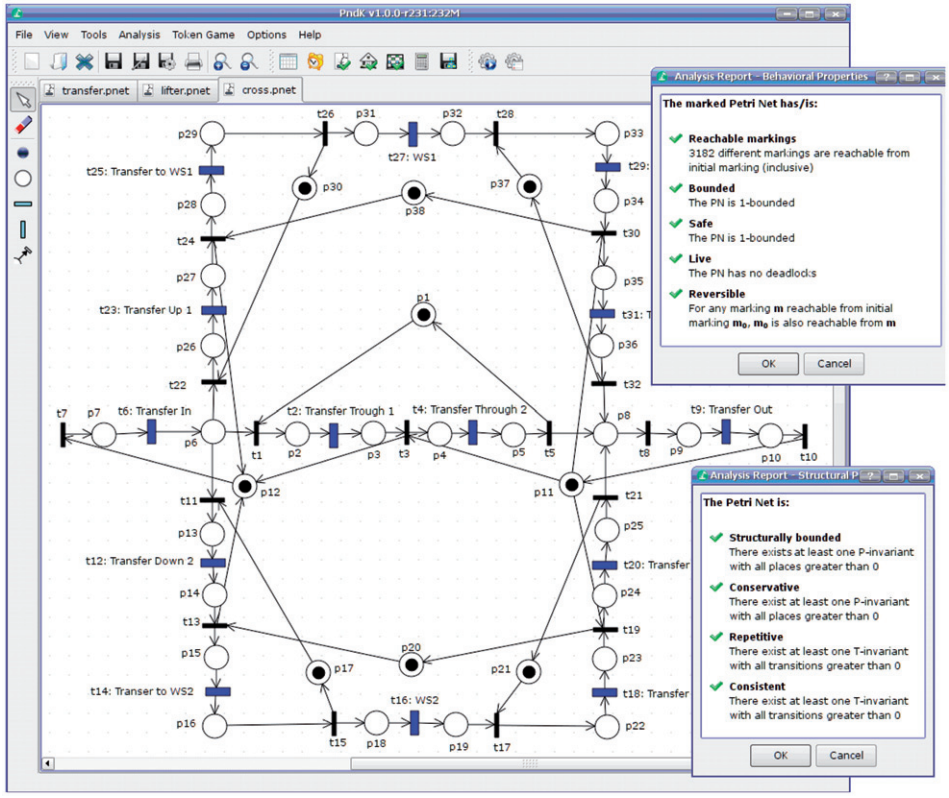

Figure 10. Structural analysis of the control model using the CDS tool.

The analysis of the *T*- and *P*-invariants allows validating several systems' specifications, namely:

- The set of *P*-invariants describes the mutual exclusion presented in each control model of the transfer units, showing that, in a certain moment of time, a pallet can only occupy one of the systems' transfer units.
- The set of *T*-invariants of the synthesised model for the transfer system describes possible sets of operations. Translated into the system topology it may refer to all possible routing sequences of pallets along this system.
- The work cycles represented by the set of *T*-invariants illustrates the sequences of possible operations, supporting the decision-making system to achieve the best and shortest path between two locations.

The model may consider deadlock situations due to the presence of circular paths if all transfer units are simultaneously occupied by pallets. The overlapping conflicts of these paths support the resolution of these issues by activating alternative ways to route some pallets.

The quantitative analysis requires the introduction of the time parameter associated to the transitions. For this purpose, deterministic distribution times have been used, since the system is composed of real hardware/software components with deterministic time behaviour. The token game simulation performed in the CDS tool allows to verify the evolution of the system behaviour and to extract performance indexes. As an example, it is possible to verify how the system behaves when different routings are selected to convey the pallets.

6.3 *Deployment and execution of the control model*

With the CDS it is possible to deploy the designed control model into a service-oriented device running a Petri net engine. The Petri net file is combined with device and service information (that are represented in the model) to generate deployment files, which are uploaded to a network-connected device. The device is then ready to request and/or be requested services in the network according to the designed model.

The examples described in this case study are based on mechanisms for the pallet transfer, but the same control solution can be extended to other control purposes and modular automation processes, in the 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. The simplicity and extensibility that the High-level Petri net approach offers in the service aggregation and composition contributes to achieve more modular, scalable and reconfigurable automation control solutions.

7. Conclusions

The paper introduces a High-level Petri net based approach for the process description and co-ordination of service-oriented manufacturing systems, aiming to achieve modularity, flexibility and re-configurability. The proposed approach constitutes an integrated methodology to develop these service-oriented systems, allowing the analysis, validation and simulation during the design phase before the deployment phase. It also supports the execution of these systems by powering logic controllers that interpret and run High-level Petri net process behaviour models, which co-ordinate the services provided by the distributed entities.

Using elementary High-level Petri net based control entities it is possible to build systems, which can be grouped to constitute larger and more complex systems. This feature allows the easy development of complex systems and supports its reconfiguration and the evolution during its life-cycle. Special attention was devoted to the features of the kind of High-level Petri nets used to develop the control models, and the procedures for their analysis and simulation, contributing to achieving the demand for modularity, flexibility and re-configurability. The proposed approach introduces important innovations, namely the formal specification and configuration of control models, the easy aggregation of individual devices to build more complex systems.

Future work is being devoted to the integration with 2D/3D engineering tools to support an even easier design of service-oriented manufacturing systems and to the integration of High-level Petri net kernels with DPWS to be embedded in portable controllers. Evaluation of the proposed work is intended to follow the implementation on real industrial equipment as well as qualitative and quantitative comparison to existing solutions.

Acknowledgements

The authors would like to thank the European Commission and the partners of the EU ICT FP7 projects 'ArchitecturE for Service-Oriented Process – Monitoring and Control' (IMC-AESOP) and 'Co-operating Objects Network of Excellence' (CONET) for their support.

References

Bonfatti, F., Gadda, G., and Monari, P.D., 1995. Re-usable software design for programmable logic controllers. *ACM SIGPLAN Notices*, 30 (11), 31–40.

Colombo, A.W., Carelli, R., and Kuchen, B., 1997. A temporised Petri net approach for designing, modelling and analysis of flexible production systems. *International Journal of Advanced Manufacturing Technology*, 13 (3), 214–226.

Colombo, A.W., Neubert, R. and Schoop, R., 2001. A solution to holonic control systems. *Proceedings of the 8th IEEE international conference on emerging technologies and factory automation*, 15–18 October 2001 Antibes-Juan les Pins, France. Piscataway, NJ: IEEE, 489–498.

Deen, S., 2003. *Agent-based manufacturing: Advances in the holonic approach*. Berlin/Heidelberg: Springer-Verlag.

Erickson, K., 1996. Programmable logic controllers. *IEEE Potentials*, 15 (1), 14–17.

Feldmann, K., Schnur, C., and Colombo, A.W., 1996. Modularised, distributed real-time control of flexible production cells, using Petri nets. *International Journal of IFAC Control Engineering Practice*, 4 (8), 1067–1078.

Holloway, L., Krogh, B., and Giua, A., 1997. Survey of Petri net methods for controlled discrete-event systems. *Discrete-Event Systems: Theory and Applications*, 7 (2), 151–190.

IEC Standard, 2003. IEC 61131-3, Programmable Controllers – Part 3: Programming Languages.

ISO/IEC Standard, 2000. High-level Petri Nets - Concepts, Definitions and Graphical Notation. Final Draft International Standard ISO/IEC 15909.

Jammes, F. and Smit, H., 2005. Service-oriented paradigms in industrial automation. *IEEE Transactions on Industrial Informatics*, 1 (1), 62–70.

Jammes, F. and Smit, H., 2005. Service-oriented Architectures for Devices – the SIRENA view. *Proceedings of the 3rd IEEE international conference on industrial informatics*, 10–12 August 2005, Perth, Australia, 140–147.

Jammes, F., Smit, H., Lastra, J.L.M., and Delamer, I., 2005. Orchestration of service-oriented manufacturing processes. *Proceedings of the 10th IEEE international conference ETFA*, 1, 19–22 September 2005, Catania, Italy, 617–624.

Jensen, K., 1992. Coloured Petri nets: Basic concepts, analysis methods and practical use. *Monographs on Theorical Computer Science*, 1.

Kurihara, K. *et al*., 2002. Factory automation control software designing method based on Petri nets. *International Journal of Production Research*, 40 (15), 3605–3625.

Leitão, P., *et al*., 2003. Formal specification of holonic control system ADACOR, using High-level Petri nets. *Proceedings of the 1st IEEE international conference on industrial informatics*, 21–24 August 2003, Banff, Alberta, Canada, 263–272.

Lewis, R., 2001. Modeling distributed control systems using IEC 61499 – applying function blocks to distributed systems. Institution of Electrical Engineers, Stevenage, UK.

Marco, M.J. *et al*., Service-oriented control architecture for reconfigurable production systems. *In*: *6th IEEE International Conference on Industrial Informatics, 2008*, 13–16 July 2008, Daejeon, South Korea, 744–749.

Mehrabi, M.G., Ulsoy, A.G., and Koren, Y., 2000. Reconfigurable manufacturing systems and their enabling technologies. *International Journal of Manufacturing Technology and Management*, 1 (1), 114–131.

Melzer, I. *et al*., 2007. Service-orientierte Architecturen mit Web Services [Service-oriented architectures with web services], 2nd ed., Elsevier, Spektrum Akademischer Verlag.

Mendes, J.M., *et al*., 2010. Composition of Petri nets models in service-oriented industrial automation. *Proceedings of the 8th IEEE international conference on industrial informatics*, 13–16 July 2010, Osaka, Japan, 578–583.

Mendes, J.M., *et al*., 2009. Customisable service-oriented Petri net controllers. *Proceedings of the 35th annual conference of the IEEE industrial electronics society*, 3–5 November 2009, Porto, Portugal, 4341–4346.

Mendes, J., *et al*., 2009. Software methodologies for the engineering of service-oriented industrial automation: The continuum project. *Proceedings of the 33rd IEEE international conference on computer software and applications*, 20–24 July 2009, Seattle, Washington, 452–459.

Moore, K.E. and Gupta, S.M., 1996. Petri net models of flexible and automated manufacturing systems: a survey. *International Journal of Production Research*, 34 (11), 3001–3035.

Murata, T., *et al*., 1986. A Petri net based controller for flexible and maintainable sequence control and its application in factory automation. *IEEE Transactions on Industrial Electronics*, 33 (1), 1–8.

Murata, T., 1989. Petri nets: Properties, analysis and applications. *IEEE*, 77, 541–580.

OASIS Standard, 2007. Web Services Business Process Execution Language Version 2.0.

OASIS Standard, 2009. Devices Profile for Web Services Version 1.1.

Peltz, C., 2003. Web Services Orchestration. Hewlett Packard, Co.

Peng, S. and Zhou, M., 2003. Sensor-based stage Petri net modelling of PLC logic programs for discrete-event control design. *International Journal of Production Research*, 41 (3), 629–644.

Popescu, C. and Lastra, J., 2008. Modelling interaction-aware services from an orchestration viewpoint. *Proceedings of the 6th IEEE international conference on industrial informatics*, 13–16 July 2008, Daejeon, South Korea, 780–785.

Shukor, S.A. and Axinte, D.A., 2009. Manufacturability analysis system: Issues and future trends. *International Journal of Production Research*, 47 (5), 1369–1390.

Vallete, R., 1979. Analysis of Petri nets by stepwise refinements. *Journal of Computer and Systems Science*, 18, 35–46.

Vyatkin, V., Hanisch, H. and Ivanov, G., 2001. Application of formal methods for deep testing of controllers in holonic systems. *Proceedings IEEE international conference on information technology in mechatronics*, 1–3 October 2001, Istanbul, Turkey, 53–58.

Wooldridge, M., 2002. *An introduction to multi-agent systems*. John Wiley & Sons.

Zurawski, R. and Zhou, M., 1994. Petri nets and industrial applications: A tutorial. *IEEE Transactions on Industrial Electronics*, 41 (6), 567–583.