Chapter 15
Factory of the Future: A Service-oriented System of Modular, Dynamic Reconfigurable and Collaborative Systems

A.-W. Colombo[1], S. Karnouskos[2] and J.-M. Mendes[3]

Abstract Modern enterprises operate on a global scale and depend on complex business processes performed in highly distributed production systems. Business continuity needs to be guaranteed, while changes at some or many of the distributed shop-floors should happen on-the-fly without stopping the production process as a whole. Unfortunately, there are limits and barriers that hinder the business requirements in beating tackled timely in the shop-floors due to missing modularity, agile reconfigurability, collaboration and open communication among the systems. However, as the number of sophisticated networked embedded devices in the shop-floors increases, service-oriented architecture (SOA) concepts can now be pushed down from the upper information technology level to the device level, and provide a better collaboration between the business systems and the production systems. This leads to highly modular, dynamic reconfigurable factories that build a collaborative system of systems that can adapt and optimize its behaviour to achieve the business goals. The work presented in this chapter shows directions to achieve this dynamism by means of SOA introduction in all layers, increased collaboration and close coupling of production sites and enterprise systems.

Keywords Service-oriented architecture, dynamic reconfigurable systems, collaborative systems

[1] A.-W. Colombo (🖂)
Schneider Electric Automation GmbH, Steinheimer Str. 117, D-63500 Seligenstadt, Germany
email: armando.colombo@de.schneider-electric.com

[2] S. Karnouskos
SAP Research, SAP AG, Vincenz-Priessnitz-Strasse 1, D-76131 Karlsruhe, Germany

[3] J. –M. Mendes
Faculty of Engineering, University of Porto, Rua Dr. Roberto Frias s/n, 4200-465 Porto, Portugal

15.1 Introduction

The business world is highly competitive, and in order to successfully tackle everyday challenges, operational managers and executives demand high dependability and wide visibility within their business process networks. The later is done usually via business key performance indicators. However, in order to provide up-to-date information and be able to react in a flexible and optimal way to changing conditions, real-time information must flow via all layers from the shop-floor up to the business process level. In that sense enterprises are moving towards service-oriented infrastructures that bring us one step closer to the vision of real-time enterprises. Applications and business processes are modelled on top of and using an enterprise-wide or even cross-enterprise service landscape. For any solution to be easily integrated in this environment, it is recommended to feature a service-based approach (Karnouskos *et al.*, 2007).

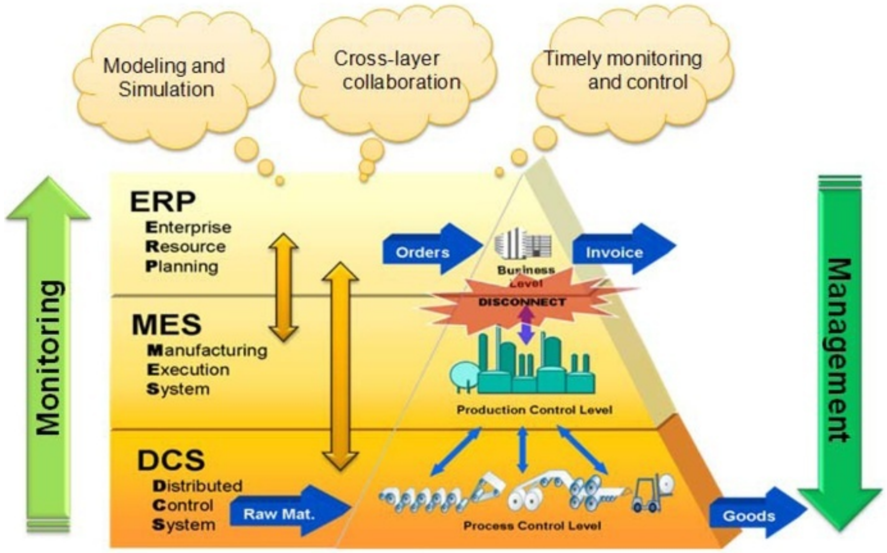


Fig. 15.1 Monitoring, control and collaboration for service-based future factories

Currently, shop-floor intelligent systems based on distributed embedded devices concentrate the programming of the behaviour and intelligence on a handful of large monolithic computing resources accompanied by large numbers of dumb devices. The intelligence and behaviour are tailored and individually programmed for each application. However, as we are moving towards the "internet of things" (Fleisch and Mattern, 2005) where millions of devices will be interconnected, provide and consume information available on the network and cooperate, new capabilities as well as challenges come into play. As these devices need to interoperate and collaborate pursuing common production management and control goals (Colombo and Harrison, 2008), the service-oriented approach seems a promising solution, i.e. each device offers its functionality as a service, while in parallel it is possible for it to discover and invoke new functionality from other services on-demand.

The future factory should be seen as a system of systems (Jamshidi, 2008), where complex and dynamic systems interact with each other in order to achieve the goals at system-wide but also at local level. To realize this, timely monitoring and control as well as open communication and collaboration in a cross-layer fashion are key issues (Figure 15.1). Modern approaches such as the service-oriented-architecture (SOA) paradigm when applied holistically can lead to the desired result. By considering the set of intelligent system units as a conglomerate of distributed, autonomous, intelligent, proactive, fault-tolerant and reusable units, which act as a set of cooperating entities, a new dynamic infrastructure, a system of systems that is able to provide a better insight to its components to the higher levels and better react to dynamic business changes, can be realized.

15.2 The Emergence of Cooperating Objects

The rapid advances in computational and communication parts in embedded system, is paving the way towards highly sophisticated networked devices that will be able to carry out a variety of tasks not in a standalone mode as usually done today, but taking into full account dynamic and context specific information. These "objects" will be able to cooperate, share information, act as part of communities and generally be active elements of a system.

Cooperating objects (COBS) is an emerging domain (Marron *et al.*, 2009) with strong roots in (i) ubiquitous computing, (ii) embedded systems and (iii) wireless sensor networks. They possess features attributed to all three aforementioned domains, in order to allow their cooperative behaviour to emerge.

Generally COBS possess the ability and possibly the willingness to work or act together; however, their cooperation can be intentional or unintentional. Intentional cooperation can be forced (rare) or voluntary (the usual case). In a system view, COBS have goals and work together because of one or more common (even partially common) goals or means to achieve the end-goals. Single COBS are parts of teams; as such cooperative behaviour may be shown at higher level, e.g., group level, and not be clearly identifiable at object level.

The COBS are physical objects and should have the means to cooperate. This assumes:

- networking capabilities – communication;
- interaction with other objects, including heterogeneous ones; and
- autonomous behaviour.

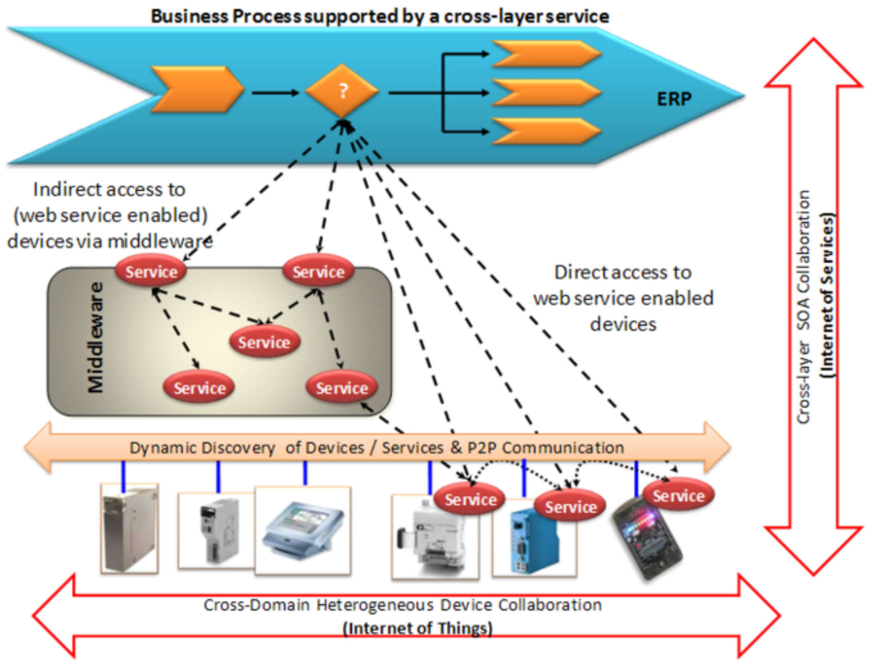

Business Process supported by a cross-layer service
?
ERP
Indirect access to (web service enabled) devices via middleware
Middleware
Service
Direct access to web service enabled devices
Dynamic Discovery of Devices / Services & P2P Communication
Cross-layer SOA Collaboration (Internet of Services)
Cross-Domain Heterogeneous Device Collaboration (Internet of Things)

Fig. 15.2 The envisioned cross-layer SOA-based vertical and horizontal collaboration

There are several flavours of COBS. Advanced COBS can process the context of cooperation intentionally, act on it and intentionally extend it, change it or stop it. As such COBS may possess logic to understand semantics and build complex behaviours. This eventually means that they can be part of dynamic complex ecosystems. Of course a cooperating object is governed by its internal rules and constrained by its resources; however, its behaviour is the result of a negotiation and potential benefit yield with respect to the external collaboration.

In our work within the SOCRADES project (www.socrades.eu), we have focused on some of the characteristics on COBS, i.e., automation devices for the future factory that have the envisioned capabilities to support web services (WS) on devices. As such their functionality is depicted as a service that can be used by other entities and eventually be part of complex orchestrations. As depicted in Figure 15.2 this will enable cross-layer collaboration not only at horizontal levels e.g., among cooperating devices but also at vertical level between systems located at different levels of a computer-integrated manufacturing or plant-wide system enterprise architecture (Pfadenhauer *et al.*, 2006; PERA, 2006; Namur, 2007). Focusing on collaboration and taking advantage of the capabilities of cooperating objects poses a challenging but also very promising change on the way future factories will operate, as well as to the way we design software and model their interactions.

15.3 The Cross-layer Service-oriented-architecture-driven Shop Floor

The key issue for achieving business continuity and realize future factories that can promptly respond to dynamic situations is the strong collaboration of the shop-floor system with enterprise and engineering systems. In order to achieve that we have to make sure that an open infrastructure (Figure 15.3) will enable the exchange of information, exposed and able to be consumed as services, and allow also the integration of legacy or isolated sub-systems.

The SOA applicability in industrial automation (Jammes and Smit, 2005) and the collaborative automation system as part of the collaborative manufacturing management (Gorbach and Nick, 2002; Nick and Polsonetti, 2003) are complementary paradigms that result from a multidisciplinary activity, including scientific and technological areas like knowledge management, production control engineering, information and communication technologies, etc. (Kennedy *et al.*, 2008; Candido *et al.*, 2009). In order to realize SOA and collaborative systems, three main activities have to be performed:

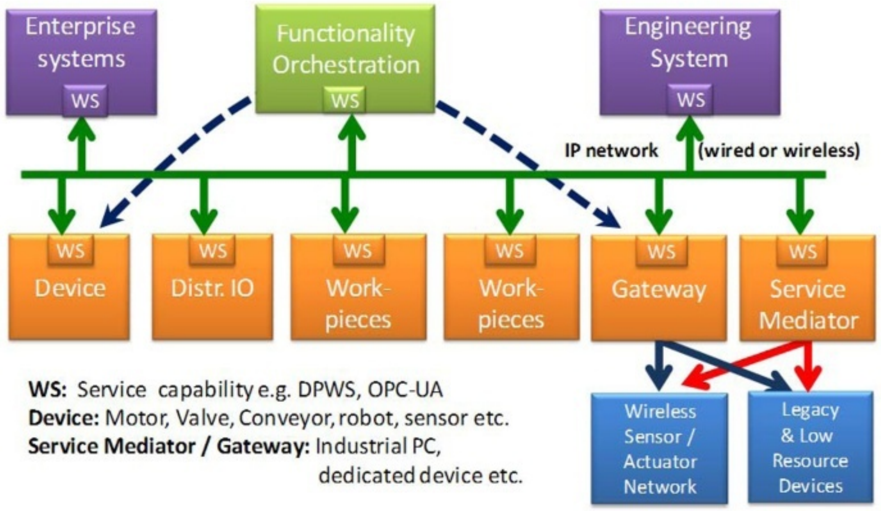

Enterprise systems
WS
Functionality Orchestration
WS
Engineering System
WS
IP network
(wired or wireless)
WS
Device
WS
Distr. IO
WS
Work-pieces
WS
Work-pieces
WS
Gateway
WS
Service Mediator
WS: Service capability e.g. DPWS, OPC-UA
Device: Motor, Valve, Conveyor, robot, sensor etc.
Service Mediator / Gateway: Industrial PC, dedicated device etc.
Wireless Sensor / Actuator Network
Legacy & Low Resource Devices

Fig. 15.3 A shop-floor seen as a cross-layer SOA-based system composed of devices, gateways, tools, engineering systems, and enterprise systems

- Identification of the cooperating entities (systems): the identification of the collaborative automation units that are able to expose and/or consume services, for each production scenario in a defined production domain, e.g., electronics assembly, manufacturing, continuous process, etc. A collaborative unit can be a simple intelligent sensor or a part/component of a modular machine, a whole machine and also a complete production system.

- Building the system of systems: networking/bringing the entities together within an SOA or collaborative infrastructure, i.e., putting the units architecturally together.
- Making the system work for reaching the production goal: collaborative behaviour of the systems for reaching common objectives, i.e., control objectives, production specifications, markets objectives, etc.

The main technical direction is to create a service-oriented ecosystem (SOA-based ecosystem): networked systems that are composed of smart embedded devices interacting with physical, engineering and organizational environments, pursuing well-defined system goals. Taking the granularity of intelligence to the device level allows intelligent behaviour to be obtained on the shop-floor of a factory by composing, aggregating and then orchestrating services offered by configurations of automation devices, engineering systems and enterprise resource planning (ERP). All these systems offer their functions as services (e.g., WS) and are able to introduce incremental fractions of the whole intelligence required for the flexible and agile behaviour of the whole system, within the enterprise/intra- and or inter-enterprise architecture (Figure15.3).

A key idea of the future SOA-based factory is to enable the participation of all factory entities including the legacy devices in the envisioned infrastructure. As of course not all devices will be able to participate, the concept of gateways and service mediators (Figure15.4) that tackle this issue has been developed (Karnouskos *et al.*, 2009).

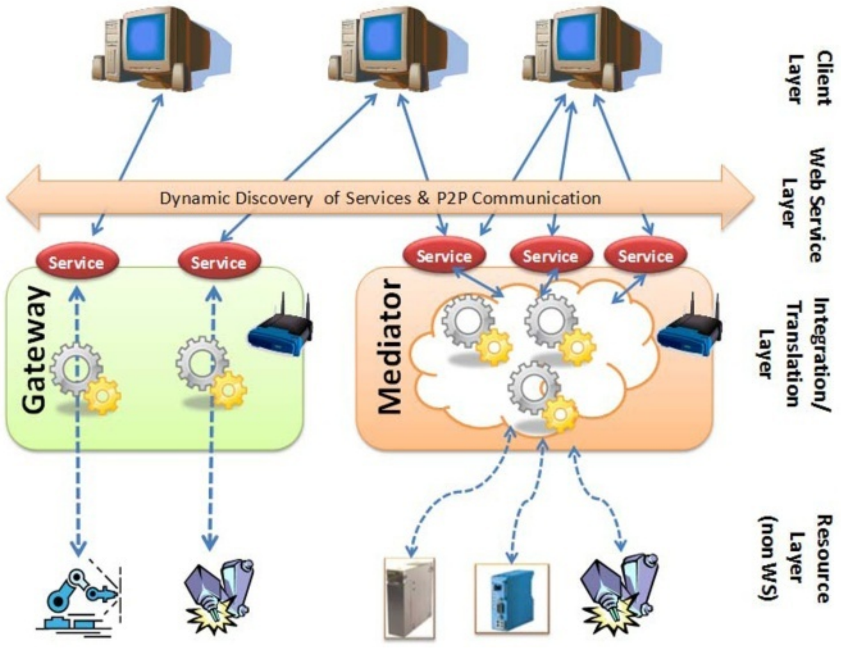


Fig. 15.4 Legacy device integration: gateway and service mediator concepts

A gateway is a device that controls a set of lower-level non-service-enabled devices, each of which is exposed by the gateway as a service-enabled device. This approach allows one to gradually replace limited-resource devices or legacy devices by natively WS-enabled devices without impacting the applications using these devices. This is possible since the same WS interface is offered this time by the WS-enabled device and not by the gateway. This approach is used when each of the controlled devices needs to be known and addressed individually by higher-level services or applications.

The service mediator not only aggregates various services but possibly also computes/processes the data it acquires before exposing it as a service. Service mediators aggregate manage and eventually represent services based on some semantics (e.g., using ontology's). In our case the service mediator could be used to aggregate various non-WS-enabled devices. This way, higher-level applications could communicate with service mediators offering WS, instead of communicating with devices with proprietary interfaces.

The holistic cross-layer SOA approach favours adaptability and rapid (production real-time conditions) reconfigurability, as re-programming of large monolithic systems is replaced by reconfiguring loosely coupled embedded units, across the complete enterprise. From a functional perspective, the main technological challenge is on managing the vastly increased number of intelligent devices and systems and mastering the associated complexity that emerges from the architectural and behavioural specifications of the factory shop-floor. In this sense, from a run-time infrastructure viewpoint, the shop-floor is now considered as a heterogeneous set of a new breed of very flexible real-time embedded devices (wired/wireless) that are fault-tolerant, reconfigurable, safe and secure, and that are exposing their functionality as a set of WS. Auto-configuration management is then addressed through basic plug-in, plug-and-work/run and plug-out mechanisms. This functional perspective must be applied to several types of devices, used in automation systems, automotive electronics, telecommunications equipment, building controls, home automation, telemetry, medical instrumentation, etc.

15.4 Dynamic Reconfiguration of a Service-oriented-architecture-based Collaborative Shop Floor

Dynamic reconfiguration of the shop-floor can be achieved by applying the concepts we introduced in engineering and operation of collaborative automation and production systems (Gorbach and Nick, 2002) composed of distributed, reconfigurable and collaborative service-oriented devices integrated in a cross-layer service-oriented enterprise architecture.

15.4.1 Methodology

The methodology we follow explains the autonomous behaviour of systems composed of service-oriented devices at the shop-floor, able to interact with the information technology (IT) enterprise systems. The resulting engineering approach permits after initial setup the automatic operation of the device and interaction to other devices and to upper IT levels of an enterprise based on, e.g., MES (manufacturing execution system)/ ERP and DMS (decision-making system).

At shop-floor level, the devices have a degree of autonomy in the sense of auto-sustainable control and necessary services to permit lateral collaboration with other devices, requesting/providing decision information from MES/ERP/DMS and integration. All interactions and resource sharing are via service orientation. There is a lose-coupling inheritance in form of bottom-up perspective (from the devices/shop-floor level), enhancing the autonomy and consequent reconfiguration capabilities.

The orchestration is described and implemented by a dynamic model-based component called the orchestrator and is supported by routines to handle undocumented events and decide over present conflict situations that arise due to shared resources, competition and concurrent relationships among the devices, as well as the occurrence of unexpected failures, errors, etc.

It is important to recall here that the operational behaviour of the devices is self-controlled and guided by internal/external events that also may correspond to service calls. Several procedures are defined for the operation behaviour life cycle of these components (see Figure 15.5):

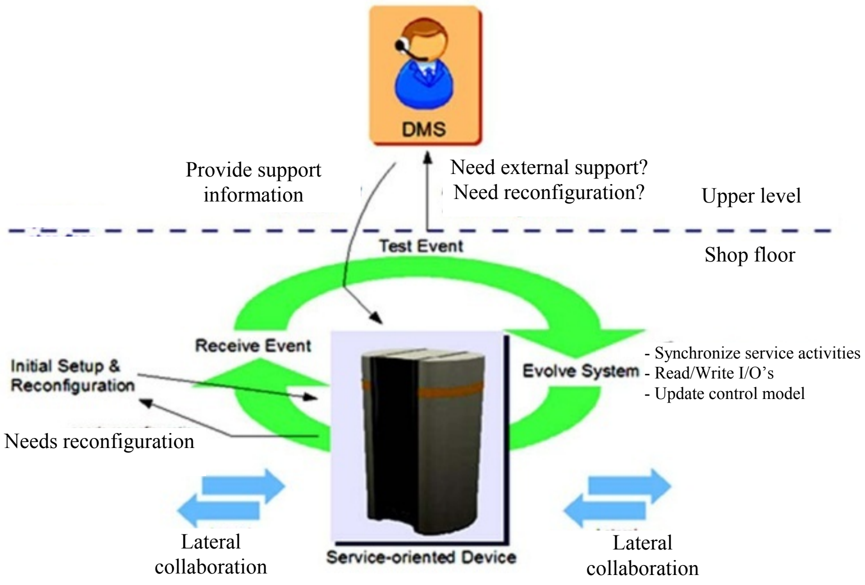


Fig.15.5 Concept of the independent behaviour of autonomous service-oriented devices/components

- Initial setup of the device, including configuration, establishment of connections to other devices/components and putting in a waiting state.
- Events are received via service operations, internal device interface to input/output (I/O) and generated directly by the control (e.g. from conflicts).
- The received event must be tested:
  – If it corresponds to some description of the control model's actual state, then the system is evolved by updating the control model, synchronizing service activities (e.g. interaction with other devices), read/write to I/O and other related tasks.
  – In the case of an exception, undocumented event or an internal conflict, some decision is required. If the device has the necessary information to resolve it, special procedures are taken to interfere in the normal system's control and new events are generated. In the case of not having sufficient control over the event, it may ask for external support (e.g., DMS) to provide more useful information for a concrete decision over the problem.
- After processing the event and evolving the system it is able to receive other events.
- Some events and consequent decisions may result in the requirement of reconfiguration of the control model and other parameters. In this situation, the device should be setup considering the new situation. Note: the reconfiguration time does disable the device, but does not imply the inability of other components/devices to operate (except on heavy dependence).

The application of this method results on autonomous devices that are self-controlled and have less dependency on other components, especially from the upper levels such the DMS. In short, the features of these devices are:

- service orientation;
- autonomous control and consequent behaviour;
- event-based life cycle.

The orchestration (control) of the lifter is defined in a high-level petri net (HLPN) model that shows the global behaviour in the different operation modes (Figure 15.6) exposed as “Services”. Please note that for simplification, only one pallet at a time may occupy the lifter. The blue-colored and larger transitions mean a complex operation (such as a service call) and can be detailed to provide a more in-depth look at the control. They represent the necessary steps to do, e.g., a top-left-transfer-in operation by reading/writing to corresponding I/O and synchronizing the service activity (e.g. a conveyor requests the service). The activation of this transition is done when the lifter is available and when a conveyor asks for the service or a sensor did detect a pallet. Other situations that are not documented can also be handled and need special procedures, as referenced previously. Transfer-out operations (such as the top-right-transfer-out transition) should be done synchronously with connected transfer devices (e.g., conveyor) to be able to provide a smooth transitional movement of the pallet from one device to another one. This requires that the lifter request a transfer in service of the connected conveyor.

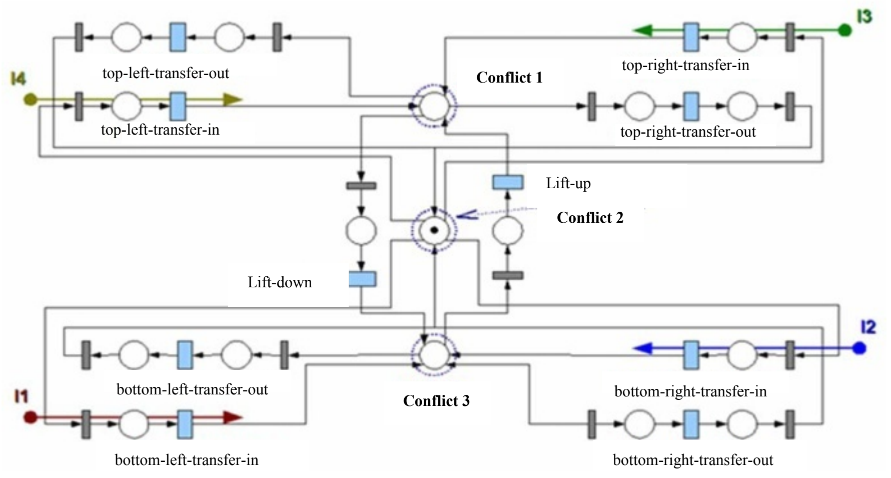


Fig. 15.6 HLPN model of the behaviour of a dynamic orchestrator for the lifter of Figure 15.5

15.4.2 Example

The methodology is applied to a mechatronics device corresponding to a lifter with two levels and four different ports where pallets can be inputted and outputted. These ports should be used to connect to other devices, such as conveyors, but can be triggered manually by placing a pallet in the lifter (since a sensor detects it). Figure 15.7 shows a representation of the lifter with all possible predicted operation modes (that correspond to the paths that a pallet may take).

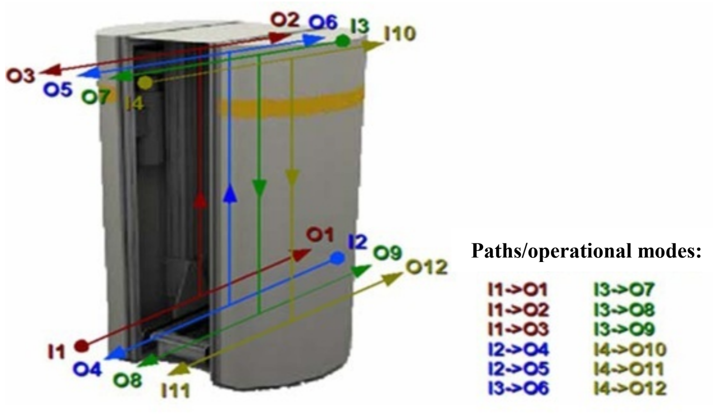


Fig. 15.7 Production device (lifter) with 12 operational modes to be exposed as services

After the initial setup and configuration of the device with the control model and additional routines, the device is available and waiting for events. For example, a connected conveyor is requesting the bottom-left-transfer-in service. In this case, and since it is a documented event in the control model, the device proceeds to evolve the system by running the HLPN model and taking the related actions. After the bottom-left-transfer-in is successfully concluded, the system has to confront an exceptional event that is thrown up by a conflict in the model (namely, conflict 2). If it does not have the necessary information to decide, it must call specialized components to help in the procedure. As depicted in Figure 15.5, DMS are used for this purpose. The lifter sends a request for support to the DMS, including supporting information (the ID of the pallet and possible outputs: lift-up and bottom-right-transfer-out). Based on the workflow of the pallet, a decision is returned in the form of an event to the lifter and now it should be able to resolve to conflict and evolve the system. For example, the lifter receives the suggestion to do a lift-up from the DMS, so it may proceed to do a lift-up. In any case, the final decision is up to the lifter that considers the received suggestion, but may operate differently in the case of internal situations (e.g. occupation of the lifter and/or connected conveyors).

The following topics summarize the initial advantages of the proposed approach:

- autonomous supervisory control and support for individual device reconfiguration without affecting whole system behaviour;
- integration into IT-enterprise and lateral collaboration among devices due to service orientation.

15.5 Analysis Behind the Engineering Methods and Tools

15.5.1 Applying Functional Analysis to Validate Service Composition Paths in High-level Petri-net-based Orchestration Models

Given a factory layout, its structure and predicted behaviour are modelled using HLPNs. A bottom-up approach is used, which consists in:

1. Modelling the behaviour of resources hardware (HW) mechatronics like robots, machines, transport components, etc. The models represent all possible discrete states of such a resource and also all manufacturing functions that this resource is able to expose as services, e.g., move-piece, pick-part, transfer-pallet, etc.
   - Remark 1: the modelling approach generates a set of a resource's discrete states that fulfil basic properties like boundedness, conservativeness, etc. (Silva and Valette, 1989; Feldmann and Colombo, 1998).

- Remark 2: The modelling approach generates a set of a resource's exposed services that fulfil basic properties like repetitiveness, liveness, etc.

2. The models of resources are composed into a "coordination model". At this point, it is possible to speak about a behavioural model of the SOA system, which represents the set of services that are correlated and able to be orchestrated and composed/coordinated following the layout and also the production specifications of the production system.
   This task follows the same rules of configuring a required HW layout, i.e., taking into account competition, concurrency and shared resources behavioural relationships, among others. The result is an HLPN model of the whole factory.
3. Due to the strong mathematical background that is behind the Petri net theory, the models can formally be analyzed (Memmi and Rucairol, 1979; Murata, 1989).
   - Co-related discrete states of composed resources belong to, at least, one state-invariant. The set of "state-invariants" and their linear compositions represents all possible configurations of HW resources that are able to expose a service.
   - Co-related exposed services belong to, at least, one service-invariant. The set of "service-invariants" and their linear compositions represents all possible "service orchestrations paths" that the whole system is able to expose.
4. The coordination model structure represents the kernel of an SOA component identified here as model-based dynamic orchestrator.
5. When the model-based orchestrator is interacting in real-time production conditions with the HW devices, interaction based on the exchange of events and "exposition/calling" services, many known "intelligent supervisory control and automation functions" like model-based monitoring, diagnosis, maintenance control, are intrinsically supported and performed. Note: the application of Petri nets to perform monitoring of flexible production systems has been addressed by (Silva and Valette, 1989; Feldmann and Colombo, 1998 and the references therein). One of the major references to model-based monitoring systems and applications is Du *et al.* (1995).
   - The intrinsically contained information about a resource's states, exposed services, controlled manufacturing process, etc. that is contained in the HLPN model can be classified/recognized as model-based monitoring indexes.
   - The model-based monitoring indexes can/must also be exposed as services.
   - The model-based monitoring services will be used to enhance the feature-based monitoring indexes that are exposed as services by the devices. Feature-based monitoring indexes refer to sensor signals, device parameters like motor velocity, electrical intensity, etc.
6. Having the minimal set of state- and service-invariants, the dynamic orchestration that can be performed by the HLPN-based orchestrator will consist in generating and exposing different compositions of those invariants.
   - Each composition is an orchestration path, allowed by the modelled system.

   - Each composition will be done by weighting the individual atomic services. It is basically a composition of services invariants done according to pre-conditions required to automate the behaviour of the system, i.e., minimal energy-consumption service path, faster throughput service path, etc.
7. Since the system (e.g., resources and factory) and their HLPN model are mathematical (algebraic) vectorial fields (both are dual fields), the service orchestration is respecting all rules that such vectorial spaces have, under the laws of functional analysis (Kreyszig, 1989). Figure 15.8 shows a simplified example.
   - There is an isomorphism between the "composition of service-invariants" coming from the processing/analysis of the HLPN model and the "composition of services" exposed by the resources/factory.
   - The cardinality of the state and service spaces of the real production environment is the same as the places and transitions of the HLPN model.

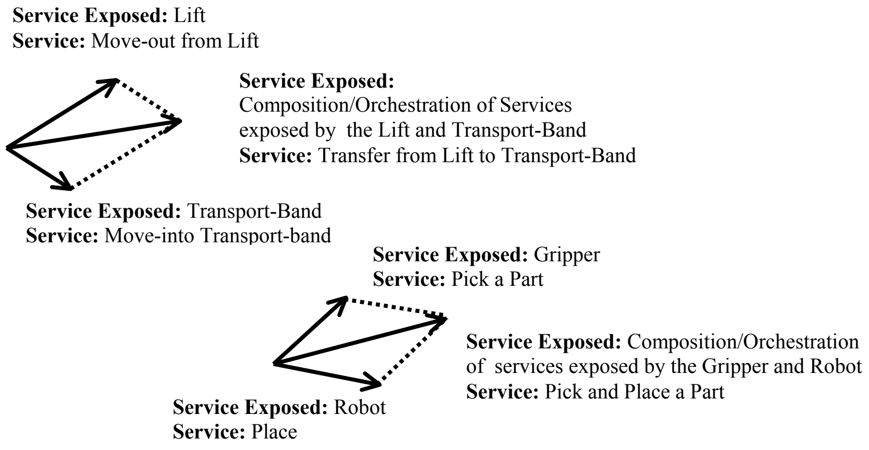

Service Exposed: Lift
Service: Move-out from Lift
Service Exposed: Composition/Orchestration of Services exposed by the Lift and Transport-Band
Service: Transfer from Lift to Transport-Band
Service Exposed: Transport-Band
Service: Move-into Transport-band
Service Exposed: Gripper
Service: Pick a Part
Service Exposed: Composition/Orchestration of services exposed by the Gripper and Robot
Service: Pick and Place a Part
Service Exposed: Robot
Service: Place

Fig. 15.8 Orchestration of services formally expressed as vectorial composition

8. The different HLPN models have to be analysed, before they can be used in the different phases of the life cycle of the SOA-based system. With the results of this analysis, a formal validation of the specification of the SOA-based architecture, structural and behavioural specifications, is performed. If and only if the basic properties of the modelled system are proved, the model can be used as the logic structure of the orchestration of services. Remark: as a matter of fact, and as an example of the high value of this analysis, only if each transition of the HLPN model belongs to at least one transition-flow/invariant, will the corresponding service at some moment be able to be exposed and/or called in the system. In a similar way of thinking, many other relationships will be validated. Figure 15.9 shows a screen-shot of the HLPN-analysis software developed in the project SOCRADES.

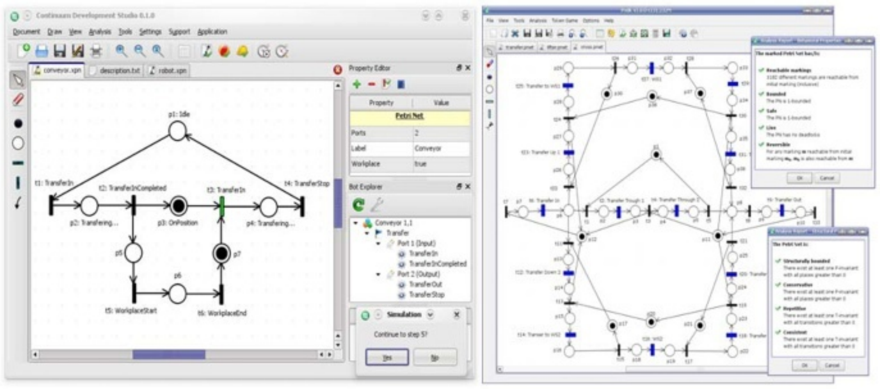


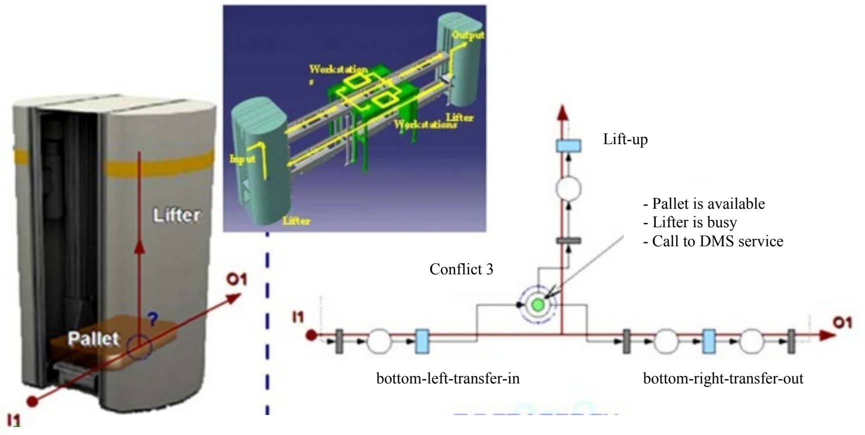
Fig. 15.9 PNDK tool to analyse the HLPN models of an SOA-based shop-floor

15.5.2 Example

1. The lifter depicted in Figure 15.7 possesses 12 operation modes.
2. Each operation mode is exposed as a service, which is the composition of atomic services.
3. Each operation mode corresponds to a transition-flow/transition-invariant or service-invariant in the HLPN model depicted in Figure 15.6.
4. The model-based orchestrator is able to expose three possible conflicts situation, i.e., states that are shared by different orchestration paths. In the language of HLPN models, the situation is recognized as "conflict" between two or more concurrently enabled transitions or transition-firing modes, as shown in Figure 15.10.
5. When the lifter is working, services are called according to an orchestrated service-based production path.
   – From control point of view, each time a service is called, it happens by the firing of a transition of the HLPN model (e.g., top-right-transfer-out in Figure 15.6).
   – A path is performed by following a precise mathematically well-defined service-invariant in the HLPN model.
   – Due to the composibility rule expressed above, at any time in an asynchronous manner (better to say, at any discrete state of the system) it is possible to take the decision for changing the next service. That is, the system follows the originally orchestrated path calling the next service of the path, or it decides to change to another service that is part of one of the minimal service-invariants that have originated the "composed path".
   – In Figure 15.6, after the system starts performing I1, after finishing the "I1-Service" and when the Petri net marking is ready to call the service

<Transfer-Up>, the service represented by O2 is no longer more exposed, e.g., blocked/failed, or the system has to change strategy due to some external conditions. The system offers the possibility to change the sequence initially orchestrated. There are three services that can follow the I1, one is O2 (out of the question), the other two are O1 and O3.

– Following the behaviour of the HLPN-based model-based orchestrator depicted in Figure 15.6, after the service <Botom-Left-Transfer-In> was executed, the conflict 3 is exposed. For the lifter, it is possible to continue with (<Lifter-Up> + <Top-right-transfer-out>) or (<Lifter-Up> + <Top-Left-Transfer-Out>) or <Botom-Right-Transfer-Out>), nevertheless, the first one according to (f.) is no longer exposed.
– The conflict situation exposed as a service in Figure 15.10 needs to be solved. The solution, i.e., the selection of the possible next-called service can be modelled in a static manner or left to an external SOA component that in the SOCRADES project has been called DMS, as depicted in Figure 15.5. Basically, the action of the DMS should be the selection of the HLPN transition or transition-firing mode that has to be fired in the next event (from the set of them that are enabled).
– This selection corresponds to a dynamic change of T-flows and it is what is called here dynamic orchestration.



- **Mechatronic transport device offers transport functions**
- **Operations are exposed as Web Service (Transfer Interface)**
  - TransferIn (int p), TransferOut (int p), TransferStop (), Transfer Completed (Pallet, p), GetStatus (Status s), …
- **Device offers additionally DPWS standard plug'n play functions for description and discovery of Device/Transfer Service**

Fig. 15.10 Conflict behaviour processing by the dynamic orchestrator for the lifter of Figure 15.7

6. From control and monitoring viewpoints, the call of a service/firing of a transition is one step (service) of an orchestration path, i.e., it is possible to immedi-

ately know how many and which are the remaining steps (services) that can be called as next in the current path and also in alternative paths.

7. At any state of the system (represented by the current marking of the HLPN), past, current and future discrete states of resources (individual or composed services) are known and ready to be used for monitoring and other supervisory control functions.

15.6 A Service-oriented-architecture-based Collaborative Production Management and Control System Engineering Application

The approach for creating complex, flexible and reconfigurable production systems is that these systems are composed of modular, reusable and collaborative components that expose their production capabilities as a set of services. This composition approach applies to most levels of the factory floor, where simple devices compose complex devices or machines, which in turn are composed to build cells or lines of a production system and so on (see Figure 15.11).

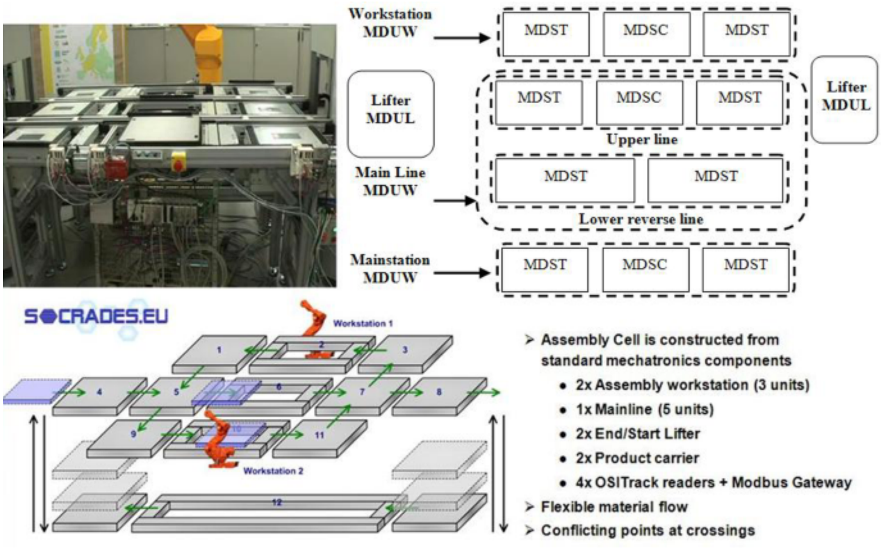


Fig. 15.11 A modular, reconfigurable shop-floor

The same applies to the concept of service-oriented production systems, where the shop-floor components expose their production capabilities as a set of services. Figure 15.12 shows the same layout of Figure 15.11 but identifying the shop-floor components by their “service exposition”. Moreover, some of the shop-floor components are also able to expose complex services built by composing and/or orchestrating simpler services, using their collaborative behaviour.

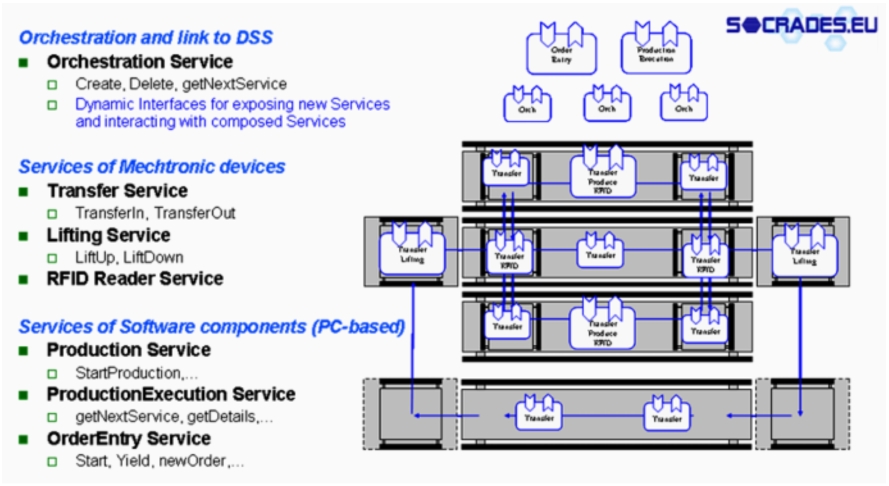

Orchestration and link to DSS
Orchestration Service
Create, Delete, getNextService
Dynamic Interfaces for exposing new Services and interacting with composed Services
Services of Mechtronic devices
Transfer Service
TransferIn, TransferOut
Lifting Service
LiftUp, LiftDown
RFID Reader Service
Services of Software components (PC-based)
Production Service
StartProduction,...
ProductionExecution Service
getNextService, getDetails,...
OrderEntry Service
Start, Yield, newOrder,...
SOCRADES.EU
Order Entry
Production Execution
Orch
Transfer
Transfer Produce RFID
Transfer Lifting
Transfer RFID

Fig. 15.12 A service-oriented view of a shop-floor

Based on the requirements of both physical hardware and the SOCRADES project, it was decided that the basic building blocks that compose the distributed system should be configurable software components assuming different tasks. Therefore, the software components were designated as "bots" (that have a so-called "orchestration engine" embedded inside) and are able (in a service-oriented fashion) to coordinate their activity and proceed also to collaboration processes with other components in the system. To design, configure and maintain bots, there is a need of specific tools that are user-friendly and speed-up the development, using a high-level programming approach (visual languages). These tools are addressed here under the name "Continuum"as depicted in Figure 15.13 (Mendes *et al.*, 2009).

Since services aren't isolated entities exposed by the intervenient software components, there should be some kind of logic that is responsible for the interaction. The modelling language of choice derives from Petri net specifications i.e., HLPN, with high-level extensions, such as time considerations, property system and customizable token game engine. The extensible property mechanism of the HLPN is used as the interface for the configuration of Web service related properties.

Following the approach and structure shown in Figure 15.5, additional requirements are the use of decision support system (DSS) and DMS (Leitao *et al.*, 2008) that are able to provide the correct information each time there are decision points in the executed Petri net model. This DSS is the main interface between the model-based approach for the shop-floor system and the production planning system.

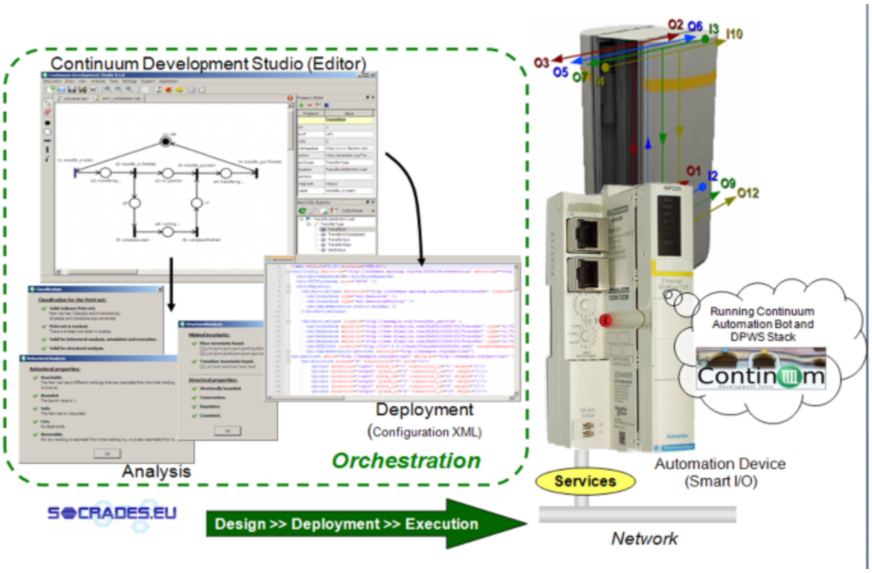

Continuum Development Studio (Editor)
Analysis
Deployment
(Configuration XML)
Orchestration
S●CRADES.EU
Design >> Deployment >> Execution
O3 O5 O7 I4 O2 O6 I3 I10
O1 I2 O9 O12
Running Continuum Automation Bot and DPWS Stack
Continuum
Automation Device (Smart I/O)
Services
Network

Fig.15.13 Petri-net-based orchestration tools and engineering continuum.

Once the models are specified for the mechatronic components or even bigger systems by the Petri net designer, tools are needed for composing systems, creating configuration files and deploying those files to simulators or even embedding the engines and files into real smart automation and control devices. Figure 15.14 shows the case of embedding the orchestration engines and deploying the orchestration models to smart I/O devices (Advantys STB from Schneider Electric).

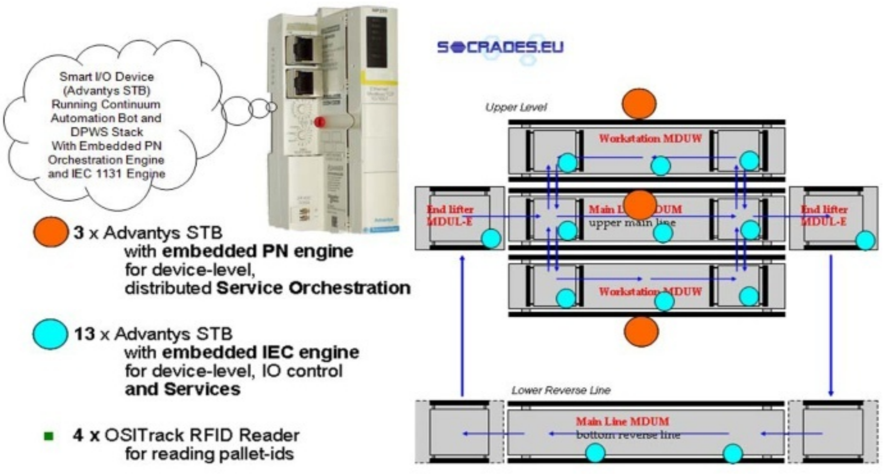

Smart I/O Device (Advantys STB) Running Continuum Automation Bot and DPWS Stack With Embedded PN Orchestration Engine and IEC 1131 Engine
3 x Advantys STB with embedded PN engine for device-level, distributed Service Orchestration
13 x Advantys STB with embedded IEC engine for device-level, IO control and Services
4 x OSITrack RFID Reader for reading pallet-ids
S●CRADES.EU
Upper Level
Workstation MDUW
End lifter MDUL-E
Main Line MDUM upper main line
End lifter MDUL-E
Workstation MDUW
Lower Reverse Line
Main Line MDUM bottom reverse line

Fig. 15.14 A shop-floor system composed of a distributed control system exposing their production control and management functionalities as services

The final result of this engineering phase allows to see the same original service-oriented shop-floor depicted in Figure15.11 as presented in Figure 15.14. That is, a complex flat distributed automation system composed of a set of smart control devices shown in Figure 15.13 wrapping the mechatronics systems with their service view of Figure 15.12.

In order to show the gain in flexibility using SOA, the application was enhanced by performing a change in the production order from the ERP system, directly on the shop-floor (see Figure 15.15).

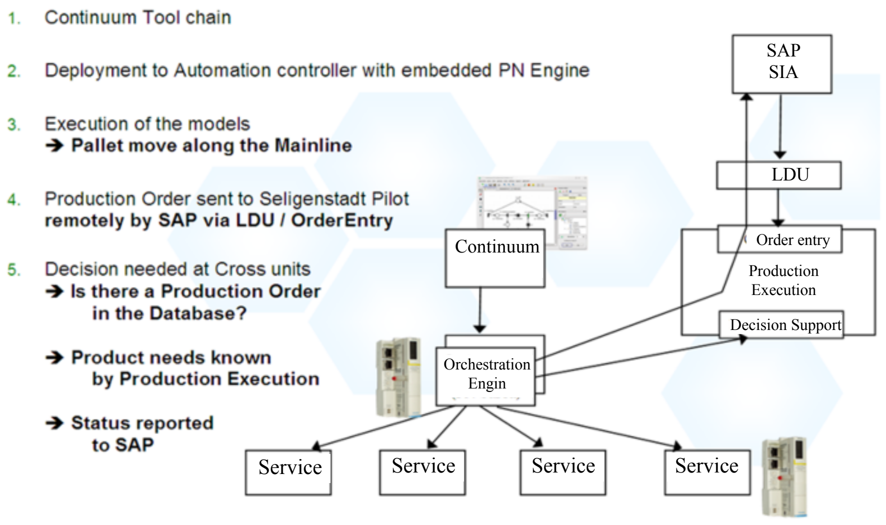

1. Continuum Tool chain
2. Deployment to Automation controller with embedded PN Engine
3. Execution of the models
➔ Pallet move along the Mainline
4. Production Order sent to Seligenstadt Pilot
remotely by SAP via LDU / OrderEntry
5. Decision needed at Cross units
➔ Is there a Production Order in the Database?
➔ Product needs known by Production Execution
➔ Status reported to SAP
SAP SIA
LDU
Order entry
Production Execution
Decision Support
Continuum
Orchestration Engin
Service
Service
Service
Service

Fig. 15.15 Production execution based on collaboration with enterprise systems and local services

Additionally only minimal assumptions about the concrete production line are present in the whole system design. The detailed production steps are stored in the production execution system, which is integrated in between of the Petri-net-based decision support module related to the smart devices and the ERP. The production order system is dynamically discovered and registers new orders in the ERP using the local discovery unit (LDU) of the SOCRADES integration architecture (SIA), further details of which are provided in Spiess *et al.* (2009).

Finally, it was possible to demonstrate the flexible integration and collaboration based on the idea of having a multi-site service-oriented enterprise in which the assembly of electromechanical components is performed in two geographically distributed assembly systems (both of them are similar to the system depicted in Figure 15.11) and production orders could be allocated to different sites. This allocation/re-allocation is done as an evaluation result of the best production facility available at the moment when a production request is made, or when – due to external factors – the production should be shifted to a different location.

The idea, as shown in Figure 15.16, is that similar production facilities are available in remote locations (e.g. Schneider Electric in Germany or Tampere in Finland). The prototypes developed and hosted in Tampere (TUT) and Seligenstadt (Schneider Electric) represent two different companies that are linked with business relations. These companies are “interconnected” via the Enterprise Applications that are hosted in Walldorf (SAP).

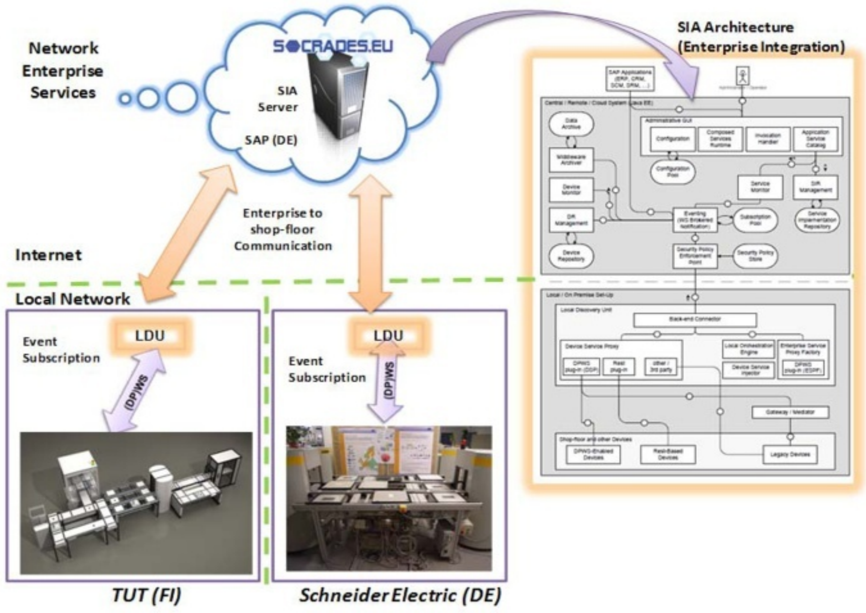


Fig. 15.16 Cross-company collaborative production based on the SOCRADES integration architecture

Both facilities provide electromechanical assembly capabilities using SOCRADES architecture; this means that the components of the production systems in these locations are abstracted and perceived externally as WS. At local level, each one of the facilities acts independently and can coordinate its service-enabled production system by using the SOCRADES tools. At global level, both facilities connect to a service-enabled ERP module provided by SAP, which is used for coordinating the production in the remote locations.

A network application (named LDU) is downloaded and this immediately provides discovery of devices and services (via the device profile for web services-DPWS) on the local network and connection to the backend system (the SIA). Different versions of the LDU can add-up functionalities, e.g., proxy also specific enterprise services at the local shop-floor, where they can be discovered and used by the devices and other services. LDUs provide a means for connecting and managing devices from different premises, without needing virtual private network connections to SAP premises.

The LDUs can discover local services and can interact directly with production execution systems exposed as WS. This is a typical example of hosted functionality on a network server at the provider side, where business services are being implemented/updated and the remote sites (Tampere/Seligenstadt) can interact over the network, with only minimal installations at their side (in order to interact with the business services). In this specific case, the software that interconnects each site is downloaded on the fly over the network.

The different sites involved are collaborating between them via interactions that previously were not possible or would require significant implementation efforts. As can be seen, this is an event-based approach where all sites are notified about the necessary status of the production in the other side, and where the enterprise systems have full visibility on the production and can re-arrange orders in order to meet business goals.

15.7 Conclusions and Future Work

The future factory should be seen as a system composed *of modular, dynamic reconfigurable and service-oriented collaborative systems*. Complex and dynamic mechatronics, control, communication and IT systems interact with each other, in a service-oriented manner, in order to achieve the goals at system-wide but also at local level (Kennedy *et al.*, 2008; Colombo and Karnouskos, 2009). To realize this, timely monitoring and control as well as open communication and collaboration in a cross-layer fashion are key issues.

Modern approaches such as the SOA paradigm when applied holistically can lead to the desired result. By considering the set of intelligent system units as a conglomerate of distributed, autonomous, intelligent, pro-active, fault-tolerant and reusable units, which operate as a set of cooperating entities, a new dynamic infrastructure, a system of systems, that is able to provide a better insight to its components to the higher levels and better react to dynamic business changes can be realized.

This work presented an overview of the engineering approach, methods and tools that have been specified and developed within the European Research and Development project SOCRADES (www.socrades.eu) and shows the results of the first set of successful applications in the area of elechtromechanical assembly systems, extending the concepts to geographically distributed service-oriented production sites.

Among many scientific and technological outlooks, an intensive study should be performed in spreading the application of the service-oriented paradigm in production control and management. This is having in mind that the roles of control and automation technology developers, producers of production components and IT systems is strongly influenced when all their products and solutions are to be integrated into a system of systems viewed/wrapped as services. A possible view could be that depicted in Figure 15.17.

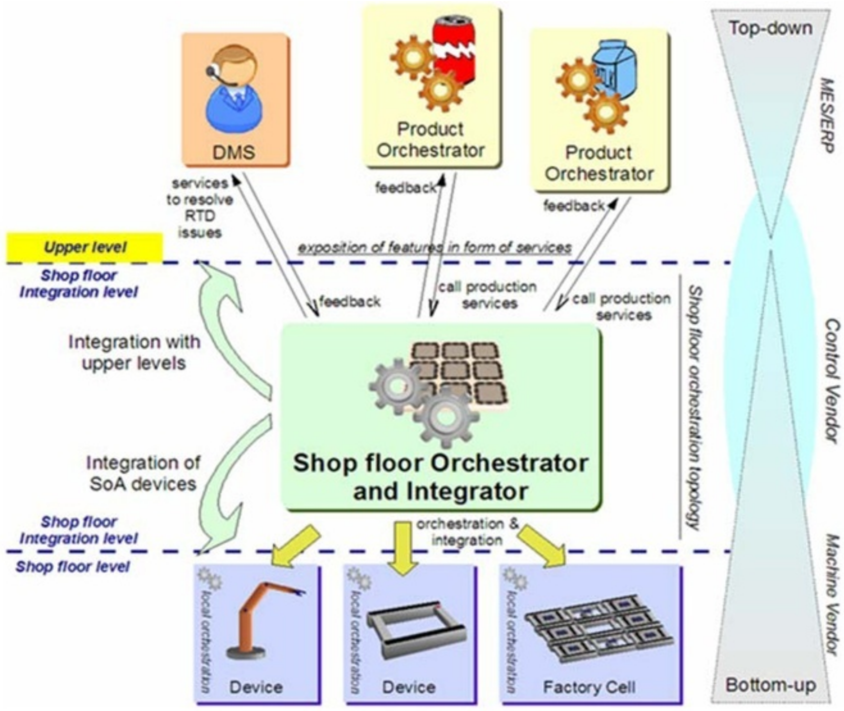

DMS
Product Orchestrator
Product Orchestrator
Top-down
MES/ERP
services to resolve RTD issues
feedback
feedback
Upper level
exposition of features in form of services
Shop floor Integration level
feedback
call production services
call production services
Integration with upper levels
Shop floor Orchestrator and Integrator
Shop floor orchestration topology
Control Vendor
Integration of SoA devices
Shop floor Integration level
orchestration & integration
Shop floor level
local orchestration
Device
local orchestration
Device
local orchestration
Factory Cell
Machine Vendor
Bottom-up

Fig. 15.17 Cross-layer systems integration facilitated by the SOA approach

Having a production system built according to the SOA paradigm, it can be seen as a conglomerate of distributed smart devices and systems able to interact at shop-floor level (horizontal) and in a cross-layer fashion integrating IT systems of the other levels of the enterprise architecture, a clear next step in a technology roadmap and research and development agenda is the realization of supervisory control, automation and production management functions as services.

Acknowledgements The authors would like to thank the European Commission and the partners of the European FP6 project "Service-Oriented Cross-layer infRAstructure for Distributed smart Embedded devices" (SOCRADES - www.socrades.eu) and the European FP7 project "Co-operating Objects Network of Excellence" (CONET – www.cooperating-objects.eu), for their support.

References

Cândido G, Barata J, Colombo AW. et al. (2009) SOA in reconfigurable supply chains: a research roadmap. Eng. Appl. Artif. Intell., 22(6):939–949

Colombo AW, Harrison R (2008) Modular and collaborative automation: achieving manufacturing flexibility and reconfigurability. Int. J. Manuf. Tech. Magmt., 14(3/4):249–265

Colombo AW, S. Karnouskos (2009) Towards the factory of the future: A service-oriented cross-layer infrastructure. In: ICT Shaping The World - A Scientific View, chapter 6, John Wiley and Sons, UK

Du R, Elbestavi, Wu S (1995) Automated monitoring of manufacturing processes, Part 1: monitoring methods. J. Eng. Ind., 117:121–130

Feldmann K, Colombo AW (1998) Material flow and control sequence specification of flexible production systems using coloured petri nets. Int. J. Adv. Manuf. Tech., 14(10):760–774

Fleisch E, Mattern F (2005) Das internet der dinge: ubiquitous computing und RFID in der praxis: Visionen, technologien, anwendungen, Handlungsanleitungen, Springer

Gorbach G, Nick R (2002) Collaborative manufacturing management strategies. White paper, ARC Advisory Group

Jammes F, Smit H. (2005) Service-oriented paradigms in industrial automation. IEEE Trans. on Ind. Info., 1(1):62–70

Jamshidi M (2008) Systems of systems engineering: principles and applications. CRC Press

Karnouskos S, Baecker O, de Souza LMS et al. (2007) Integration of soa-ready networked embedded devices in enterprise systems via a cross-layered web service infrastructure. In: Proceedings of the IEEE Conference on Emerging Technologies and Factory Automation (ETFA), pp. 293–300

Karnouskos S, Bangemann T, Diedrich C (2009) Integration of legacy devices in the future SOA-based factory. In: Proceedings of the 13th IFAC Symposium on Information Control Problems in Manufacturing (INCOM'2009), Moscow, Russia

Kennedy P, Bapat V, Kurchina P (2008) In pursuit of the perfect plant. Evolved technologist, ISBN 978-0978921866

Kreyszig E (1989) Introductory functional analysis with applications. Wiley and Sons

Leitão P, Mendes JM, Colombo WC (2008) Decision support system in a service-oriented control architecture for industrial automation. In: Proceedings of the 13th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA'08), pp. 1228-1235, Hamburg, Germany, 15–18 Sep

Marron PJ, Karnouskos S, Minder D et al. (2009) Research roadmap on cooperating objects. European communities, ISBN 978-92-79-12046-6

Memmi G, Roucairol G (1979) Linear algebra in net theory. Lect. Notes in Comp. Sci., 84:213–223

Mendes JM, Bepperling A, Pinto J et al. (2009) Software methodologies for the engineering of service-oriented industrial automation: The continuum project. In: Proceedings of the 33rd Annual IEEE Computer Software and Applications Conf. (COMPSAC'09), Seattle, USA. 20-24 July

Murata T (1989) Petri nets: properties, analysis and applications. Proc. of the IEEE, 77(4):541–580

Namur (2007) http://www.namur.de/fileadmin/media/Pressespiegel/atp/atp_05_2007_DIN_EN_62264.pdf

Nick R, Polsonetti C (2003) Collaborative automation: The platform for operational excellence. White paper, ARC Advisory Group

PERA (2006) Purdue reference architecture http://pera.net, http://iies.www.ecn.purdue.edu/IIES/PLAIC/

Pfadenhauer K, Kittl B, Dustdar S et al. (2006) Shop-floor information management and SOA. Lect. Notes in Comp. Sci., 4103:237–248

Silva M, Valette R (1989) Petri nets and flexible manufacturing. Advances in Petri nets. Lect. Not. In Comp. Sci., 424:374–417

Spiess P, Karnouskos S, Guinard D et al. (2009) SOA-based integration of the Internet of things in enterprise services. In: Proceedings of the IEEE International Conference on Web Services (ICWS09), Los Angeles, CA, USA, 6–10 July