CHAPTER

4

INDUSTRIAL AGENTS IN THE ERA OF SERVICE-ORIENTED ARCHITECTURES AND CLOUD-BASED INDUSTRIAL INFRASTRUCTURES

Armando Walter Colombo[1,2], Stamatis Karnouskos[3], João Marco Mendes[2], and Paulo Leitão[4,5]

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

[2]Schneider Electric Automation GmbH, Marktheidenfeld, Germany

[3]SAP, Karlsruhe, Germany

[4]Polytechnic Institute of Bragança, Bragança, Portugal

[5]LIACC—Artificial Intelligence and Computer Science Laboratory, Porto, Portugal

4.1 INTRODUCTION

The umbrella paradigm underpinning novel collaborative industrial systems is to consider the set of intelligent system units as a conglomerate of distributed, autonomous, intelligent, proactive, fault-tolerant, and reusable units, which operate as a set of cooperating entities (Colombo and Karnouskos, 2009). These entities are forming an evolvable infrastructure, entering and/or going out (plug-in/plug-out) in an asynchronous manner. Moreover, these entities, having each of them their own functionalities, data, and associated information are now connected and able to interact. They are capable of working in a proactive manner, initiating collaborative actions and dynamically interacting with each other in order to achieve both local and global objectives. New emergent behaviors resulting from the co-operations arise and need to be managed in a smart manner.

Service-oriented architecture (SOA) principles and technologies are considered an adequate backbone to enable the industrial implementation of such collaborative industrial automation and management systems corresponding to the, for example, ISA-95 standard, from the sensor/actuator level through the control devices (Computer Numeric Control (CNC), programmable logic controller (PLC), Robot Controls) and Supervisory Control and Data Acquisition (SCADA) to the Manufacturing Execution System (MES) levels and above. Another very important result of the implementation of the SOA paradigm in the collaborative industrial environment is associated with

the digitalization (virtualization) of the physical environment—i.e., (1) "things in the real world" may get a digital address (get connected to the Internet) and expose their own data and information, and (2) the Internet "things in the cyber world" get real (physical)-world aware.

A first consequence of the digitalization of the industrial environment is that "services" are having a direct physical impact and the real physical world integrates part of the cyber world. A second major consequence is the big amount of machine processable data, servitized functions, generated by heterogeneous data sources located both in the physical and cyber world. Both functions and data, but also information, derived from the data processing and intelligent decision-making systems are offered/exposed as services in both worlds—i.e., the physical world by devices and systems, and the cyber world by a cloud of services. Each entity, located in the physical and/or in the cyber (cloud-based) world, connected into the cyber-physical systems (CPSs) network, is then able to access and consume those services, and also to use these services for generating new ones.

Smartness is intrinsically embedded in an immense set of distributed but networked physical and cyber entities: products, solutions, and services. Major challenges arise when this smartness of such collaborative industrial infrastructures needs to be mastered—i.e., mastering the inherent autonomy of each of the entities and mastering the co-operation capabilities of the networked entities.

The application of the industrial agents paradigm is well-fit to act as an enabler for mastering such collaborative industrial systems. Physical agents following the "Holonic Control" principles (Leitão et al., 2005) are capable of using the information exposed as services in an autonomous manner to perform their own functions and are able to negotiate among them to achieve common goals such as controlling emergent behaviors of the multi-agent community by processing, combining, orchestrating, and composing that data. In summary, both kinds of data sources in a digitalized industrial environment—i.e., physical and cyber (cloud) entities need the support of the agents for fulfilling many of their collaborative behaviors, and for achieving their "common goals."

Although the adoption of service-oriented CPSs is increasingly getting industrial consensus, it should not be underestimated that this kind of system needs connectivity and interoperability with real-time decision systems responsible for supporting the management of the emergent behaviors and timely assessment of the big amount of reachable digital data. On the one side, multi-agent-based real-time decision systems that have been designed for managing emergent behaviors need access to the information/data exposed by the components of the industrial environment. They need the SOA-based cyber-physical infrastructure. On the other side, the functionality and usability of SOA-based industrial CPSs need to be enriched by multi-agent decision-making systems.

In this work, a brief overview of the SOAs paradigm and related technologies that are currently used as a backbone to implement industrial cloud-based CPSs is discussed. Additionally, arguments to consider industrial agents as an unavoidable complementary automation and management system in that CPS industrial environment are analyzed.

The chapter is organized as follows: First, key concepts such as SOAs, cloud systems, and the way they can be used in industrial automation systems is discussed. Subsequently, it is investigated how multi-agent systems (MASs) and SOA principles can be combined to extract the best of the two worlds. Some example use cases are then analyzed, with the first related to a cyber-physical simulation infrastructure using agents and services, and the second one related to a prototype industrial implementation of service-oriented industrial automation systems. The last section rounds up the chapter with some considerations.

4.2 TECHNOLOGIES

4.2.1 TOWARD SOAS

In the last several years, significant efforts have been made (Colombo et al., 2014) to investigate the benefits as well as the impact of emerging technologies, such as SOA, cloud computing, CPS, etc. Industrial agents have long been recognized as a key approach (Leitão et al., 2013) for developing intelligent solutions—e.g., for simulating behaviors, monitoring and autonomously taking decisions in the field, as well as acting as a glue among disparate systems and functionalities.

The SOA paradigm is a way of building distributed systems, originally designed for business systems and electronic commerce, but progressively adopted in other domains. SOA is based on the concept of providing and requesting services. Basically, a service is a software piece that encapsulates the control logic or functionality of an entity that responds to a specific request. In such systems, a provider entity hides its internal structure and functionalities by encapsulating them as services and offering them to the other entities (requesters) by publishing them in a service registry central repository, as illustrated in Figure 4.1.

The list of provided services must be published so they can be discovered by the service requester. Using discovery mechanisms—e.g., UDDI (Universal Description, Discovery, and Integration)—service requesters can find the services they need. After getting information about the available services, the service requester can invoke the execution of those services. More complex services may be created by aggregating the functionalities provided by simpler (atomic) ones. This functionality is referred to as service composition and the aggregated service becomes a composite service (Chafle et al., 2004). The composition of services requires mechanisms for coordination and synchronization and shares many common features with workflow systems. However, service composition requires additional functionalities for discovery and checking the interoperability of the services (Karakoc et al., 2006).

Other concepts, such as service orchestration and choreography, are important for the coordination and composition, and particularly in determining how the services "play" together. Orchestration is

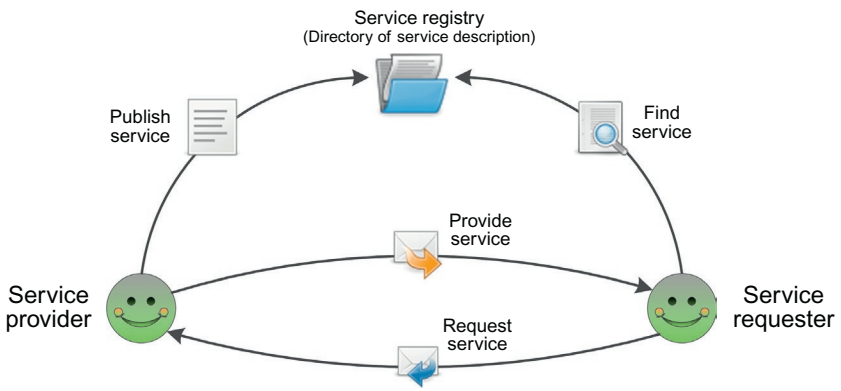

FIGURE 4.1

SOA concepts.

the practice of sequencing and synchronizing the execution of services, which encapsulate business or manufacturing processes (Jammes et al., 2005). An orchestration engine implements the logic for the workflow-oriented execution and sequencing of atomic or composed services, and provides a high-level interface for the composed process. The 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.

Despite the possibility of using other implementation strategies, SOA is commonly implemented using web services (OASIS, 2006). A web service, as defined by the World Wide Web Consortium (W3C), is a software system that supports interoperable machine-to-machine interaction over a network (W3C, 2004). The use of the service-oriented paradigm, implemented through web services technologies, enables the adoption of a unifying technology for all levels of the enterprise, from sensors and actuators to enterprise business processes (Bepperling et al., 2006; Karnouskos et al., 2010).

4.2.2 TOWARD WEB SERVICE-ENABLED DEVICES: DPWS, REST, OPC-UA

Current industrial monitoring and control applications are facing many challenges as the complexity of systems increases and the systems evolve from synchronous to asynchronous. When hundreds of thousands of devices and service-oriented systems are asynchronously interconnected and share and exchange data and information (i.e., services, for monitoring, controlling, and managing the processes), key challenges such as interoperability and real-time performance constraints, among others, arise and need to be addressed. Several Internet-based technologies and concepts have found their way into industrial automation, and especially onto integration of devices (Bangemann et al., 2014). Some of the most widely used constitute the Devices Profile for Web Services (DPWS), Representational State Transfer (REST), and OPC Unified Architecture (OPC-UA).

A standard dealing with ubiquitous device integration is DPWS as described in the OASIS (2009) standard, which is a collection of web service standards. Initially, DPWS was conceived as a successor of UPnP (Universal Plug and Play) for home automation scenarios, but recent works have shown its applicability to the automation world (Karnouskos et al., 2010). DPWS advances previous dynamic discovery concepts, such as Jini (www.jini.org) and UPnP (www.upnp.org) to integrate devices into the networking world and make their functionality available in an interoperable way. DPWS is an effort to bring web services to embedded devices, taking into consideration their constrained resources. Several implementations exist in Java and C (e.g., www.ws4d.org, www.soa4d.org), while Microsoft has also included a DPWS implementation (WSDAPI) by default in Windows Vista onwards and in Windows Embedded CE operating systems. DPWS exists in a number of devices today, and basically brings the SOA world down to the devices, hence extending a fully service-oriented infrastructure down to the physical world and resource-constrained networked embedded systems.

An alternative integration approach is REST, as described by Fielding (2000), which is the architectural principle that lies at the heart of the web and shares a similar goal with integration techniques, such as WS-* web services, that is increasing interoperability for a looser coupling between the parts of distributed applications. However, the goal of REST is to achieve this in a more lightweight and simple manner; therefore, it focuses on resources, not functions, as is the case with WS-* web services. In particular, REST uses the web as an application platform and fully leverages all the features inherent

to HTTP, such as authentication, authorization, encryption, compression, and caching. This way, REST brings services "into the browser"—i.e., resources can be linked and bookmarked and the results are visible with any web browser. There is no need to generate complex source code out of WSDL (Web Services Description Language) files to be able to interact with the service.

Finally, OPC-UA (Mahnke et al., 2009) was developed with the goal to provide a path from the traditional OPC communications model to a SOA. OPC-UA supports a binary protocol for high performance and a web service protocol (e.g., SOAP (Simple Object Access protocol)), which is firewall friendly and uses standard http/https ports. IEC 62541 is a standard for OPC Unified Architecture.

OPC-UA, DPWS, and REST constitute some of the "emerging" technologies and blend with many other traditional ones in the shop floor (Bangemann et al., 2014). The selection of the best-fit technology depends on the scenario and the requirements posed, as at this stage all of them have benefits but also drawbacks (Jammes et al., 2014). Lighthouse projects, such as SOCRADES (www.socrades.eu) and IMC-AESOP (www.imc-aesop.eu), have developed and tested prototypes in industrial settings that use a mix of these technologies to integrate industrial systems (Colombo et al., 2014), as well as couple them with information and business systems (Karnouskos et al., 2010). There are also ongoing efforts—e.g., to further enhance the performance in DPWS with the introduction of Efficient XML Interchange (EXI), as well as integrate more lightweight protocols, such as the IETF Constrained Application Protocol (CoAP), and the fusion of DPWS and OPC-UA (Colombo et al., 2014; Jammes et al., 2014).

All of these efforts that promote modularization and easy integration over heterogeneous infrastructures act as enablers for industrial agents. The latter can be realized both within the device itself, as well as externally, and interact with the devices via well-defined services, as will be analyzed later in this chapter.

4.2.3 CLOUD-BASED INDUSTRIAL SYSTEMS

Future industrial automation systems are expected to be complex system of systems that will empower a new generation of what today would be considered hardly realizable applications and services. The rapid advances in technology during the last years have given rise to virtualization and cloud systems. Virtualization addresses many enterprise needs for scalability, more efficient use of resources and lower total cost of ownership (TCO), to name a few. Cloud computing has emerged powered by the widespread adoption of virtualization, SOA, and utility computing. IT services are accessed over the Internet and local tools and applications (usually via a web browser) offer the feeling that they were installed locally. However, the important paradigm change is that the data are computed in the network but not in *a priori* known places. Typically, the physical infrastructure may not be owned, and various business models exist that consider access-oriented payment for usage (Karnouskos et al., 2014a).

New industrial systems and architectures are being developed to take advantage of the cloud and its services (Karnouskos et al., 2014b). Figure 4.2 illustrates such an effort carried out within the IMC-AESOP project (Colombo et al., 2014). There we see the emergence of an information-based infrastructure that is built in a complementary fashion to the traditional automation "pyramid," as defined in ISA-95. The ever-increasing need for rapid development and deployment of applications and services has taken advantage of the modularization of functionalities and the availability of services at the different traditional automation levels (Level 0 up to Level 4) and combined them in a lightweight application-specific manner.

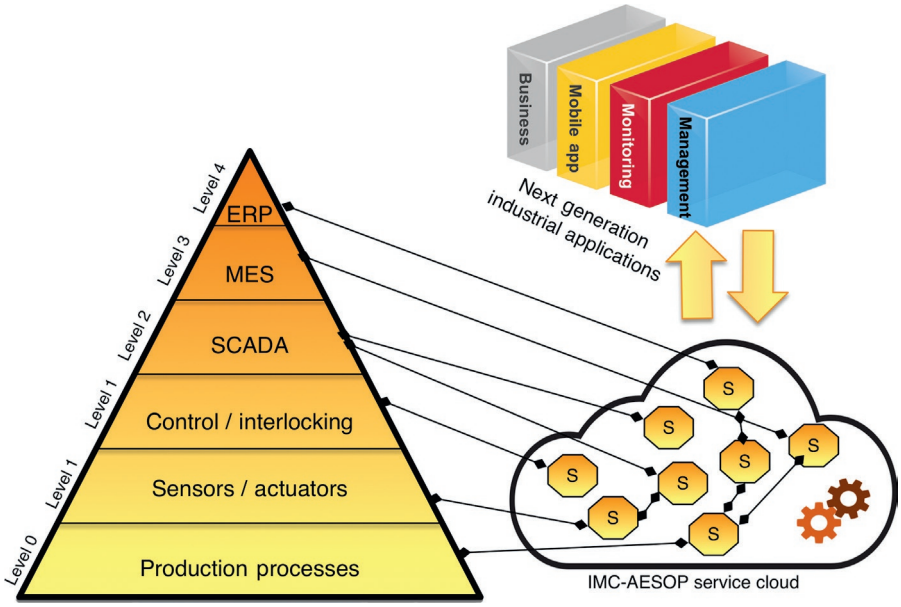

FIGURE 4.2

Industrial automation evolution: complementing the traditional ISA-95 automation world view (pyramid on the left side) with a flat information-based infrastructure for dynamically composable services and applications (right side).

Hence, although the traditional hierarchical view is left untouched, hooks in the form of services enable now the emergence of a flat information-based architecture. Next-generation industrial applications can now rapidly be composed by selecting and combining the new information and capabilities offered (as services in the cloud) to realize their goals. The envisioned transition to the future cloud-based industrial systems is depicted in Figure 4.2.

For industrial agents, such visions and technology trends signal a new era. Industrial agents can very well act as enablers for the servicification of the traditional ISA-95 infrastructure by capturing key functionalities and providing them as services. In addition, they could play coordination roles by orchestrating the integration of various services in the cloud while hosting the intelligence needed.

4.3 BRIDGING AGENTS AND SOA-ENABLED DEVICES

The Internet of Things is prevailing in the industrial domain where devices are acquiring increasingly sophisticated computing and communication capabilities. As such, these are envisioned to play active roles in emerging collaborative infrastructures and systems. Hence, we witness efforts to migrate advanced functionality previously hosted in powerful static back-end systems toward more lightweight mobile distributed embedded devices. Web services nowadays can be implemented directly on devices, providing them with the necessary technology abstraction and making them easily integratable in

heterogeneous environments. Additionally, intelligence can also be realized in various forms including those in the form of agents. In such systems, agents can be integrated within the intelligent device or as an orchestrator at a higher level. Therefore, coupling agents and devices for industrial purposes could yield several benefits.

4.3.1 AGENT AND SERVICE COMMONALITIES

Service-oriented principles can be integrated with MAS to enhance some functionalities and overcome some limitations, namely in terms of interoperability, legacy system integration, and IT-vertical integration. In spite of being based on the same concept of providing a distributed approach to the system, MAS and SOA present some important differences, namely in terms of computational requirements and interoperability, as illustrated in Table 4.1. (Ribeiro et al. (2008) provide a deeper study of these differences.)

These differences highlight the complementary aspects of the two paradigms, suggesting the benefits of combining them to extract the best of both worlds: the intelligence and autonomy provided by MAS solutions and interoperability offered by SOA solutions (Huhns, 2002). This suggestion is not new since services are already part of the agents' specification (e.g., included in the Foundation for Intelligent Physical Agents (FIPA) specification (FIPA, 2002)), and agents are also present in standard documents of SOA methodologies (e.g., in the OASIS (2006) standard). However, the under-considered elements (services in MASs and agents in SOA) are vaguely defined and have a more passive and customized role.

4.3.2 APPROACHES TO COMBINE AGENTS AND SERVICES

Traditionally, the combination of MAS and SOA paradigms can be performed in different ways, as illustrated in Figure 4.3 (Mendes et al., 2009a). The first traditional option, illustrated in Figure 4.3a,

Table 4.1 Differences Between MAS and SOA

Multi-Agent Systems	Service-Oriented Architectures
Well-established methods to describe the behavior of an agent	Focus is on detailing the public interface rather than describing execution details
Agents denote social ability regulated by internal or environmental rules	Social ability is not defined for SOA
Most implementations are optimized for LAN use, but Internet is also possible	Supported by Web-related technologies and can seamlessly run on the internet
Reactive to changes in the environment	Reconfiguration often requires reprogramming
Interoperability heavily dependent on compliance with FIPA-like standards	Interoperability assured by the use of general-purpose Web technologies
Heavy computational requirements	High performance without significant interoperability constraints

Adapted from Ribeiro et al. (2008)

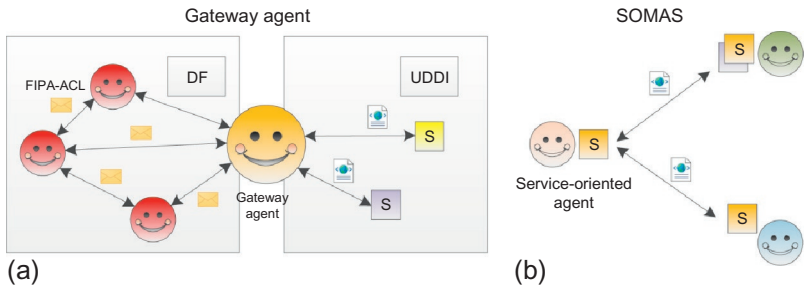

FIGURE 4.3

Common approaches for integrating SOA and MAS.

considers gateways to translate the semantics from the agent world to the services world. According to the FIPA specifications, this task is basically performed by translating:

- Service registration: DF (Directory Facilitator) ↔ UDDI
- Service description: agent service ↔ WSDL
- Message: ACL (Agent Communication Language) ↔ SOAP

An example is the Web Services Integration Gateway (WSIG) plug-in provided by the Java Agent Development (JADE) framework to offer an implementation of the concept of gateway (Bellifemine et al., 2007). This plug-in, in the form of a gateway agent, was implemented by Whitestein Technologies and allows transparent and bidirectional transformations between FIPA-compliant services and web services, employing the WSDL/SOAP/UDDI stack (i.e., publishing agents' capabilities as web services used in a SOA environment). The communication between the WSIG Gateway Agent and the other agents use FIPA-ACL, as illustrated in Figure 4.4, and the service discovery is performed by using two repositories: DF (for the agents world) and UDDI (for the services world). The discovery transformation performed by the gateway agent allows agents to perform service discovery in the web services registry using the UDDI and lets web service clients perform service discovery in the MAS registry using the DF.

Other similar examples are the WS2JADE (Nguyen and Kowalczyk, 2007) and AgentWeb Gateway (Shafiq et al., 2005). Several applications combining MAS and SOA principles employing the concept of gateway agents are reported in the literature. For example, Jacobi et al. (2010) use a model-driven approach that combines SOA and MAS to model a segment of a production chain in the steel industry, and Fayçal et al. (2010) propose the integration of legacy systems by the encapsulation of its features by agents. Another idea is to join the subscribing directories from the agent side (DF) from the web services side (UDDI) in just one common place named UD^3 (Cheaib et al., 2008).

Utilizing the described approach, the design of truly service-oriented MASs is far from the real expected potential and benefits, because the combination is only focused in the communication perspective offered by SOA approaches, and it does not fully explore the potential of designing the system using service-orientation. Another option, illustrated in Figure 4.3b, was introduced by Mendes et al. (2009a) and is characterized by the use of a set of autonomous agents that use the SOA principles (i.e., oriented by the offer and request of services) to fulfill industrial system goals. The achieved

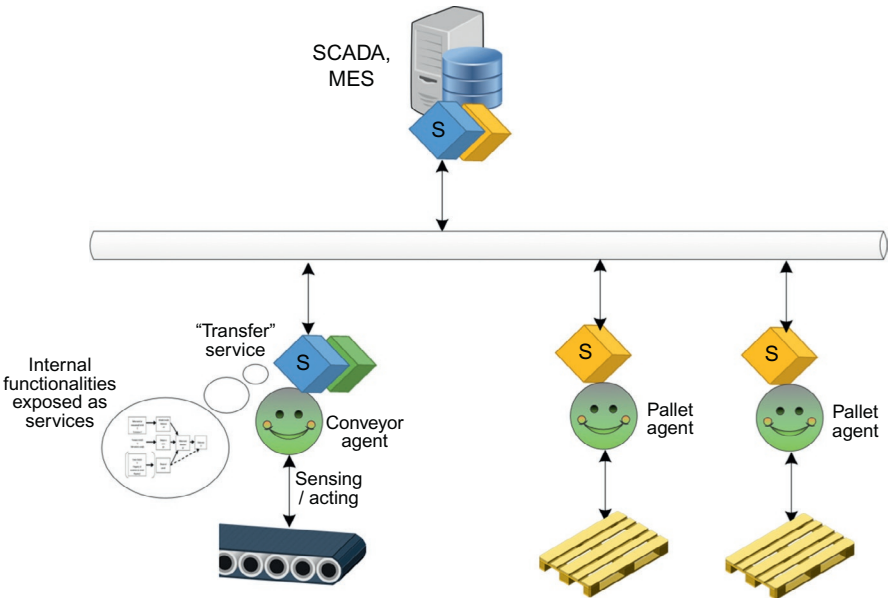

FIGURE 4.4

Example of a service-oriented multi-agent system.

service-oriented multi-agent systems (SOMAS) approach is different from the traditional MAS mainly because agents are service-oriented—i.e., according to Mendes et al. (2009a):

- Agents share services as the major form of communication among agents.
- Individual goals of agents may be complemented by services provided by other agents.
- The internal functionalities of agents can be offered as services to other agents.

An important note is that these service-oriented agents do not only share services as their main form of communication, but also complement their own goals with externally provided services.

An example of using the SOMAS approach is illustrated in Figure 4.4, where devices represent conveyors (transporting pallets) and pallets, and have associated service-oriented agents that are responsible for part of their environment (Leitão, 2012). The conveyor agent provides a service, called the transfer pallet, which encapsulates its internal functionality of transferring the pallet from the input location to the output location. Therefore, it has the ability to read the sensors, execute the embedded logic control and send commands to the actuators of the conveyor. This service is published in the Service Registry to be discovered by other agents representing devices—e.g., conveyors or pallets.

Other neighbor devices (e.g., a pallet agent that needs this transfer service to accomplish its goals) may request the service to the conveyor agent. However, to complete the execution of the service and also to respect global objectives, the conveyor must request an availability service from the next transport unit or workstation connected to its output, using the SOAP protocol. This can be seen as the form of collaboration among the service-oriented agents in the system.

4.3.3 ENTERPRISE SERVICE BUS-BASED SOLUTIONS

SOA-based systems can be realized by an Enterprise Service Bus (ESB) that provides a layer on top of an implementation of an enterprise messaging system (Ziyaeva et al., 2008), acting as backbone for supporting the interoperability among the connected software applications. Typically desirable capabilities of ESBs include, without being exhaustive, process orchestration (typically via WS-BPEL), protocol translation, hot deployment, versioning, life-cycle management, and security. The use of an ESB constitutes an alternative way to implement the integration of MAS and SOA following the SOMAS concept, where software applications are MAS-based systems that are interacting through the use of the ESB by exposing and consuming services.

An example of the use of this approach to integrate MAS and SOA paradigms is provided by the EU FP7 Adaptive Production Management (ARUM) project (arum-project.eu) that addresses the development of solutions to handle emergent challenges in ramping up production of complex and highly customized products, such as for the aircraft industry, and particularly mitigation strategies to respond faster to unexpected events and intelligent decision support systems for planning and operation (Marín et al., 2013).

Aiming to achieve a full interoperability across the entire ARUM solution, traditional ESBs—e.g., the open source JBoss ESB (Jboss, 2014) and the proprietary TIE Smart Bridge (TSB, 2014)—are enriched with a plethora of advanced modules and functionalities that support the tool's life cycle from creation time until they are unplugged from the system, resulting in an intelligent enterprise service bus (iESB). Examples of such modules are the Ontology Service, Data Transformation Service, Sniffer, Node Management, and Life-Cycle Management. The iESB provides a common infrastructure for the integration of heterogeneous agent-based planning and scheduling tools, and legacy systems using the services principles, as illustrated in Figure 4.5.

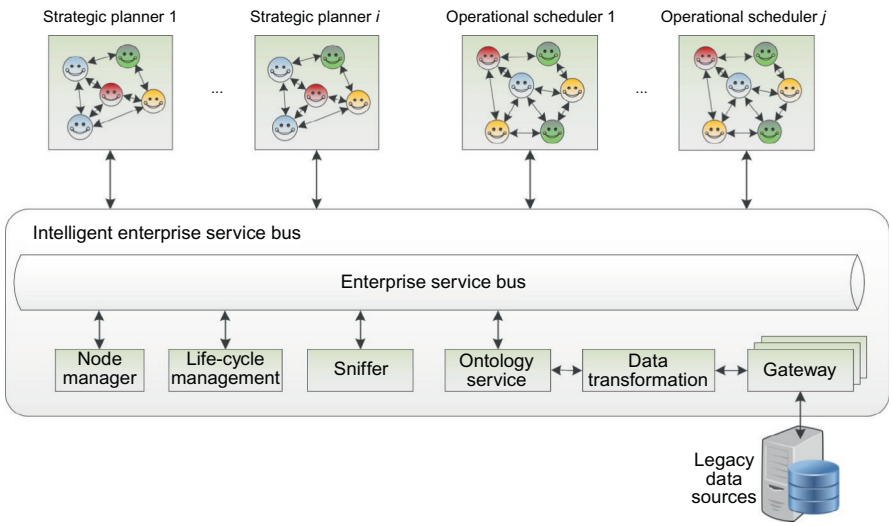

FIGURE 4.5

Integration of MAS and SOA using an Enterprise Service Bus.

The plugability of the agent-based tools is facilitated by the exposition of their functionalities as services and by the use of the ontology services for the representation of the shared knowledge, improving the interoperability in such distributed and heterogeneous systems.

4.4 USE CASE: CYBER-PHYSICAL INFRASTRUCTURE SIMULATION BY COUPLING SOFTWARE AGENTS AND PHYSICAL DEVICES

Today, we see the emergence of cyber-physical infrastructures composed from a high number of heterogeneous devices. The latter may as well be SOA-enabled devices on the basis of technologies such as OPC-UA, DPWS, and REST, as we have already discussed. However, in order to study large-scale systems, the development of real testbeds with hundreds or thousands of such devices is costly. Hence, a compromise might be to simulate their behavior as realistically as possible. Simulating an infrastructure populated by a high number of web service enabled devices is not trivial, but it could provide a very useful tool in the hands of enterprise application developers.

Coupling agents with such physical devices could provide an interesting approach for investigating some of these aspects, including management and network aspects. An architecture for such a simulation is depicted in Figure 4.6 (Karnouskos and Tariq, 2009).

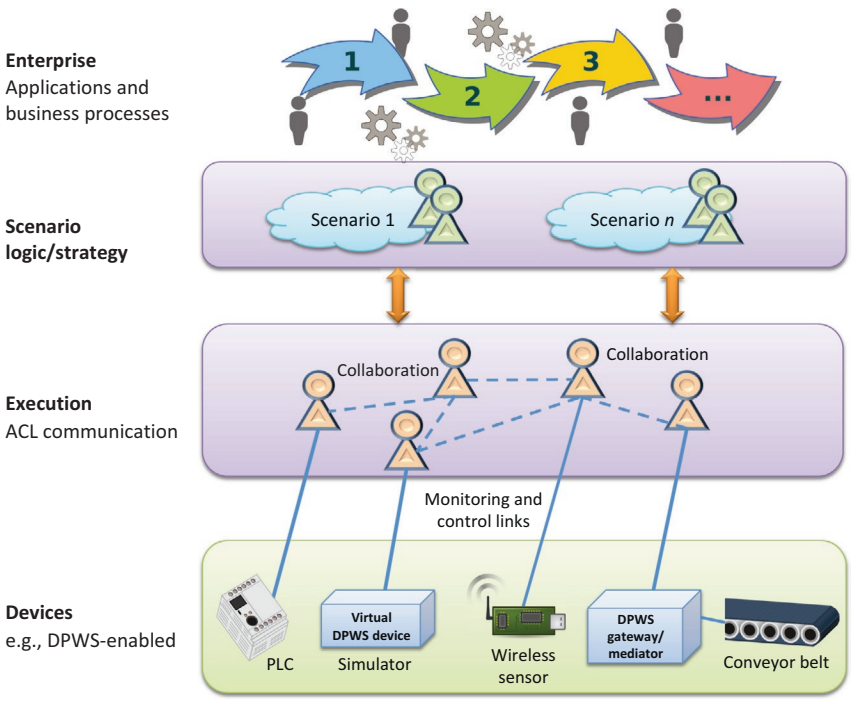

FIGURE 4.6

A simulator of CPS infrastructures relying on agent-driven integration.

The devices at the lowest layer make available their functionality via web services, while a subscription can be made to their services. The device layer consists of devices that directly implement web services—e.g., via the DPWS protocol, and/or via the DPWS gateway (due to resource constraints, etc.). Typical examples of such devices that implement web services (SOA-ready) are PLCs, robots, advanced wireless sensors (e.g., SunSPOTs etc.), and examples of devices connected via a DPWS gateway, which could be Radio-Frequency IDentification (RFID) tags that connect via an RFID reader that acts as a DPWS gateway.

At the execution layer, the mobile agent system hosts several agents that not only cooperate but also control the created virtual devices. One layer higher is the logic, which describes the scenarios that users run within the simulator. The scenarios range from simple ones running standalone, up to complex ones which may start other simpler scenarios first. Finally, at the enterprise layer, various services and applications can communicate via web services with the devices, both real and simulated ones.

For the implementation, the JADE multi-agent platform (Bellifemine et al., 2007) is used to create the agents representing DPWS devices. Each agent represents one DPWS device, which needs to be created using the DPWS toolkit (www.soa4d.org). This integration has been achieved by creating two types of agents interacting with the DPWS toolkit: (i) a DPWS Client Agent (DC-Agent); and (ii) a DPWS Server Agent (DS-Agent), as analyzed in detail in Karnouskos and Tariq (2008).

The DC-Agent implements the client part of the DPWS toolkit, acting as a client for consuming services offered by devices, as well as for services offered by DS-Agents. This agent acts as a bridge between a device and a DS-agent offering service(s) to applications. Tasks assigned to the DC-Agent include discovery of other in-network DPWS-enabled devices, the acquisition of services and data offered by those devices, the processing of data, and the exposition of data to other applications via the DPWS protocol.

The DS-Agent implements the server part of the DPWS toolkit and is more complex because it consists of two distinct components: a server and a service. The server part instantiates the services, registers them, and listens at the specified port for the client requests. The service part is exposed to the external world and handles all the client requests.

As can be seen in Figure 4.7, simulated and real DPWS-enabled devices can be discovered by third-party DPWS clients. These appear as normal devices (distinguishable only by their name), and coexist

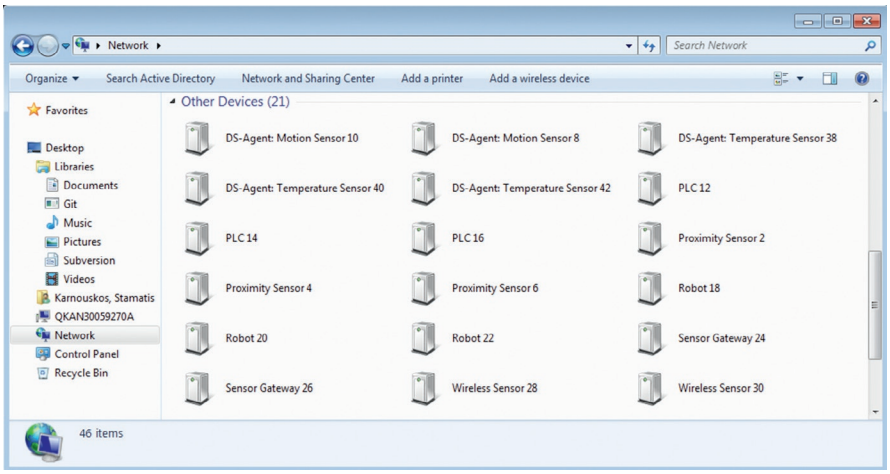

FIGURE 4.7

DS-Agents and DPWS devices discoverable in Windows.

with other devices such as a robotic arm, a SunSPOT sensor (www.sunspotworld.com), and a windows computer. This makes it obvious that the simulator-created devices can at least be discovered/used by other infrastructure actors in an agnostic, non-intrusive way.

The simulation environment consists of a basic set of agents, each of which has its goals and internal logic (Karnouskos and Tariq, 2009):

- *Management Agent*: Tasks of this agent include the evaluation of user arguments, the creation of other agents and other management functions (e.g., logging).
- *Device Explorer Agent*: This agent is based on the concept of the DC-Agent with the aim to discover all the DPWS-enabled devices in the network based with a specific scope.
- *Device Generator Agent*: The core function of this agent is to receive and execute requests towards creating and initializing service agents that simulate a specific service.
- *Scenario Agent*: This agent is specific for each scenario because it executes its strategy/logic.
- *Service Agent*(*s*): The design of a service agent is based on the DS-Agent model. Such types of agents simulate a DPWS service and are visible to the external world via the DPWS communication.

Using the capabilities of the simulator, thousands of DPWS devices were instantiated and investigated (Karnouskos and Tariq, 2009). However, limitations in the hosting computer(s) played a role, and potentially these results can be revisited with more powerful hardware, larger distribution of the agents (e.g., in the cloud), and more efficient implementations of the DPWS toolkit.

The agents played various key roles in this system. First, they acted as "glue" that serviced physical devices and exposed their capabilities via web services, and more specifically the DPWS protocol. As such, any "legacy" or other non-SOA devices could now be easily integrated via web services. The agents also acted as simulation scenario orchestrators, holding the intelligence needed to execute the simulation. As such, we can witness a diverse utilization of their capabilities and some potential roles they can play in industrial settings.

4.5 USE CASE: SERVICE-ORIENTED INDUSTRIAL AUTOMATION SYSTEM

The European research project SOCRADES had explored the application of service-orientation and web services for the next generation of industrial automation systems. In particular, an engineering framework for the development of service-oriented automation systems was introduced by Mendes et al. (2008), using the Petri nets formalism as a unified tool for the specification, modeling, analysis, and execution of service-based automation systems. Petri nets are also exploited as the form of orchestration and composition in service-oriented automation systems.

The application scenario used to demonstrate the SOA approach was a dynamic assembly system based on a customized and modular factory platform for the light assembly, inspection, test, repairing and packing applications as shown in Figure 4.8, left part.

The SOA-based prototype comprises a flexible production system with two work stations (that can be used by operators and robots), several conveyors that route production pallets into/out of the system and to the workstations, and also two lifters. Schematic depicted on the right side of Figure 4.8, the central part of the transfer system (C1-C9) is made of nine transfer units (conveyors) of 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 axis, but also in the transversal axis. The lower

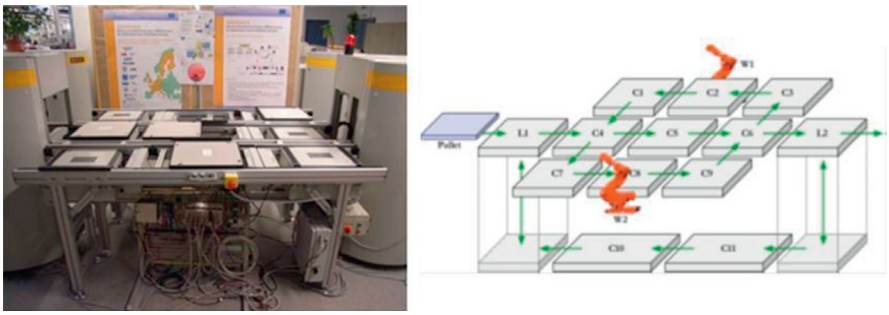

FIGURE 4.8

Prodatec/FlexLink DAS 30 used for the SOCRADES demonstration (located at Schneider Electric Automation GmbH in Germany).

transfer units (C10, C11) have the same behavior as the normal unidirectional transfer units (such as unit C5), but are physically longer. Lifter units (L1 and L2) are responsible for the interface between the upper and lower part of the system, and also for transferring pallets into and out of the automation system.

The pallets enter in the system via the unit C4 and are conveyed using alternative paths to the two workstations W1 and W2. The routing is done at the transfer units based on the required production operations needed by the product mounted on a particular pallet and based on the location and availability of production services in the system (at W1 and W2). A workstation can provide more than one type of production operation, and one kind of production operation could be provided by more than one workstation. The units C4, C6, C2, and C8 are equipped with RFID which is able to read/write information from/to tags attached to the pallets.

A composition approach applies to most levels of the factory floor; simple devices compose complex devices or machines, which in turn are composed to build cells or lines of a production system, and so on. The same applies to the concept of service-oriented production systems and composing complex services from simpler services, complemented with orchestration engines, as illustrated in Figure 4.9. As a matter of fact, the orchestration engines will be located (embedded) into selected devices and their orchestration/composition functionalities exposed from the devices or directly from the service bus, considered here as the service recipient of the service cloud. Note: Orchestration engines appear where atomic services discovered in the service bus have to be composed or orchestrated to generate new services or to manage and control the results of service compositions.

Since services are not isolated entities exposed by the intervenient software components, a kind of logic that is responsible for the interaction is needed. This SOA-based function is depicted by the block "Orch" in the Figure 4.9.

As a matter of fact, the Orch-component of the SOA-architecture is an engine developed and implemented to compose services and to generate high-level functionalities that are results of those service compositions. In the case of a model-based orchestration engine, it is able to interpret a given work-plan made of services (an orchestration) and execute it. The work-plan can be defined in Business Process Execution Language (BPEL) as defined in OASIS (2007), Petri nets formalism e.g., Hamadi and Benatallah (2003) and Bing and Huaping (2005), or even in adapted IEC 61131-3 languages, beside others.

The modeling language used in the EU FP6 SOCRADES work derives from Petri net specifications, including time considerations, property system and customizable token game engine. The developed

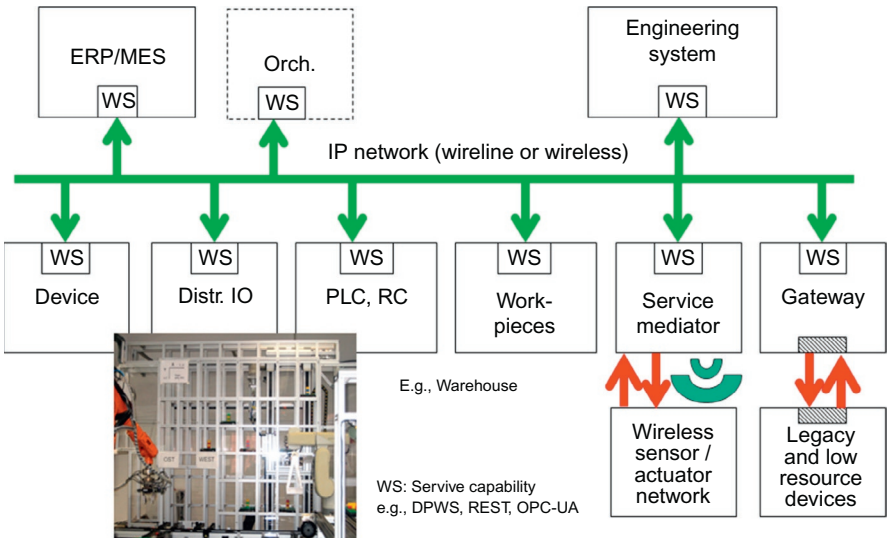

FIGURE 4.9

Important elements of the service-oriented automation system.

Petri net orchestration engine needs to know how and when to respond to services and to represent them in the model. This is done by describing transitions in the Petri net model. A transition willing of sending a request/response or an event must be enabled, and the action is done when it fires. In the other hand, a transition receiving a message from a request, response or event, will only fire if it is enabled and the message is there.

The information to be used by transitions is gathered by an imported WSDL file that contains the description of the service. Depending on the operation, transitions can be part of a client request/response, server request/response, client event and server event. The first two types require two transitions: one for initializing the request and one for the response. It is also possible to test responses by their return parameters, implying the use of one response transition for each test. The difference of an operation being a server or client is obvious: a server waits for the request and then gives a response, and a client makes a request and waits for a response. Events are possible as client and server, but only require one single direction (and consequently, one transition for each test. The difference of an operation being a server or client is obvious: a server waits for the request and then gives a response, and a transition).

The fully distributed service-based automation system with its associated service ecosystem for the case study addressed in Figure 4.8 is represented in Figure 4.10. The atomic services are exposed by the transfer units (Transfer), lifters (Lifting) and RFID devices (RFID) through the smart embedded I/O units (STBs and gateways). These services are the building blocks for the construction of more advanced production automation scenarios, so that they can be associated and composed depending on the requirements and objectives of the application.

The approach for creating complex, flexible and reconfigurable production systems is based on a network of modular, reusable entities that expose their production capabilities as a set of services. Data and information associated to industrial equipment, i.e., physical entities like a warehouse unit, a lifter

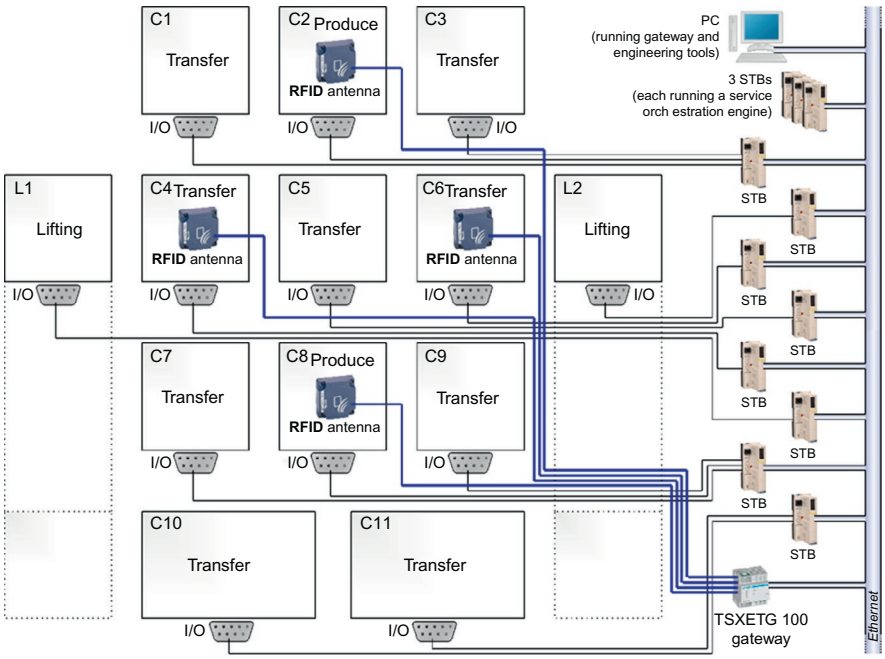

FIGURE 4.10

Service landscape and fully distributed SOA-based automation system.

or a robot, as shown in Figure 4.11, are digitalized by smart embedded devices and exposed as services into a cyber-infrastructure such as a "Service Bus" (cloud of services).

The next major task during the engineering development process of the SOA-based automation system is to fit the automation bot, including the orchestration engine and web service technology into an automation device. The resulting smart embedded device hosts most of the services exposed in the system and is also responsible for the coordination and control of the mechanical parts of the mechatronics system, as represented in Figure 4.11. As one of the first industrial prototypes, the Telemecanique Advantys STB (Small Terminal Box) NIP2311 prototype devices were used in the case of the EU FP6 SOCRADES project, which provide two main interfaces: mediating the automation equipment via input/output modules and managing the access to the service bus by exposing and requesting services (using the Ethernet network interface module). Atomic services representing resources and functions of the connected equipment are provided by the device interface. Some of them may include an orchestration engine to "link" services together and create new composite services. An internal decision support system is responsible for sustaining the engine for decisions (e.g., selecting the best process based on the decision criteria).

The controller of the Ethernet module is used to host the service infrastructure, based on the SOA4D implementation of DPWS (forge.soa4d.org), allowing the deployment of user-defined applications as DPWS-compliant service components. The services are implemented by the STB with an embedded IEC-61131 engine. The ControlBuild prototype developed by Geensys (www.geensys.com) is used to specify the logic and services offline and then deploy those into STBs. Another STB prototype has been

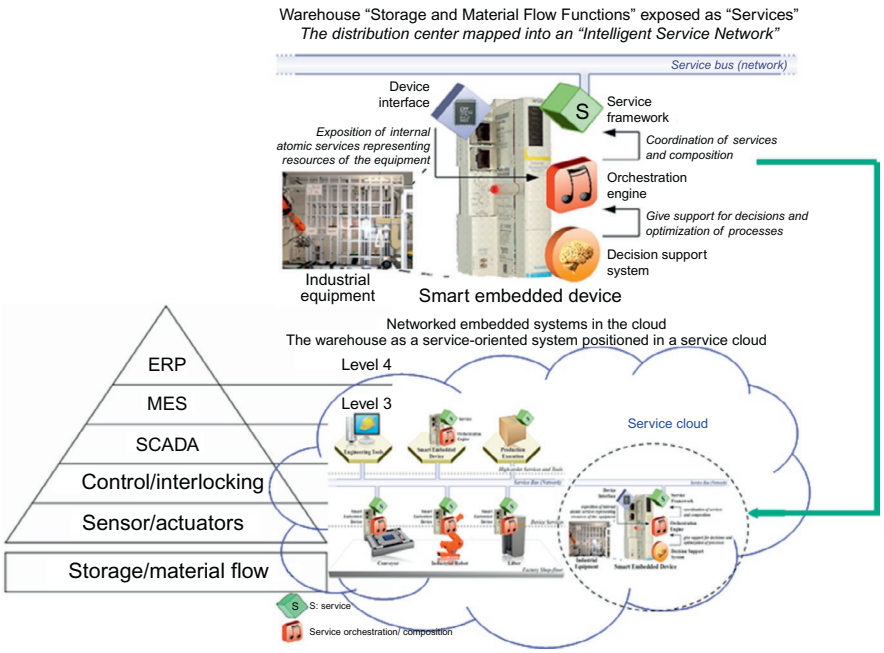

FIGURE 4.11

Implementing a SOA-based automation system with smart embedded devices using industrial Advantys STB NIP2311 components.

implemented that provides an embedded service orchestration engine based on the Continuum Bot Framework with Petri nets kernel (Mendes et al., 2009a) and the DPWS stack with the same deployment mechanisms as for the STB with IEC-61131 engine. The orchestration engines run on their own STBs and provide composed services to the system.

The control logic is managed by the Petri nets kernel module that interprets a given Petri net model (Mendes et al., 2009b). During the execution of the behavioral models, some decision nodes may appear, requiring their real-time resolution. In the case of Petri nets to represent the system behavior, this detection is performed with the identification of marked places that can evolve into more than one alternative way: the marked places that have connected more than one enabled transition. As illustrated in Figure 4.12, the place p1 constitutes a decision node because there are three alternatives to evolve the model—the operation service can be performed using three distinct machines. The decision point is translated in the Petri net model as a conflict, making it necessary that someone, in this case a decision support system, resolves the conflict—that is, selects one of the machines depending on various criteria.

The degree of complexity associated with the decision support system can range from simple algorithms to complex cognitive systems, making the use of agents a natural option in providing intelligence during the orchestration process. After selecting the best option to evolve, the achieved decision is translated to the Petri nets model by increasing the priority associated with the selected transition—in this case, transition t3. Analyzing the priority of alternative transitions, the logic controller will evolve the system by firing the transition with a higher priority, activating the corresponding web services, and sending a message to the machine.

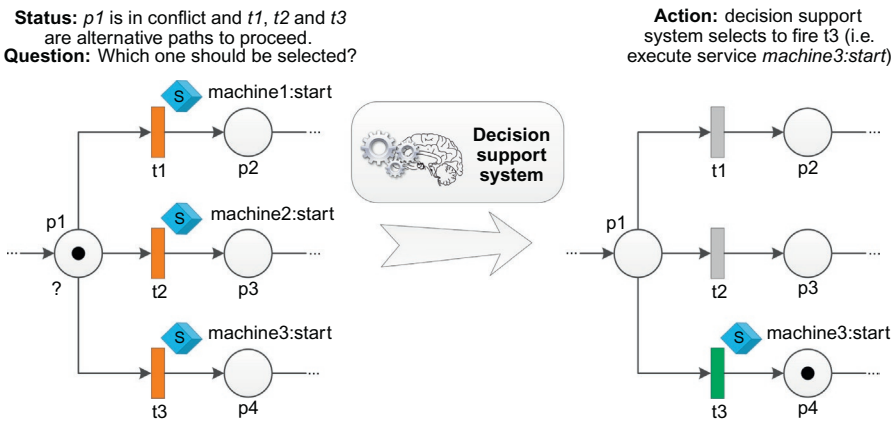

FIGURE 4.12

Petri net-based orchestration with a decision support system.

The orchestration models can be connected together via the ports of the models, using two alternative ways:

- *Offline composition*, which permits generating a new model based on the connection of individual ones. For this connection, the information has to be set up in the Petri net models, and an XML connection file must be defined to describe which models will be connected and through which ports.
- *Online composition*, which permits the intercommunication of two engines and their respective models via the exposition and request of services (this is already part of the information of the models designed before).

At the time of the experimentation, there were only three available STB devices embedding Petri net orchestration engines, which are only able to run one model at a time. The solution was using the offline composition to generate only three composed models (one for each orchestration device) and let them work together in real time using the online composition. Afterward, the decision was to split the system into three clusters of units, resulting in one model for C1-C3; one model for C4-C5, L1, L2, C10, and C11; and another model for C7-C9, ending up in three composed Petri nets models. The generated models communicate via each other (for the inter-transfer operation of pallets) using service invocation (i.e., the "TransferIn/TransferOut" mechanism).

The composition application shows that it is possible to design individual models without knowing the availability and disposability of the final orchestration devices. The experiment shows one possible way to compose the system using three devices and a defined distribution, but it could also be done with a different number of devices and other ways of division. Offline composition is used to limit the use of devices and network traffic, but introduces more complex models to be orchestrated (considering the limitations of embedded devices). On the other hand, the online composition is focused more on the distributed orchestration and the synchronization thereof. The correct division and use of the composition types depends always on the available resources, the optimization strategies, and the layout of the system, but orchestration models can be individually developed without knowing this information.

4.6 CONCLUSIONS AND FUTURE DIRECTIONS

Although agents in general, as well as industrial agents, have been investigated for several years, their productive use in industrial settings has been demonstrated but is limited. Other technologies and approaches that complement them have been used, as we have already discussed. However, with the prevalence of a new high-tech infrastructure driven by CPSs, as defined in the Industrie 4.0 vision, industrial agents have come again to the forefront of realizing the key features needed. As such, we see a renewed interest in the practical applications of industrial agents, especially in conjunction with CPSs, SOAs, and cloud computing. Their roles can vary from delivering intelligence to the infrastructure, acting as "glue" for legacy systems, and negotiating or mediating functionalities and services, etc.

To achieve large portions of the Industrie 4.0 vision, further research is required, with a focus on the usage of modern Internet technologies and services, but in industrial production. The latter assumes a good understanding of the challenges and limitations posed in real-world industrial systems, as well as the optimization of agent systems to make them sustainably operational in such environments, as shown in the first industrial prototype applications reported at the beginning of the last decade (Colombo et.al. 2006).

ACKNOWLEDGMENTS

The authors would like to thank the European Commission for their support, and the partners of the EU FP6 project SOCRADES (www.socrades.eu), EU FP7 IMC-AESOP (www.imc-aesop.eu) and EU FP7 ARUM (www.arum-project.eu) for their fruitful support and discussions.

REFERENCES

Bangemann, T., Karnouskos, S., Camp, R., Carlsson, O., Riedl, M., McLeod, S., Harrison, R., Colombo, A.W., Stluka, P., 2014. State of the art in industrial automation. In: Colombo, A.W., Bangemann, T., Karnouskos, S., Delsing, J., Stluka, P., Harrison, R., Jammes, F., Martínez Lastra, J.L. (Eds.), Industrial Cloud-Based Cyber-Physical Systems: The IMC-AESOP Approach. Springer, Switzerland, pp. 23–47. http://dx.doi.org/10.1007/978-3-319-05624-1_2.

Bellifemine, F., Caire, G., Greenwood, D., 2007. In: Developing Multi-Agent Systems with JADE. Wiley, USA.

Bepperling, A., Mendes, J.M., Colombo, A.W., Schoop, R., Aspragathos, A., 2006. A framework for development and implementation of web service-based intelligent autonomous mechatronics components. In: Proceedings of the IEEE International Conference on Industrial Informatics, Singapore, pp. 341–347.

Bing, L., Huaping, C., 2005. Web service composition and analysis: a Petri-net based approach. In: First International Conference on Semantics, Knowledge and Grid (SKG'05), November.

Chafle, G.B., Chandra, S., Mann, V., Nanda, M.G., 2004. Decentralized orchestration of composite web services. In: Proceedings of the 13th International World Wide Web Conference on Alternate Track Papers & Posters. ACM Press, New York, pp. 134–143.

Cheaib, N., Otmane, S., Mallem, M., 2008. Combining FIPA agents and web services for the design of tailorable groupware architecture. In: Proceedings of the 10th International Conference on Information Integration and Web-based Applications & Services. pp. 702–705.

Colombo, A.W., Schoop, R., Neubert, R., 2006. An agent-based intelligent control platform for industrial holonic manufacturing systems. IEEE Trans. Ind. Inform. 53 (1), 322–337.

Colombo, A.W., Karnouskos, S., 2009. Towards the factory of the future: a service-oriented cross-layer infrastructure. In: ICT Shaping the World: A Scientific View. European Telecommunications Standards Institute (ETSI), Wiley, New York, pp. 65–81.

Colombo, A.W., Bangemann, T., Karnouskos, S., Delsing, J., Stluka, P., Harrison, R., Jammes, F., Lastra, J. (Eds.), 2014. Industrial Cloud-Based Cyber-Physical Systems: The IMC-AESOP Approach. Springer, Switzerland, ISBN: 978-3-319-05623-4.

Fayçal, H., Habiba, D., Hakima, M., 2010. Integrating legacy systems in a SOA using an agent based approach for information system agility. In: Proceedings of the International Conference on Machine and Web Intelligence (ICMWI'10). pp. 338–343.

Fielding, R.T., 2000. Architectural Styles and the Design of Network-Based Software Architectures (Ph.D. Thesis). University of California, Irvine, CA.

FIPA, 2002. FIPA Abstract Architecture Specification. Standard of the Foundation for Intelligent Physical Agents. http://www.fipa.org/specs/fipa00001.

Hamadi, R., Benatallah, B., 2003. A Petri net-based model for web service composition. In: Proceedings of the 14th Australasian Database Conference, Darlinghurst, Australia, pp. 191–200.

Huhns, M.N., 2002. Agents as web services. IEEE Internet Comput. 6 (4), 93–95.

Jacobi, S., Hahn, C., Raber, D., 2010. Integration of multiagent systems and service oriented architectures in the steel industry. In: Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT'10), vol. 2, pp. 479–482.

Jammes, F., Smit, H., Martinez Lastra, J.L., Delamer, I., 2005. Orchestration of service-oriented manufacturing processes. In: Proceedings of the 10th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA'05), vol. 1, pp. 617–624.

Jammes, F., Karnouskos, S., Bony, B., Nappey, P., Colombo, A.W., Delsing, J., Eliasson, J., Kyusakov, R., Stluka, P., Tilly, M., Bangemann, T., 2014. Promising technologies for SOA-based industrial automation systems. In: Colombo, A.W., Bangemann, T., Karnouskos, S., Delsing, J., Stluka, P., Harrison, R., Jammes, F., Martínez Lastra, J.L. (Eds.), Industrial Cloud-Based Cyber-Physical Systems: The IMC-AESOP Approach. Springer, Switzerland, pp. 89–109. http://dx.doi.org/10.1007/978-3-319-05624-1_4.

JBOSS, 2014. JBOSS Middleware. http://www.jboss.org (accessed 23.09.14).

Karakoc, E., Kardas, K., Senkul, P.A., 2006. Workflow-based web service composition system. In: Proceedings of the 2006 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology. IEEE Computer Society, Hong-Kong, pp. 113–116.

Karnouskos, S., Tariq, M.M.J., 2008. An agent-based simulation of SOA-ready devices. In: Proceedings of the 10th International Conference on Computer Modeling and Simulation. IEEE Computer Society, Cambridge, England, pp. 330–335.

Karnouskos, S., Tariq, M.M.J., 2009. Using multi-agent systems to simulate dynamic infrastructures populated with large numbers of web service enabled devices. In: Proceedings of the International Symposium on Autonomous Decentralized Systems (ISADS'09), Athens, Greece, pp. 1–7.

Karnouskos, S., Savio, D., Spiess, P., Guinard, D., Trifa, V., Baecker, O., 2010. Real world service interaction with enterprise systems in dynamic manufacturing environments. In: Benyoucef, L., Grabot, B. (Eds.), Artificial Intelligence Techniques for Networked Manufacturing Enterprises Management. Springer, Switzerland, pp. 423–457. http://dx.doi.org/10.1007/978-1-84996-119-6_14.

Karnouskos, S., Colombo, A.W., Bangemann, T., 2014a. Trends and challenges for cloud-based industrial cyber-physical systems. In: Colombo, A.W., Bangemann, T., Karnouskos, S., Delsing, J., Stluka, P., Harrison, R., Jammes, F., Martínez Lastra, J.L. (Eds.), Industrial Cloud-Based Cyber-Physical Systems: The IMC-AESOP Approach. Springer, Switzerland, pp. 231–240. http://dx.doi.org/10.1007/978-3-319-05624-1_11.

Karnouskos, S., Colombo, A.W., Bangemann, T., Manninen, K., Camp, R., Tilly, M., Sikora, M., Jammes, F., Delsing, J., Eliasson, J., Nappey, P., Hu, J., Graf, M., 2014b. The IMC-AESOP architecture for cloud-based industrial CPS. In: Colombo, A.W., Bangemann, T., Karnouskos, S., Delsing, J., Stluka, P., Harrison, R.,

Jammes, F., Martínez Lastra, J.L. (Eds.), Industrial Cloud-based Cyber-Physical Systems: The IMC-AESOP Approach. Springer, Switzerland, pp. 49–88. http://dx.doi.org/10.1007/978-3-319-05624-1_3.

Leitão, P., 2012. Towards self-organized service-oriented multi-agent systems. In: Borangiu, T., Thomas, A., Trentesaux, D. (Eds.), Service Orientation in Holonic and Multi-agent Manufacturing and Robotics. Springer-Verlag, Berlin, Heidelberg, pp. 41–56.

Leitão, P., Colombo, A.W., Restivo, F., 2005. ADACOR, a collaborative production automation and control architecture. IEEE Intell. Syst. 20 (1), 58–66.

Leitão, P., Marik, V., Vrba, P., 2013. Past, present, and future of industrial agent applications. IEEE Trans. Ind. Inform. 9 (4), 2360–2372.

Mahnke, W., Leitner, S.H., Damm, M., 2009. OPC Unified Architecture. Springer, Heidelberg, ISBN: 978-3-540-68899-0.

Marín, C., Mönch, L., Leitão, P., Vrba, P., Kazanskaia, D., Chepegin, V., Liu, L., Mehandjiev, N., 2013. A conceptual architecture based on intelligent services for manufacturing support systems. In: Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics (SMC'13), pp. 4749–4754.

Mendes, J.M., Leitão, P., Colombo, A.W., Restivo, F., 2008. Service-oriented process control using high-level Petri nets. In: Proceedings of the 6th IEEE International Conference on Industrial Information (INDIN'08), Daejeon, South Korea, 13–16 July, pp. 750–755.

Mendes, J.M., Leitão, P., Restivo, F., Colombo, A.W., 2009a. Service-oriented agents for collaborative industrial automation and production systems. In: Marik, V., Strasser, T., Zoitl, A. (Eds.), Proceedings of the 4th International Conference on Industrial Applications of Holonic and Multi-Agent Systems (HoloMAS'09). Springer-Verlag, Berlin, Heidelberg, pp. 1–12 (LNAI 5696).

Mendes, J.M., Bepperling, A., Pinto, J., Leitão, P., Restivo, F., Colombo, A.W., 2009b. Software methodologies for the engineering of service-oriented industrial automation: the Continuum Project. In: Proceedings of the 33rd Annual IEEE International Conference on Computer Software and Applications (COMPSAC'09), Seattle, WA, USA, 20–24 July, pp. 452–459.

Nguyen, X.T., Kowalczyk, R., 2007. WS2JADE: integrating web service with jade agents. In: Huang, J., et al. (Eds.), Proceedings of the SOCASE 2007 Conference on Service-Oriented Computing: Agents, Semantics, and Engineering. Springer-Verlag, Berlin, Heidelberg, pp. 147–159 (LNCS 4504).

OASIS, 2006. Reference Model for Service Oriented Architecture 1.0. http://docs.oasis-open.org/soa-rm/v1.0 (accessed 12.10.06).

OASIS, 2007. Web Services Business Process Execution Language Version 2.0. OASIS Standard. http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html.

OASIS, 2009. Devices Profile for Web Services (DPWS). http://docs.oasis-open.org/ws-dd/dpws/1.1/os/wsdd-dpws-1.1-spec-os.html.

Ribeiro, L., Barata, J., Mendes, P., 2008. MAS and SOA: complementary automation paradigms. In: IFIP International Federation for Information Processing, vol. 266. Springer, Boston, pp. 259–268.

Shafiq, M.O., Ali, A., Ahmad, H.F., Suguri, H., 2005. AgentWeb gateway—a middleware for dynamic integration of multi agent system and web services framework. In: Proceedings of the 14th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprise.

TSB, 2014. TIE Smart Bridge. http://businessintegration.tiekinetix.com/nl/contact/smartbridge-for-suppliers (accessed 23.09.14).

W3C (World Wide Web Consortium), Web Services Glossary, 2004. http://www.w3.org/TR/ws-gloss/.

Ziyaeva, G., Choi, E., Min, D., 2008. Content-based intelligent routing and message processing in enterprise service bus. In: Proceedings of the International Conference on Convergence and Hybrid Information Technology (ICHIT'08), pp. 245–249.