Proceedings of the 23rd IASTED International Multi-Conference
PARALLEL AND DISTRIBUTED COMPUTING AND NETWORKS
February 15-17, 2005, Innsbruck, Austria

SERVICE-ORIENTED PARADIGMS IN INDUSTRIAL AUTOMATION

François Jammes
Schneider Electric
Corporate Research
38050 Grenoble Cedex 9, France

Harm Smit
Schneider Electric
Corporate Research
38050 Grenoble Cedex 9, France

Abstract – This paper outlines opportunities and challenges in the development of next-generation embedded devices, applications and services, resulting from their increasing intelligence – it plots envisioned future directions for intelligent device networking based on service-oriented high-level protocols, in particular as regards the industrial automation sector – and outlines the approach adopted by the SIRENA project, as well as the business advantages this approach is expected to provide.

Keywords – Industrial Automation, Service-Oriented Architectures, Web Services, Holonic Architectures.

1. Overview

After a description of the challenges faced by the manufacturing community and of the opportunities resulting from increasing miniaturization and usage of standard communication protocols (chapter 2), an overview is given of how the introduction of service-oriented paradigms – in particular when implemented using Web Services – may help to address these challenges and to promote these opportunities (chapter 3). Chapter 4 then outlines the service-oriented communication framework proposed by the SIRENA project for networking of industrial devices, and discusses the anticipated benefits of this approach.

As implementation of the SIRENA framework is ongoing, no information on its experimental usage can be reported as yet.

2. Introduction

Innovation landscape in the manufacturing industry

The environment of future manufacturing enterprises will be characterized by frequently changing market demands, time-to-market pressure, continuously emerging new technologies and, above all, global competition. Therefore, next-generation manufacturing strategies must support global competitiveness, innovation and introduction of new products, and strong market responsiveness. Resultantly, if cost and quality remain vital concerns, manufacturing systems need to become more strongly time-driven and time-oriented. This evolution requires considerably more flexibility and adaptability to change than present-day manufacturing systems can afford.

Currently, one third of the total cost of a manufacturing plant over its lifetime is spent on installation and set-up. Maintenance downtime accounts for another substantial portion of the operating costs. If a plant has to be adapted to new products by changing its process flow and introducing new or replacing obsolete or non-competitive equipment that is provided by different makers, then the downtime and installation costs rise considerably. The inflexible communication infrastructure among the manufacturing process components and the difficulty of porting existing application software to new configurations are the principal roadblocks.

Today, machine controls are categorized according to their physical functionality (programmable logic controller, motion control, regulators) and are programmed separately to execute sequences of commands as function primitives. Communication between the individual controls is facilitated by a central system in a hierarchical network. This traditional design approach presents major deficiencies when used as a basis for an intelligent manufacturing control structure.

An open, flexible and agile environment with "plug-and-play" connectivity is desperately needed. Open platforms have been proposed for years in the Information and Communication Technologies (ICT) domain. This movement has prompted the equipment industry to look at open solutions for manufacturing plants. Several proposals have been put forth by a variety of consortia and standards bodies. However, reality today still shows the dominance of proprietary standards that severely impede the progress towards flexibility and agility.

As documented in [1], the fundamental requirements to be satisfied by the manufacturing plants and control systems of the future include the following:

- Intra-enterprise dynamic integration capabilities
- Cross-enterprise cooperation
- Support of heterogeneous yet interoperable hardware and software environments
- Agility through adaptability and reconfigurability
- Scalability by adding resources without disrupting operations
- Fault tolerance and graceful recovery from failures

The issues at stake far exceed the technological challenge per se: in today's global economy, optimizing manufacturing processes is certainly more important than in the past to maintain productivity and competitiveness.

In the manufacturing community, multi-agent and holonic systems have been the subject of great attention

(ref. e.g. [2], [3]), but despite their promise, they have not made significant inroads in manufacturing plants in use today. In addition to the lack of widely accepted standards, one of the reasons for this situation seems to be that their implementations only cover part of the manufacturing landscape, whilst other areas remain subjected to the reign of proprietary standards, methods and mechanisms, resulting in a rigid patchwork of technology islands with poor scalability.

The technological orientations presented in this paper intend to contribute to the creation of the open, flexible and agile environment referred to above, by extending the scope of the holonic architecture approach through the application of a unique communications infrastructure, down from the lowest levels of the manufacturing device hierarchy up into the manufacturing enterprise's higher-level business process management systems.

Opportunities & challenges

Internet and Web technology is on its way to underpin a pervasively networked world in which billions of people and trillions of devices are going to be interconnected in various ways. As part of this evolution, Internet technology is emerging as the basic carrier for interconnecting electronic devices – used in industrial automation, automotive electronics, building controls, home automation, etc. This tendency is the result of several converging evolutions:

- The availability of affordable, high-performance, low-power electronic components allows embedding unprecedented horsepower into very tiny components. This technology can be leveraged to build advanced functionality into embedded devices, thus enabling new distributed application paradigms based on interconnected, self-reliant "smart devices".
- Owing to their low cost, both wireline and wireless local area networks of the Ethernet type are becoming widely accepted as the medium of choice for device networking. This migration process is quite recent in the industrial sector, where industrial Ethernet is gradually replacing traditional fieldbus networks.
- On top of these networks, due to their widespread adoption Internet protocols of the TCP/IP family are becoming the standard vehicle for exchanging information between networked devices. Again, in the industrial sector, this adoption process is ongoing and far from being fully accomplished.
- The emergence of data interchange mechanisms based on Extensible Markup Language (XML) data formatting paves the way for developing high-level data interchange standards at the device level.
- The advent of the XML-based Web Services paradigm for interconnecting heterogeneous applications through a lightweight communications infrastructure opens a perspective of universal, platform- and language-neutral connectivity.
- The ubiquitous presence of Internet technology allows "invisible" embedded devices and user-facing devices as well as higher-level information systems to coexist on the same network and, hence, to communicate.

As a consequence, the device networking ecosystem is expected to substantially evolve in the forthcoming years. This evolution will pave the way for novel, cost-effective communication paradigms at the level of basic industrial devices like sensors and actuators. This upcoming device networking ecosystem shall address the following requirements and challenges.

- Internet and Web standards shall be used pervasively.
- A device shall provide a universal, interoperable and secure access interface, independently of any operating system or programming language.
- It shall be possible to control a device from an ordinary ICT environment.
- It shall be easy to integrate a device into complex subsystems, and to do so in a scalable manner.
- Such a subsystem shall itself be exposed as a device that can, in turn, be integrated into more complex (sub)systems.
- A device shall be readily re-usable at various architecture levels.
- It shall be possible to connect devices together without extensive installation procedures (plug-and-play connectivity).
- A device shall present a high-level management interface in order to facilitate configuration, monitoring, fault diagnosis and maintenance.
- Interactions shall be made predictable (real-time demands).

It is advocated here that these requirements are best addressed by adopting a service-oriented architecture at all levels of a manufacturing enterprise, including at the level of network-connected elementary devices like sensors and actuators. The next chapter will highlight the fundamental aspects of such a service-based approach and how to transpose it into the device space.

3. Service-Oriented Architectures for Manufacturing Systems

Service-Oriented Architecture in a nutshell

There are many definitions of the concept of a service-oriented architecture. For the purpose of this paper, we use the following basic definition:

> ***A service-oriented architecture (SOA) is a set of architectural tenets for building autonomous yet interoperable systems.***

This definition is definitely incomplete, but it includes two key words: *autonomous* and *interoperable*.

The principal characteristics setting apart ***autonomous*** systems are that:

- they are created independently of each other,
- they operate independently of their environment,
- they provide self-contained functionality, i.e., this functionality would be useful even if it was not associated with any higher-level systems.

Interoperability is favored by clearly abstracting the interface that a service exposes to its environment, from the implementation of that service.

Autonomy and interoperability are contradictory properties. One of the challenges of SOA is, therefore, to reconcile these opposing principles.

The service-oriented architectural principles for achieving combined autonomy and interoperability can be summarized as follows.

- The design of a service interface follows an outside-in approach: rather than deriving it from the details of its implementation, the interface of the service is defined by focusing on how the service may fit in a larger business process context. Indeed, SOA is business process centric rather than technology centric – a service usually represents a business task. This property induces the granularity of services: a service interface is mostly coarse-grained and stateless, and is based on the exchange of documents. This is one of the aspects differentiating service-oriented from object-oriented design; the latter usually involves a conversation-style interaction pattern addressing individual object attributes.
- In a SOA, communicating entities are loosely coupled, by virtue of the fact that a service's functionality is exposed at its boundary, in terms of service "contracts" (interfaces) together with schemas describing the documents exchanged in a standardized, platform-agnostic format. Thus, the implementation of the service is totally opaque and may be modified without the service's users being affected. In contrast, the tightly coupled, context-related and stateful interactions typical of distributed object architectures create artificial dependencies between the communicating entities, resulting in rigid and fragile systems.
- SOA-based communications are of an asynchronous nature: when an action is invoked on a service, the result – if any – is returned to the invoking application entity without the latter suspending its operations. This is a departure from the synchronous remote procedure call mechanisms underlying distributed object systems, which have proven to be much less scalable than asynchronous communication patterns, especially in the presence of complex business processes where many operations with variable response times run concurrently.
- In addition to its interface description, a service can have requirements, properties or capabilities – expressed through "policies" – that can be negotiated at run-time. This is particularly useful for specifying characteristics like quality-of-service (QoS) level, security support, performance requirements, etc. Requirements for such properties may vary depending on how the service is used in a particular application context. The capability to negotiate their use at configuration time helps preserving the service's genericity and re-usability.

It may be noted that the above architectural tenets are inter-related: richer content enables looser coupling, which in turn enables asynchronous communication.

By achieving combined autonomy and interoperability, SOA addresses many of the above-mentioned requirements for the manufacturing system of the future:

- Integration capability: services can be readily integrated with other services, either statically or dynamically. Furthermore, services can be readily composed into higher-level services. Legacy technology can be encapsulated through service façades, according to a "wrap-and-re-use" rather than a "rip-and-replace" approach.
- Owing to the abstraction between service interface and service implementation, services can be materialized on heterogeneous software and hardware platforms. This opens an unprecedented perspective of being able to mix and match automation equipment from disparate vendors in the same manufacturing environment.
- Agility, flexibility and adaptability to change are greatly increased as services can be easily reconfigured or replaced, service deployment can be conducted incrementally and scaling can take place over time. Communicating entities can share and exchange resources and collaborate with each other through direct, peer-to-peer communication, i.e. without depending on the assistance and control of some higher-level entity. Decision-making can thus be driven down to the source of the information acted upon. This in turn enhances responsiveness and efficiency, while improving configurability. A decentralized mode of operation further adds resilience against failures by eliminating single-point-of-failure hazards.
- Development cost is reduced as re-use of services is facilitated and application programming is done at the highest possible level of abstraction.
- As each service encapsulates its own complexity, scalability becomes a built-in feature. By the same token, manageability and maintainability are greatly enhanced.
- Building fault-tolerant systems using a collection of self-reliant components is far less cumbersome than if using a set of tightly inter-related components.

SOA using Web Services

Web Services technology constitutes the preferred implementation vehicle for service-oriented architectures.

Web Services are services made available from a Web server for Web users or other Web-connected programs. The accelerating creation of Web Services is a major Web trend: backed by all major players in the computing industry, it constitutes the next wave of Web-based computing, laying the groundwork for a service-centric communications infrastructure. Web Services are totally platform-agnostic and can communicate with and/or be aggregated with other Web Services. Besides the standardization and wide availability of the Internet itself, Web Services are also enabled by the ubiquitous use of XML as a means of standardizing data formats and exchanging data.

As more fully documented in [4], the core Web Services standards are the following.

- **WSDL** (Web Services Description Language) for the abstract description of service interfaces and their binding to transport protocols. Interface definitions are made up of messages (requests, responses, notifications).
- **XML Schema** for the definition of the data formats used for constructing the messages addressed to and received from services.
- **SOAP**, the protocol for transporting service-related messages formatted in accordance with the corresponding WSDL definitions. The extensibility features built into SOAP make the Web Services architecture highly composable, allowing the various Web Service protocols to be integrated individually and incrementally, as well as to be improved and versioned in isolation, without affecting the rest of the protocol stack.
- **WS-Addressing** is closely associated to SOAP and concentrates all message addressing information into the header of the SOAP message envelope, thereby allowing the message content to be carried over any transport protocol (HTTP, SMTP, TCP, UDP, …).
- **WS-Policy** is used to express policies associated to a Web Service in the form of "policy assertions", complementing the WSDL description of the service.
- **WS-MetadataExchange** allows to dynamically retrieve metadata associated to a Web Service (description, schema and policy), thus providing a Web Service introspection mechanism.
- **WS-Security** is an optional set of mechanisms for ensuring end-to-end message integrity, confidentiality and authentication. Depending on the particular security requirements of a given application context, it may be complemented by WS-SecurityPolicy (policy assertions expressing security requirements), WS-Trust (management of security tokens), WS-SecureConversation (management of a security context between communicating entities using shared secrets), as well as by Transport Layer Security (TLS) in case there are no intermediaries between communicating endpoints.

SOA at the device level

The above-mentioned evolution of the device networking ecosystem paves the way for extending the SOA paradigm into the device space, that is, implementing a high-level communications infrastructure based on Web Services protocols at the device level, including in the lowest-level devices.

This approach to device-level SOA has already been adopted several years ago by the UPnP (Universal Plug and Play) initiative [5]. Like the Web Services architecture, the UPnP architecture leverages Internet and Web technologies including IP, TCP, UDP, HTTP, SOAP and XML. However, since the launch of UPnP predated the widespread adoption of Web Services, the current version of UPnP is not fully compatible with Web Services technology. In particular, instead of WSDL, it uses a specific – but nevertheless XML-based – language for service description. Furthermore, it uses specific protocols for discovery and event notification purposes.

It can be expected that UPnP will migrate to full alignment with Web Services, but for reasons of market strategy related to the lack of backwards compatibility, no date is set for this transition.

Meanwhile, the Web Services protocol suite has been extended with a profile specifically targeted at the device space: the "Devices Profile for Web Services" (DPWS) [6]. The DPWS specification is intended to foreshadow the next major upgrade of UPnP. The DPWS protocol stack is depicted below.

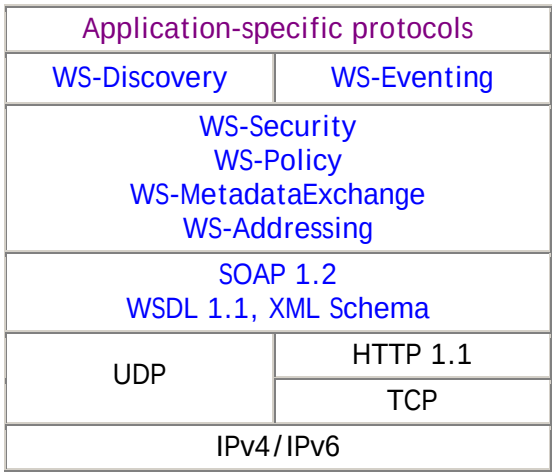

Fig. 1 – Devices Profile for Web Services protocol stack

With DPWS, all messaging, whether related to discovery, control or event notification, is based on the use of SOAP.

To the above-mentioned Web Services core protocols, DPWS adds WS-Discovery and WS-Eventing.

WS-Discovery [7] is a protocol for plug-and-play discovery of network-connected resources. Leveraging the SOAP/UDP binding, it defines a multicast protocol to search for and locate so-called target services. In the context of DPWS, a target service is a device. Once discovered, a device exposes the services it provides. The primary mode of discovery is a multicast probe for devices of a given type or scope; devices matching the probe send a unicast response. Devices can also be localized by name, through a similar protocol exchange. WS-Discovery protocol messages are sent over UDP in order to minimize network traffic overhead. To reduce the need for repeated probing, when a device joins the network, it announces itself by sending a multicast Hello message. When leaving the network in an orderly manner, a device announces this through a Bye message.

Multicast-based discovery is limited to local subnets. In order for discovery to be scalable to enterprise-wide scenarios, WS-Discovery introduces the notion of discovery proxy (DP).

If required by the application context, WS-Discovery may work in concert with WS-Security protocols and mechanisms to secure the communications.

WS-Eventing [8] defines a publish-subscribe event handling protocol allowing one Web Service ("event sink") to register interest ("subscribe") with another Web Service ("event source") in receiving messages about events ("notifications"). A subscription is leased by an event source to an event sink and expires over time. An event sink therefore has to renew the subscription periodically.

Event notification messages themselves are one-way messages, the content of which may include any data of any type. Filtering may be used in order to avoid sending unnecessary notifications.

Depending on the application context, it may be necessary that the communication between services be secured using WS-Security mechanisms.

WS-Eventing is intended to enable implementation of a range of applications, from device-oriented to enterprise-scale publish-subscribe systems, on top of the same substrate.

4. Industrial Device Networking with SIRENA

The SIRENA project [9] is an ongoing European R&D project aiming to develop a Service Infrastructure for Real time Embedded Networked Applications. SIRENA is part of the ITEA initiative [10], itself a "cluster" organization within the Eureka framework. The SIRENA consortium is made up of fifteen partners from three European countries.

SIRENA is creating a service-oriented framework for specifying and developing distributed applications in diverse real-time embedded computing environments, including not only industrial automation, but also automotive electronics, home and building automation and telecommunications systems. Though very diverse, these domains have a lot in common as far as the basic communications and control infrastructure is concerned. SIRENA will develop a set of common services to address this common denominator, complemented with domain-specific services for each of the target domains.

Given the tendencies outlined above towards the adoption of high-level communication functionality in intelligent devices, it has been decided to adopt the DPWS as the foundation technology for the SIRENA framework.

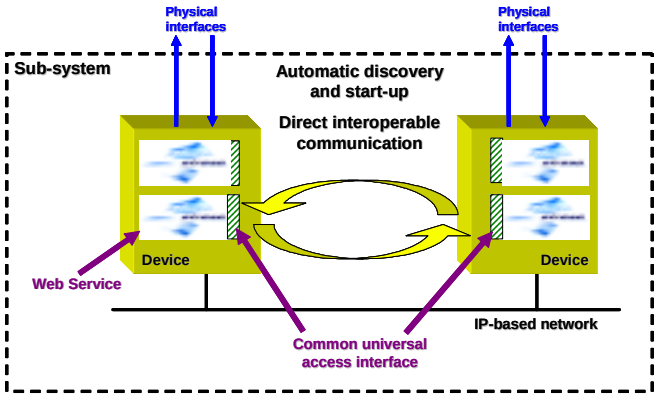

Fig. 2 – The SIRENA vision in a nutshell

The SIRENA vision is illustrated by fig.2, showing a subsystem made up of two SIRENA-enabled devices.

In the industrial automation sector, application of the SIRENA framework will enable new forms of device networking, breaking away from traditional master-slave architectures. This chapter highlights some of the anticipated benefits.

Interoperability

As illustrated by fig. 2, adoption of a high-level, universal access interface enables connection of devices from different vendors. This is a great stride forward with respect to the status quo in the industrial sector: although there exist several commercially available solutions for industrial device networking based on Ethernet and TCP/IP – such as those based on EtherNet/IP [11], PROFINET [12] and Transparent Ready [13] – there is no compatibility whatsoever between these technologies at the application level. The proposed solution framework has a strong unifying power, in particular due to its neutrality with respect to implementation technologies.

Scalable service composability and aggregation

We may illustrate this aspect through a very simple but real-world example, the "dose-maker" device illustrated by fig. 3.

Its role is to fill small bottles with granules flowing from a tank. It comprises a motor that causes the granules to leave the tank and a trap situated between the tank and the bottle to be filled; when the trap is opened, the granules can flow through. Sensors allow to determine when the trap is opened or closed, the tank is empty, the carter opened, etc. In order to fill a bottle, the dose-maker needs to open the trap, run the motor for a certain period of time, then close the trap.

Fig. 3 – Dose-maker device

When SIRENA-enabled, the dose-maker device may be organized as in fig. 4.

The Dose-maker is a composite device made up of the following logical components:

- a "Smart Motor", exposing a service that includes an operation of the type *'RunMotor(duration)'*, and that asynchronously notifies the completion of this operation;
- a "Smart Trap", exposing a service that comprises operations like *'OpenTrap'* and *'CloseTrap'*, and that sends asynchronous notifications when the trap is opened or closed, respectively;

- a "Dose-maker control" that orchestrates the operations of the Smart Trap and the Smart Motor, exposing itself a service that includes a command message of the type *'MakeDose(volume)'*, where *'volume'* can be either *"HALF"* or *"FULL"*, and that asynchronously notifies the completion of the dose making process.

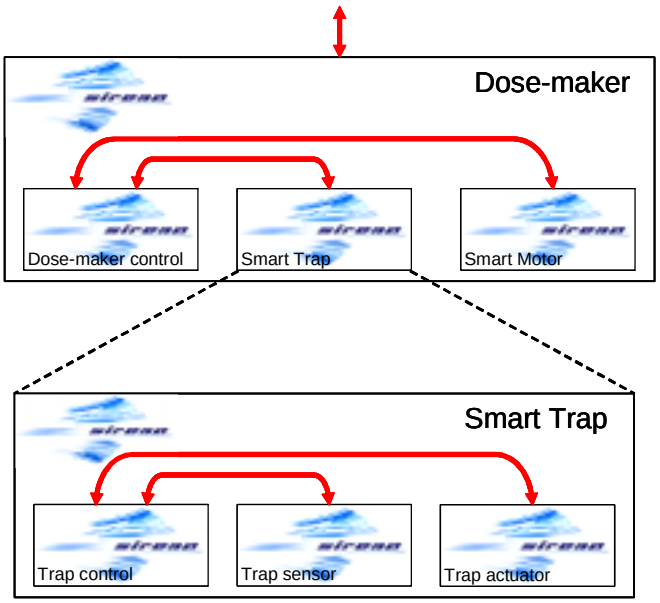

Fig. 4 – Dose-maker logical constituents

In turn, the Smart Motor and the Smart Trap are themselves composite devices constructed in a similar manner, as illustrated for the Smart Trap. The latter encompasses the actuator that actually controls the physical mechanism for opening and closing the trap, as well as the sensors that determine the trap's actual state.

This example illustrates how device-level services can be composed and aggregated into higher-level services, much in the way of "Russian dolls". Its principle is congruous to the holonic structuring paradigm: the Smart Trap is a *holon* in that it is both an autonomous entity (useable as such in devices other than dose-makers) and a subsumed part of a higher-level entity, viz. the Dose-maker.

The holonic structuring principle is greatly beneficial to scalability: hiding all the intricacies of dose-making and those of controlling a trap behind the high-level service interfaces of the Dose-maker and the Smart Trap, respectively, grants extensibility without interference with other system components. Scalability is further favored by the fact that event-driven communications are substantially more efficient than polling-based communications – in terms of both bandwidth usage and processing demands.

Uncoupling of logical and physical aspects

It is worth noting that the services exposed by the logical entities shown in fig. 4 – such as *'MakeDose'*, *'RunMotor'* and *'OpenTrap'* – are, in fact, *business processes* at the device level, in the same way that order handling is a business process at the enterprise or plant level. As noted before, this is a major characteristic of SOA: a service is directly related to a business task.

Consequently, as soon as SOA can be applied at the lowest level of the device hierarchy, it will be possible to map each of the logical elements in fig. 4 to a physical implementation component on a one-to-one basis. By the same token, it will be possible to express the entire architecture of a manufacturing installation in terms of business processes. Even if this is not yet feasible today – for reasons of cost-effectiveness and of responsiveness in the presence of severe real-time constraints – this is definitely the ultimate perspective of the SIRENA approach.

Nevertheless, the logical, service-based view of a device is completely independent of the device's physical realization. It is therefore possible to realize a pure SOA down to a certain level of the device hierarchy and to use currently available technology – such as the devices of the Transparent Ready family (ref. [13]) – in order to emulate the SOA below this level. For example, the Dose-maker device may be implemented using the ingredients shown in fig. 5. Each of the logical entities shown in fig. 4 can then be allocated to one of these physical entities.

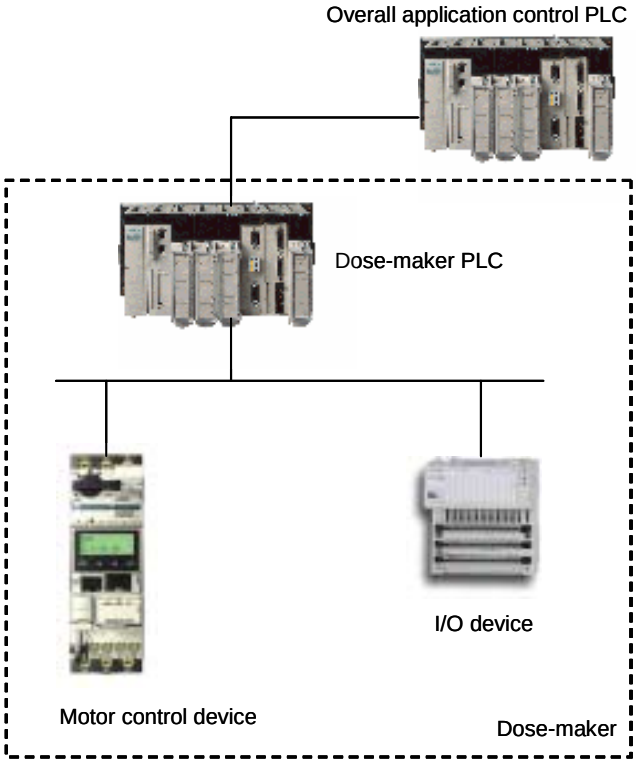

Fig. 5 – Present-day physical implementation components

A crucial aspect illustrated by this example is that, as technological progress allows intelligence to be driven further down into the lowest device levels, the mapping between logical and physical implementation components can gradually evolve over time without jeopardizing the architecture of the encompassing manufacturing system.

By virtue of this same property, it will also be possible to progressively apply SIRENA technology to more and more devices subjected to stringent real-time constraints.

Plug-and-play connectivity

Under the SIRENA approach based on DPWS, devices are able to automatically discover each other's presence.

In simple cases, devices can thus start to communicate once connected. For example, if all logical entities in fig. 4 are implemented as separate devices, the Dose-maker device becomes operational as soon as it has recognized the presence of the Smart Motor and Smart Trap devices, which themselves have gone through a similar discovery phase.

In more complex scenarios, e.g. with multiple devices of the same type, some initial configuration may be required.

Seamless integration with enterprise networks

The introduction of DPWS paves the way for the usage of a unique technology base, viz. Web Services, across the entire spectrum of enterprise applications, down from the sensor/actuator level up into the ERP/MES level. This perspective includes the integration of device-level networks into agent-based holonic systems built with Web Services.

Such strong integration power based on the ubiquitous use of Web Services technology does not preclude the usage of other service-oriented technologies. For example, if the OSGi Alliance [14] defined a "DPWS Service" like it has defined a "UPnP Service", OSGi could be used at intermediate levels in the manufacturing system hierarchy.

Integration with legacy technology

Migration from present-day architectures to an entirely different architecture cannot happen instantly. Given the large installed base of industrial device networks, coexistence of existing technology and newer ones must be planned for and migration paths must be devised that allow for gradual replacement.

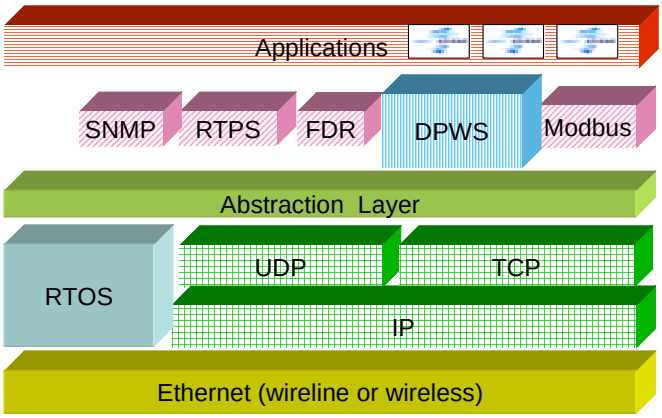

Fig. 6 – Dual protocol stack for industrial devices

There are two ways to achieve such legacy integration:

- A gateway approach, in which existing equipment is extended with a gateway device or service allowing the existing equipment to act as a member of the new device architecture. This is akin to the application integration paradigm in which legacy applications are front-ended with a façade making them look and behave like a Web Service.
- A dual stack approach allowing existing and new protocols to coexist. This is more expensive in terms of memory footprint, but provides a finer level of control over which protocols to use for which functions. Fig. 6 outlines a dual protocol stack being implemented by Schneider Electric. This setup will allow to use a protocol like Modbus for real-time sensitive tasks that cannot currently be handled satisfactorily by the DPWS stack, while preserving the possibility for future migration to full application of DPWS.

In practice, both approaches to legacy integration may be adopted at the same time for different application scenarios.

Simplified application development

In current industrial automation application practice, a Programmable Logic Controller (PLC) periodically scans sensors, processes their inputs, and periodically sends outputs to actuators. In this scheme, illustrated by fig. 7, the sensor and actuator devices are modeled as a collection of variables. Thus, all intelligence is concentrated in the PLC and a high cost is associated to the development, testing and maintenance of the PLC program, particularly on account of all the error handling it must incorporate.

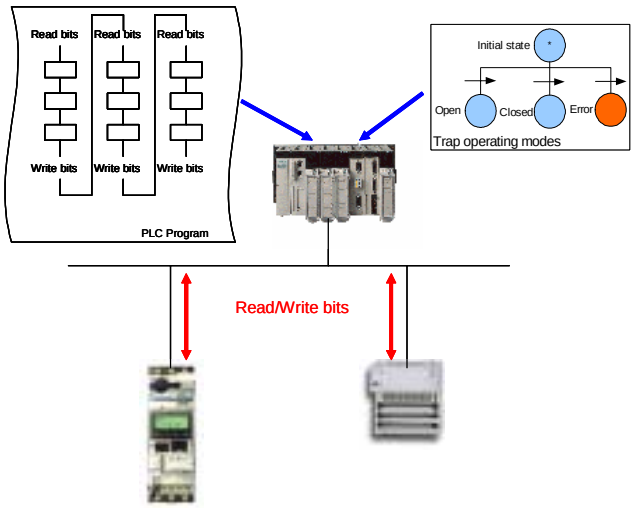

Fig. 7 – Present-day application development

In contrast, under the SIRENA approach (fig. 8) the complexity of operating the trap device is entirely concentrated inside the Smart Trap and programming the Dose-maker is done at the level of service invocations like *'OpenTrap'* and *'RunMotor'* and is essentially reduced to the orchestration (or "choreography") of the corresponding business processes. This regards both operational control and manageability aspects, as further discussed below.

Thus, providing the service interface of a given device type should be sufficient in order to:

- program the application,
- configure a particular device of that type,
- test and debug the application.

Furthermore, describing device operations through high-level service interfaces shall considerably facilitate the creation of tools for simulating the operation of a complex manufacturing system or subsystem.

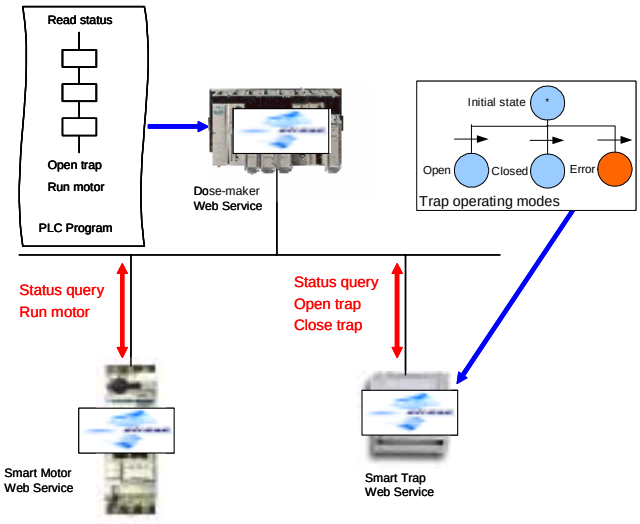

Fig. 8 – Service-based application development

Manageability

In present industrial automation practice, substantial effort is spent on diagnosing and repairing device malfunctions. The SOA-based SIRENA device architecture holds the promise of enabling significantly better device diagnostics capabilities than is achievable with current architectures. Indeed, under this approach the intricacies of a device's management functionality shall be an integral part of the complexity encapsulated within the device, with only high-level, synthetic information being communicated to the outside. Thus, in the case of a composite device made up of several lower-level devices, the correct functioning of the composite device as a whole (e.g. the Dose-maker) can be based on the management functionality provided by each of the embedded devices (Smart Motor and Smart Trap). Such health-checking or fault diagnosis functionality can be more or less sophisticated, depending on the capabilities of the embedded devices, but in any case its implementation will be facilitated by the fact of each embedded device providing a high-level manageability service interface.

Such manageability functionality may be supported by a generic manageability framework and it is anticipated that the SIRENA framework will evolve in this direction. Such a generic manageability framework should be highly extensible so as to cover the broad spectrum of device complexity to be coped with. For higher-level devices, management functionality may go as far as to include provisions for reliability engineering, self-reconfiguring or self-healing. Over time, the sophistication barrier may gradually shift as embedded processing capabilities evolve.

The recently published WS-Management specification [15] is expected to provide a valuable foundation for defining such a generic manageability framework. Its basic functionality is lightweight and resembles that provided by SNMP, which it is destined to replace.

5. Conclusion

The evolution of and the convergence between computing and networking technology, fuelled by advances in semiconductor and transmission technology, are bound to revolutionize the way communications are organized between systems and devices, in particular, embedded devices. Indeed, the emergence of increasingly powerful, network-connected devices will allow to drive the intelligence of computing and communications down to the device level, introducing possibilities for new, higher-level communication paradigms supported by open Internet protocol standards. Homing in on this tendency and leveraging the widespread adoption of service-oriented architectures using Web Services standards, the SIRENA project is implementing a high-level framework for communication and data exchange between devices, as well as between devices and applications. This approach will enable novel device networking architectures and will allow to seamlessly integrate device-level networks and enterprise-level networks. It thus holds the promise of prolonging the paradigm of holonic manufacturing systems across the entire spectrum of industrial automation networks.

6. Acknowledgement

This work was supported by the SIRENA project [9] in the framework of the European R&D program ITEA [10].

References

[1] W. Shen, D. Norrie, Agent-Based Systems for Intelligent Manufacturing: a State-of-the-Art Survey, *International Journal on Knowledge and Information Systems*, *1*(2), 1999, 129-156

[2] A. W. Colombo, R. Neubert, R. Schoop, A Solution to Holonic Control Systems, *Proc. IEEE-ETFA 2001*, 489-499

[3] M. Ulieru, R. Unland, Enabling Technologies for the Creation and Restructuring Process of Emergent Enterprise Alliances, *International Journal of Information Technology and Decision Making*, *3*(1), March 2004, 33-60.

[4] An Introduction to the Web Services Architecture and its Specifications, http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnwebsrv/html/introWSA.asp

[5] The UPnP Forum: http://www.upnp.org

[6] Devices Profile for Web Services, August 2004: http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnglobspec/html/devprof.asp.

[7] WS-Discovery, October 2004: http://ftpna2.bea.com/pub/downloads/ws-discovery.pdf

[8] WS-Eventing, August 2004: ftp://www6.software.ibm.com/software/developer/library/ws-eventing/WS-Eventing.pdf

[9] The SIRENA project: http://www.sirena-itea.org

[10] The ITEA initiative: http://www.itea-office.org

[11] EtherNet/IP: http://www.ethernet-ip.org

[12] PROFINET: http://www.profinet.com

[13] Transparent Ready: http://www.transparentready.com

[14] The OSGi Alliance: http://www.osgi.org

[15] WS-Management, October 2004: http://developers.sun.com/techtopics/webservices/management/WS-Management.pdf