Use of Web Services for next-generation automation systems

François Jammes & Harm Smit, Schneider Electric, Antoine Mensch, Odonata

Robert Harrison & Tom Kirkham, Loughborough University

January 2009

Abstract

This paper outlines the shortcomings of current automation architectures as well as the technological opportunities that will allow facing the requirements of contemporary and future automation systems. In particular, the customer values of applying the service-oriented architecture (SOA) paradigm implemented through Web Services technology, adopted at all levels of the automation hierarchy, are highlighted. The promises and initial results of this approach are presented, together with an outline of the standardization landscape.

Issues and challenges

Present-day industrial automation systems are no longer in line with market demands and requirements.

- ***Market dynamics***. Under today's global competition, production lines must be fast to set up and to reconfigure in order to promptly respond to varying market demands, to allow for mass customization and for very small lot sizes.
- ***Lack of interoperability***. Today's industrial installations are often rigid patchworks of technology islands with poor scalability, due to a lack of widely accepted standards and to the fact that systems are built up from too low-level components.
- ***Engineering practices***. Today, applications are completely static, as the engineering of manufacturing systems focuses on the construction of special-purpose machines for a single product and for a constant output flow and virtually every new piece of automation has its own unique control system. As a result, nearly every machine is built as "one-of-a-kind" and 80% of the engineering effort is devoted to re-implementing the control and related electrical systems each time a new machine is implemented on a new project. A lack of modularity and re-use, and a dependency between hardware and software development inhibit running engineering tasks in parallel, so that application testing cannot take place before final installation at the end user site. The presence of a plethora of – often incompatible – tools, due to the use of many disparate technologies, further reduces efficiency, both at design time and during operation. Since engineering cost accounts for 70% of the overall cost of an automation project, "automating the automation engineering work" is strongly needed. Hardware only accounts for 30% of the overall cost of an automation project, the remaining 70% being personnel cost (specifications, design, programming, cabling, commissioning, tests, maintenance, etc.). Consequently, the main challenge is not to reduce the device costs, but to improve the cost-effectiveness of the automation engineering work. As illustrated in Figure 1, a higher degree of modularity is definitely needed in order to allow engineering activities to take place in parallel and time-to-market to be reduced accordingly.

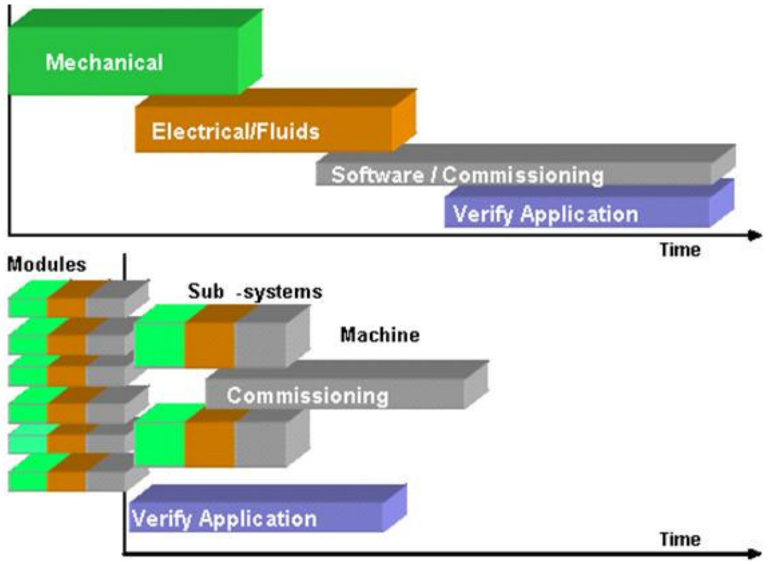

Figure 1 – Serial vs. parallel engineering tasks

- ***Disconnect between the shop floor and the top floor***. The convergence of enterprise resource planning (ERP) systems and manufacturing execution systems (MESs) is hampered by the fact they come from different horizons and are separated by different underlying technologies. Therefore, the exchange of data between the production control level (shop floor) and the enterprise application level (top floor) is often done either manually or semi-automatically, e.g., by exporting a production plan to a spreadsheet file that is then fed into the MES. Also, whilst enterprise applications are product-centric, plant-level applications are process-centric and asset-centric. This integration gap between the enterprise level and the plant level results in untimely and error-prone data exchanges, which may have very severe cost implications.
- ***Loss of process know-how***. The current personnel aging phenomenon is leading to a substantial shortfall of skilled labor among end users. This spurs a growth in supplier-provided services over the next several years, as end users will increasingly tend to outsource their automation, demanding an increasing level of vertical industry expertise from their suppliers. At the same time, it underscores the need for building systems out of higher-level technological components.

An open, flexible and agile environment with plug-and-play connectivity is desperately needed, to enable rapid reconfiguration and re-use of higher-level modules – integrating a variety of heterogeneous devices into an interoperable network of resources – to account for the introduction of new product models, as well as the ability to dynamically provide new value-added services and efficient diagnosis and maintenance.

Trends and opportunities

Attempts to address the need for more configurable production systems better able to meet the requirements of agile manufacturing have led to a growing interest in automation paradigms that model and implement production systems as sets of collaborating production units. Major trends, enabled by technological advances and evolutions, support this movement.

- ***Smarter, networked devices***. Sate-of-the-art electronics provide unprecedented computing horsepower in very tiny components. Devices embedding such components can thus be made "smart" and self-reliant, moving intelligence away from central controllers into individual devices. In addition, such devices can be network-connected and thus – if using universally adopted, vendor-neutral networking standards like TCP/IP, XML and Web Services – become members of the "Internet of things" or "Web of objects" that is gradually pervading the world. Furthermore, usage of a uniform communication paradigm will greatly contribute to the creation of an open, flexible and agile environment, and hence facilitate the elimination of the existing technology islands.
- ***Higher-level modularity***. On the basis of smart, networked devices, it becomes feasible to build automation systems out of higher-level, network-connected modules, each consisting of mechanical, electrical and control elements. Such a module is a controllable, reusable and reconfigurable part of a system, with process-oriented functionality. This functionality is encapsulated, i.e. exposed through well-defined interfaces of the highest possible level of abstraction, thus hiding to the outside world how the functionality is implemented. A module may be used in its own right or be combined with other modules to form composite modules, according to the "Russian dolls" composition paradigm. Indeed, as documented in [1], the notion of module covers different automation concepts: a module can be a standard machine, a cell, a workstation or a unit. The latter concept of "unit" or "mechatronic module", as the lowest level of autonomous unit of a production system – each with its specific process-oriented functionality, such as positioning, drilling, transportation, etc. – is the enabler of new, distributed control system architectures. Modules can be pre-tested as stand-alone components and such tested and proven components can be re-used across applications and systems. A similar line of thinking can be found in [2], even if it mostly addresses logical units and relies on proprietary technology.
- ***Reconfigurability***. Only systems capable of being rapidly reconfigured with very short design-development-ramp-up-produce life cycle phases can stand the pressures of a continuously changing market. Reconfigurability goes hand in hand with dynamicity, the capability to automatically add or remove devices, machines, functions, sub-systems, etc. at any time in an application. Reconfigurability is strongly favored by plug-and-play or plug-and-produce connectivity and by the use of tested and proven components.
- ***Maintainability***. One of the first requirements expressed by virtually all industrial customers is "improve the maintenance". The use of smart devices that encapsulate their own complexity and provide plug-and-play connectivity has the potential to substantially improve the effectiveness of device and system maintenance. To a large extent, smart devices shall be able to diagnose themselves, thus obviating the need for exposing a multitude of disparate information items, as is the current practice. Also, structuring devices in a modular, "Russian dolls" fashion helps zooming in on diagnostics in a top-down manner. Many connectivity and interoperability issues can be overcome by using open interoperable protocols.

Leveraging SOA based on Web Services in the device space

In just a few years, the concept of SOA has gained substantial traction in business system environments. In the context of automation systems, the characteristics of SOA closely match the technical and business level requirements outlined above.

In a nutshell, SOA is an architectural paradigm for building systems from autonomous yet interoperable components. A service provides business-meaningful, self-contained functionality that is not tied to particular usage scenarios service; it only exposes its interface ("contract"), which fully encapsulates the complexity of its implementation. Services can be published and discovered dynamically. SOA is characterized by coarse-grained service interfaces, loose coupling between service providers and service consumers, and message-based, asynchronous communication.

Leveraging the SOA paradigm allows for services to be re-used across processes and systems, and systems to be "built for change". Reliability is improved as applications and systems can be made up of tested and proven components. SOA offers the potential to provide the necessary system-wide visibility and interoperability in complex systems subject to frequent changes and operating in a multi-vendor environment. The net result is a substantial cost benefit together with increased agility, both from a technical and a business point of view.

The use of open, vendor-neutral standards, in particular those of the Web Services family, allows for implementing SOA in a platform-agnostic fashion, a key asset for gaining widespread adoption.

In order to apply the SOA paradigm in the device space using Web Services technology, the Devices Profile for Web Services (DPWS) was defined [3]. In addition to the core Web Services standards, such as SOAP, WS-Addressing, WSDL and XML Schema, DPWS includes WS-Discovery for plug-and-play device discovery and WS-Eventing for publish-subscribe asynchronous event notification.

DPWS was designed in 2004 by an industry consortium led by Microsoft and is natively integrated in Microsoft's Windows Vista platform, thus enabling intelligent devices, including PCs, to communicate across a network using Web Service protocols.

In the context of the ITEA/SIRENA project [4], Schneider Electric delivered the first implementation of DPWS for simple, low-cost, embedded devices and demonstrated that DPWS connectivity could be implemented using a 4€ processor chip. The SIRENA project was granted the ITEA Achievement 2006 award for this pioneering effort and the successful proof-of-concept delivered. Its results are used in the ongoing IST/SOCRADES project [5], as well as in the ITEA/SODA project [6] that finished at the end of 2008. The SIRENA and SODA projects also demonstrated that the proposed approach cuts across many application domains, including home automation, automotive electronics and telecommunications equipment and services; it has thus become feasible to create applications that integrate device-provided services with IP Multimedia System (IMS) services, allowing, for example, an alarm situation detected by a device to result in a phone call to an emergency center.

Schneider Electric's DPWS component was optimized for device implementation (footprint, performance). It has been ported to many target platforms and is interoperable with the Microsoft Vista implementation. Message processing performance is adequate for many applications; for demanding applications, binary instead of textual encoding of SOAP messages is expected to fit the bill. Schneider Electric's DPWS component is available as Open Source software, in two versions: C and Java. A developer community is now working and implementing solutions on this basis.

Device control and device management

For a given device type, two types of functions can be distinguished:

- those used for "peer-to-peer" communication between devices of the same hierarchical level; these functions constitute the control interface and are situated in the "control plane";
- those used for supervising a device by an entity belonging to a higher hierarchical level (MES, SCADA, ERP); these functions constitute the management interface and are situated in the "management plane".

The control plane and the management plane are largely orthogonal, but some functions may belong to both.

DPWS is well-suited for realizing control interfaces, which are primarily command-centric. In general, control interfaces are rather narrow and lend themselves to standardization.

A management interface is more comprehensive as it combines functions usually categorized according to the FCAPS model (Fault/alarm handling, Configuration, Accounting/administration, Performance monitoring, Security management). It addresses lower-level details than the control interface and is mostly data-centric. Only part of the management interface lends itself to standardization, the rest of it being vendor-specific. For the management interface, both DPWS and a management protocol like WS-Management can be used, the choice is not always obvious. Exclusive use of DPWS would result in the definition of a Get operation and (possibly) a Set operation for each and every device attribute, and parameterizing a device would require the execution of a usually long list of specific Set operations. This demonstrates the need for a management protocol, such as WS-Management, through which a device is perceived as a complex data structure (the resource model) that can be accessed both at the individual attribute level and at a more global level. Thus, parameterizing a device could be done in one fell swoop using a single Set command and a device's global configuration state could be obtained through a single Get command. Furthermore, the resource model may specify rules for coherence between different attributes.

WS-Management is a standard for identifying and communicating with manageable resources. It cuts across hardware, software and applications and was designed to scale down to all types of resources, including embedded devices. It nicely complements DPWS, as it uses the same base protocols, including WS-Eventing; it is lightweight and therefore usable in resource-constrained devices.

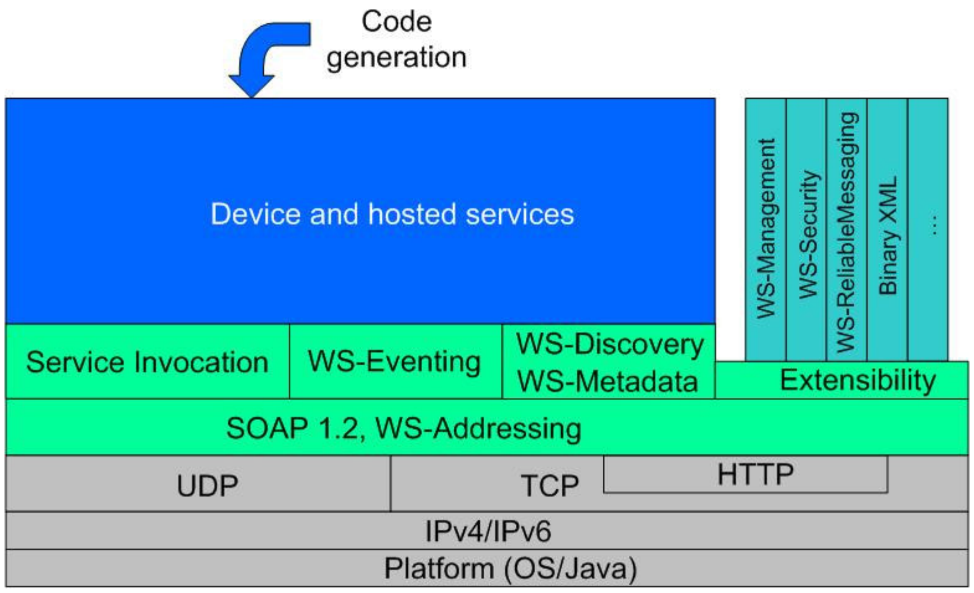

Figure 2 – DPWS-based device software stack

Figure 2 shows the device software stack, as implemented by Schneider Electric, supporting DPWS together with extensions such as WS-Management. The code that implements the functionality of the device and its hosted services is generated by a DPWS toolkit. The TCP/IP stack is usually provided with the device platform.

The standards landscape

Standardization of DPWS and WS-Management

Originally, DPWS was intended to be submitted for standardization to the UPnP Forum as the next major version of UPnP, referred to as UPnP V2. For reasons of marketing strategy, given the lack of backwards compatibility between DPWS and the current version of UPnP, the UPnP Forum decided to put this standardization path on hold. Consequently, in July 2008, a group of companies led by Microsoft submitted DPWS for standardization to OASIS (Organization for the Advancement of Structured Information Standards), as part of a set of three related standards for identifying, communicating with, and controlling network-connected devices of all kinds using Web Services: DPWS, WS-Discovery, and SOAP-over-UDP – collectively referred to as WS-DD (Web Services Discovery and Web Services Devices Profile). The target date for delivering the WS-DD standards – to be offered for royalty-free implementation – is March 2009.

Members of the OASIS WS-DD Technical Committee [8] include Microsoft, IBM, Schneider Electric, University of Rostock, University of Dortmund, CA, Novell, Software AG, Red Hat, Progress Software, WSO2, Canon, Fuji Xerox, Lexmark and Ricoh.

With 117 products from 37 vendors currently supporting it, either natively or through a bridge, the momentum of DPWS among device manufacturers is steadily growing. DPWS is natively incorporated in Microsoft's Windows Vista, Windows Embedded CE and .NET Micro Framework and will be so in Windows 7 and .NET 4.0. DPWS was also adopted by Beckhoff Automation, a prominent vendor in the industrial and building automation fields.

Standardization of DPWS represents a great stride forward, but is not self-sufficient. In addition, in order to allow for building systems out of higher-level basic units than is currently the case, it should be complemented by a set of "vertical" (domain-specific) service interface definitions for a set of basic, commonly used devices. Indeed, in studies undertaken by Loughborough University with Ford and their power-train assembly machine builder, ThyssenKrupp Krause, it was determined through system decomposition that an engine assembly machine is largely composed of a relatively small number of common control elements that provide standard control functionalities to the system. In fact, Krause's commissioning engineers have determined that typically 80% of the controls and equipment for power-train assembly are standard, and tremendous effort could be saved if the design, development and implementation methods for such control elements could be encapsulated and reused. In the continuous process industry, a similar proportion of recurring control elements can be observed in distillation columns [2].

In order to avoid being swamped by the high degree of variability in device characteristics and properties, such a standardization effort should not attempt to address all aspects of each of the device types involved, but mainly concentrate on their control interfaces. This is what the UPnP Forum did with its "device control protocols" for various categories of home control and entertainment devices, which were recently submitted, together with the UPnP base specification, for standardization to ISO/IEC.

WS-Management was ratified as a DMTF (Distributed Management Task Force) standard in April 2008. This standard only defines a management protocol, it does not prescribe any particular resource modeling framework. DMTF retains the existing CIM (Common Information Model) as its management modeling technology, in particular through WS-CIM, which describes the structure of XML Schema, WSDL and metadata constructs for elements of CIM models. A more comprehensive initiative in this direction is the Service Modeling Language (SML), submitted to the W3C in March 2007 by BEA, CA, Cisco, EMC, HP, IBM, Intel, Microsoft, and Sun. SML goes beyond CIM by writing definitions in native XML, making SML built for Web Services from the ground up.

Common ground between OPC UA and DPWS/WS-Management

Since a few years, a more or less a competitive solution to DPWS is emerging under the name of OPC UA [9], a Web Services based evolution of the well-known OPC technology, under the auspices of the OPC Foundation, which has submitted OPC UA for standardization to IEC. Despite its claims, OPC UA addresses communications between devices and higher-level systems rather than communications between devices and is not fit for implementation in resource-constrained devices – except when using the "UA Native" variant, which is unlikely to gather widespread support. Moreover, OPC UA protocols exhibit several drawbacks:

- OPC UA introduces a set of protocols of its own, e.g., its session management and event collection protocols; even if these are based on XML and Web Services technology, they are not standard Web Services protocols and cannot be readily composed with other Web Services protocols; consequently, an OPC UA server can only communicate with dedicated OPC UA clients, not with standard Web Services clients.
- Message exchange patterns are limited: OPC UA uses stateful, connected, synchronous message exchange patterns, even for managing events, which are received through polling; lack of support for asynchronous and stateless communications hampers the architecture's scalability and flexibility.
- OPC UA extensibility is very limited and disallows complete modeling of device functionalities as plain Web Services.

Furthermore, the entire OPC UA specification is grounded in the OPC UA resource modeling framework ("information model", [10]), which is inherently complex and imposes too much of a burden when dealing with simple resources, and has been designed for modeling objects used in industrial automation, not for modeling resources like applications and services.

Nevertheless, combined use of DPWS and OPC UA may provide a rich framework for device-level SOA. Due to the similarities in the basic building blocks of both specifications, a concerted implementation would avoid the duplication of effort in implementing a SOAP 1.2 stack, and additional Web Services protocols, as well as the extra memory footprint that would be induced by a duplicate implementation of some of these technologies.

There may indeed be an opportunity to define a resource modeling framework based on a subset of the OP UA information model that is useable for simple devices and provides a dual view of managed resources, that is, useable both through OPC UA and through WS-Management. Such a proposal is currently under discussion in the IST/SOCRADES project.

Proof-of-concept demonstrations

At the ITEA/ARTEMISIA Symposium, held in Rotterdam in October 2008, a test rig developed by Loughborough University in cooperation with Ford and Schneider Electric was used to demonstrate SOA-based plug-and-play interactions with sensors and actuators, both for device control and for device configuration and monitoring purposes. The demonstrator, pictured in Figure 3, is a miniature production line made up of electro-mechanical units from Festo. Communication with the production line takes place through Field Terminal Block (FTB) devices from Schneider Electric. Each FTB supports DPWS and WS-Management and acts as a nano-controller for one of the four sections of the production line; the FTB hardware comprises an ARM9-based chip incorporating 512 KB of Flash memory, 96 KB of RAM, an Ethernet connection and I/O connections. The operations of the FTBs are driven by a PC-based orchestrator. The demonstrator illustrated how production data can be combined with other enterprise data to improve the accuracy of decisions relating to production routing and supply chain management. The corresponding data was fed to project partners in the enterprise computing field (SAP) and industrial automation management sector (ARC Informatique).

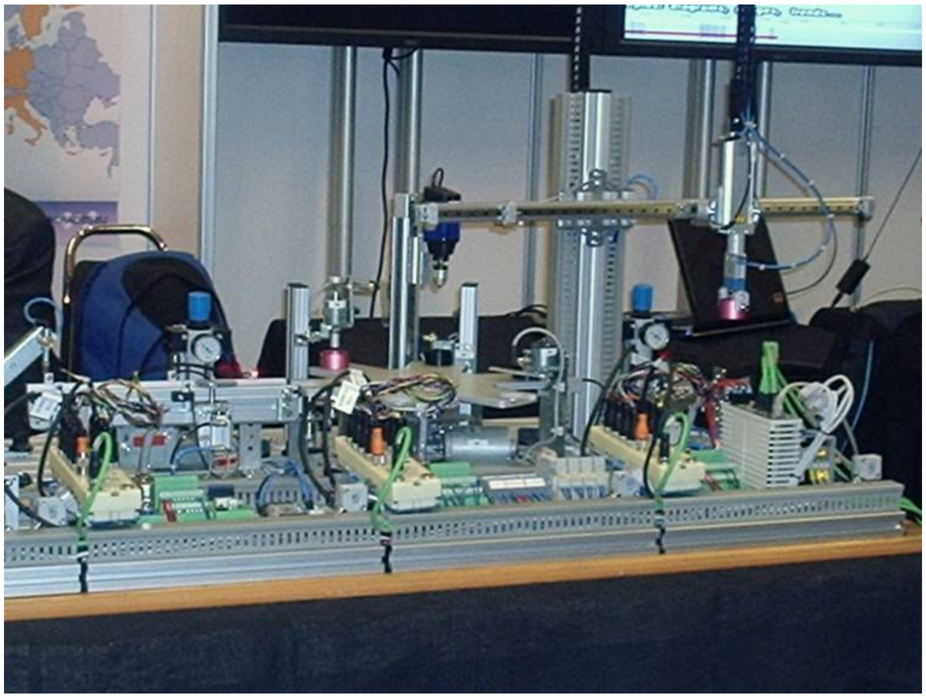

Figure 3 – Demonstration test rig

Similar demonstrations took place at the MACH 2008 Fair, held in Birmingham, UK, in April 2008 and at the ICT 2008 exhibition, held in Lyon, France, in November 2008.

References

[1] C. Schäfer: *On the Modularity of Manufacturing Systems*, IEEE Industrial Electronics Magazine, Vol. 1(3), Fall 2007, pp. 20-27

[2] O. Lorenz, B-M. Pfeiffer, *In Units denken*, http://www.computer-automation.de/nachrichten/steuerungsebene/steuernregeln/article/in_units_denken/6042/ea3b036c-b79c-11dd-8b44-001ec9efd5b0

[3] Outline of the Devices Profile for Web Services: http://en.wikipedia.org/wiki/Devices_Profile_for_Web_Services

[4] ITEA/SIRENA (Service Infrastructure for Real-time Embedded Networked Applications) project: http://www.sirena-itea.org

[5] IST/SOCRADES (Service-Oriented Cross-layer infRAstructure for Distributed, smart Embedded deviceS) project: http://www.socrades.eu

[6] ITEA/SODA (Service-Oriented Device and Delivery Architecture) project: http://www.soda-itea.org

[7] Web Services for Management, http://www.dmtf.org/standards/published_documents/DSP0226.pdf

[8] OASIS Web Services Discovery and Web Services Devices Profile (WS-DD) Technical Committee, http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=ws-dd

[9] OPC UA Specification: Part 1 – Concepts, http://www.opcfoundation.org/UA/Part1

[10] OPC UA Specification: Part 5 – Information Model, http://www.opcfoundation.org/UA/Part5