Integration of SOA-ready Networked Embedded Devices in Enterprise Systems via a Cross-Layered Web Service Infrastructure

†Stamatis Karnouskos, ‡Oliver Baecker, †Luciana Moreira Sá de Souza, †Patrik Spieß

†SAP Research, Vincenz-Priessnitz-Straße 1,
D-76131 Karlsruhe, Germany
{stamatis.karnouskos,luciana.moreira.sa.de.souza, patrik.spiess}@sap.com

‡SAP Research, Blumenbergplatz 9,
CH-9000 St. Gallen, Switzerland
oliver.baecker@sap.com

Abstract

Today manufacturers require efficient reaction to critical events occurring at the shop floor. Therefore, device-level data needs to be integrated into business processes in a standardized and flexible way to avoid time-consuming media breaks. Current approaches are characterized by a late indication of changes in the production environment and a delayed implementation of changed production plans. As a solution, we propose a web service-based integration of enterprise systems with shop-floor activities, using SOA-ready networked embedded devices. We examine the requirements for the integration and derive an appropriate architecture that tries to close the integration gap. The timely provision of data, the impact of device-level information on business processes, as well as the direct bidirectional communication with device-level services promotes the vision of adaptive manufacturing and leads to reduced production costs.

1. Introduction

Enterprises are moving towards service-oriented infrastructures where applications and business processes are modelled on top of cross-organization service landscapes. Currently, shop-floor intelligent manufacturing systems based on distributed embedded devices, incorporate system intelligence in a limited amount of monolithic computing resources accompanied by large numbers of resource constrained devices. The intelligence and behaviour are application-specific usually tailored to specific use cases.

However, as we are moving towards an "Internet of Things" [10] — where millions of devices are interconnected, provide and consume data about the real world and make it available to business processes that run in the digital world — more flexible and adaptive concepts for the integration of enterprise services with smart embedded devices are necessary. In the last years we have also witnessed the prevailment of Service oriented architectures (SOA) that use loosely coupled services to support more effectively the requirements of business processes and users. The work presented here proposes a web service-based approach where each device offers its functionality as a set of services. At the same time, each device can discover and invoke new functionality offered by services of other devices dynamically and on-demand. The cooperation of distributed, autonomous and reusable entities allows for a new dynamic infrastructure that is able to provide a better insight into shop-floor activities for higher level business processes and is able to adapt to changing business needs.

Our motivation stems from the integration gap between the business level and the production control level. Existing manufacturing execution systems (MES) and enterprise resource planning (ERP) systems are loosely connected, usually via inflexible proprietary hierarchical systems. From the business service layer, the access to specific device functionality is often restricted, and may not be tailored to the specific requirements of the enterprise service. Therefore, the exchange of data between the shop floor and the top floor is often done manually or usually semi-automatically, e.g. by exporting a production plan into a spreadsheet file that is then fed into the MES.

The described media break implies that data is not exchanged in a timely fashion and is error-prone, which leads to poor information visibility and dissemination. In

1-4244-0826-1/07/$20.00 © 2007 IEEE

addition, it results in business process delays and a lower responsiveness to problems because managers learn about critical issues too late. For example due to the denoted integration gap, if a machine breaks down in the production line, it can take hours if not days until a key account manager learns about missed due dates and customer orders might get lost. A web service-based direct integration of the production machine with an alert resolution dashboard observed by the key account manager would give enough time to contact the customer in advance and initiate counter-measures to prevent the loss of revenues. The direct accessibility of networked embedded devices via web services also allows for a more efficient interaction in scenarios where a communication flow via middleware layers is not required. Examples are the alteration of embedded software services during a feature up-selling business process or remote maintenance activities.

Assuming that networked embedded devices can be SOA-ready (i.e. offer their functionality via a web-service), has imminent influence on the way of designing and integrating future components and services. In the next sections, we discuss the requirements of a device-to-business integration infrastructure and focus on an architecture that can realise the integration of manufacturing systems and smart embedded devices with enterprise systems based on web services.

2 Requirements of a Device-to-Business Integration Infrastructure

In order to effectively integrate networked smart embedded devices with enterprise systems in a service-oriented way, several key requirements need to be fulfilled.

2.1 Web Service Support

Web Services have emerged as the de facto standard for enterprise application integration (EAI) [14] and are the common denominator that needs to be supported by all components participating in a service-oriented infrastructure as the one envisioned in this paper. Coupling web services with shop floor devices has the potential to increase the efficiency of business processes and allows for an overview of the network and interaction with its components. In this context, to improve the transparency of the connection between back-end systems and shop floor machinery, these machines should be able to expose their functionality as web services.

2.2 Event Driven Architecture (EDA)

Thousands of events are generated on the shop floor during normal operation. Some of these events can provide an overview of the current status of the network while others can indicate unexpected problems. Therefore, eventing support is a requirement for business applications to become deeply connected to the shop floor. Filtering (to select the messages that are of real interest to the back-end), local processing, and evaluation, are additional mechanisms that can enhance the performance and scalability of the eventing support.

2.3 Service Lifecycle Management

Service Lifecycle Management deals with the administration of services during their entire lifecycle. It manages the deployment of new services, the update of existing services, the starting and stopping of services, and the configuration and parameterization of running services. A sophisticated service lifecycle management has the potential to increase the availability of enterprise systems as it extends the possibilities of changing business processes without considerably influencing the efficiency of the entire system.

2.4 Business Process Modelling

Business architects are counting heavily on business process modelling tools to envision, design, and evaluate future business processes. In a real-time enterprise, device-level events can be actively integrated and evaluated during the execution of a business process. Therefore, devices, their data, and their operations need to be integrated in business process modelling tools that have the capability to plan a business process in detail and specify the interaction between the enterprise level and the device level.

2.5 Intermittent Connected Assets

Often devices connect only occasionally to the back-end system or suffer sudden disconnections, e.g. when a mobile device enters an area that provides back-end connectivity and updates its info in an ad-hoc manner or when moving between locations with varying quality of wireless connectivity. An architecture for the integration of smart embedded devices and enterprise systems has to ensure that the envisioned scenarios and its components also work while there is no permanent connection or a very limited one with the back-end system. Therefore, scenarios that require only local business intelligence and local interaction should be possible as well. A device needs to be able to find (and possibly cache) the data it requires locally and in a peer-to-peer way, without necessarily being connected to the back-end system.

2.6 Business Process Monitoring

Business Process Monitoring deals with the proactive and process-oriented monitoring of a company's core business processes. It includes the observation of all technical and application-related functions that are required for a smooth and reliable flow of the core business processes and comprises detailed procedures for error handling and problem resolution. To detect problem situations as early as possible in order to solve them before they become critical for the business, the monitoring of business processes needs to be accomplished in close cooperation with the affected devices themselves. It has to

be assured that monitoring from the business process level is feasible directly for a specific device and events can be actively sent to the responsible business process component.

2.7 Alerting

Alerting has to be supported especially for mission-critical devices. In a critical situation, messages have to be treated with high priority, and might be inefficient to defer this to a higher level. Furthermore, the business application should get only the necessary decision-critical information and not get overwhelmed with all alerting data from the shop floor. Therefore, support for the exchange of emergency data and a common alerting protocol have to be in place, which will regulate the communication flow between the different architecture layers.

2.8 Multi-faceted Enterprise Services

The proposed infrastructure must provide support for integrating back-end services that use online information of the shop floor. Examples of these services are risk management and maintenance control.

A sophisticated risk management makes the required decision-critical information available significantly earlier and leads to reduced threats. Based on this, strategies can be changed proactively while minimizing overall costs. Since the device itself knows how a failure of one of its components affects it in total, this info must be accessible so that business processes can take advantage of it.

Maintenance, Repair and Overhaul (MRO) ensures the availability of a device or restores it to a state in which it can provide its functionality. Any deviation from the expected business behaviour (e.g. based on sensor reading), should be reported, ideally by the device itself. An improvement to maintenance control is to predict when maintenance will be required. This can be achieved by performing equipment monitoring with the goal of conducting maintenance "just in time", that is before the equipment fails.

2.9 Standardised Communication and Information Exchange

For an efficient implementation of a plant-to-business integration, existing protocols for communication and information exchange should be used or extended. This mostly refers to the communication of back-end systems with devices, as well as device-to-device communication. Such support has implications regarding interoperability as it will ease the device integration and cooperation in a highly heterogeneous infrastructure. In that context for example OASIS [11] standards, the DPWS specification [9], OPC-UA [2], or the Business-to-Manufacturing Markup Language (B2MML) [16], which is an implementation of the ISA-95 standard, can be applied where appropriate.

2.10 Access to Device Status

For the management of smart embedded devices, a rich interface to the device status is necessary and options that can configure it or even allow code to be downloaded to the device are required. All devices should provide a standardized view of their capabilities according to a common ontology and scenario needs. Furthermore, the devices need to send events to a number of business layer services as well as to expert and planning systems. Generally, devices should provide full real-time access to their status to any authorized entity. In case of gateways that control a couple of devices, several status reports can be aggregated to hide complexity and provide a comprehensive view.

3 Architecture

The fundamental goal of the proposed architecture is to integrate device-level services with enterprise software systems running on the application level using web services technology as depicted in Figure 1. It is shown that there are at least two different ways to couple networked embedded devices with enterprise services. One way is a direct integration of device-level services in business processes, while another way exposes device functionality to the application layer via a middleware layer. Depending on the underlying use case, one or the other data flow is chosen leading to a hybrid architecture.

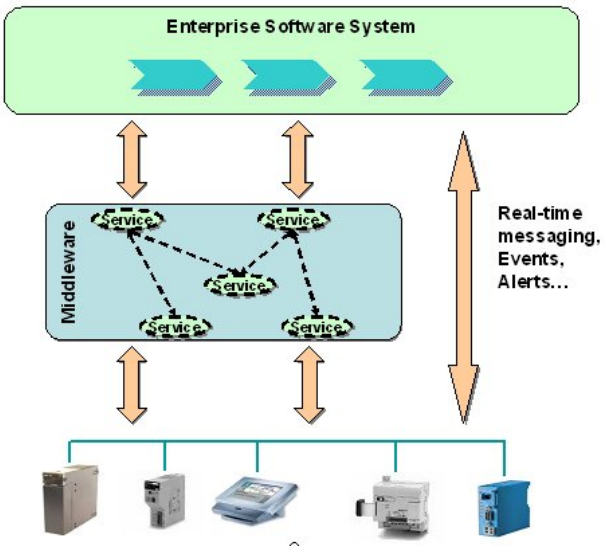

Figure 1. Coupling of Enterprise Services with Networked Embedded Devices.

In the context of this paper, a service is defined as a reusable software component that encapsulates device-specific functionality and makes it available in an interoperable and self-descriptive way over a network. It advertises this functionality to other networked embedded devices or further entities and enables them to locate and invoke the service. Thereby the invoking party is not aware

of how the service functionality is implemented. The service offers a given functionality to the service user, provides a well-defined interface that the service can be invoked through, runs transparently from the user's point of view (encapsulation), and can be composed of several other services (compound service).

The service-oriented architecture (SOA) approach allows for an increased flexibility and reusability when it comes to the provisioning of data and functionality coming from smart embedded devices. One key advantage of using services is that functionality provided by different smart devices can be composed to allow for more sophisticated application behaviour. The use of services is also desirable because today's business software is built more and more in a service-oriented way based on web services. Therefore, the SOA approach allows for an easy integration of smart embedded devices with business applications leading to a cross-layer composition of services. The integration architecture depicted in Figure 2 shows components that realize a device-to-business integration derived from the aforementioned requirements.

3.1 Device Layer

Devices of the manufacturing and process automation shop floor are represented in the device layer. The smart networked devices expose their functionality directly as web services using DPWS, via DPWS-enabled controllers, or by means of legacy system connectors (see requirement 2.1). In addition, they exchange messages via a peer-to-peer network. The landscape of devices is highly heterogeneous in terms of complexity and communication capabilities.

3.2 Composition Layer

The composition layer contains intermediate systems that provide compound services by combining the capabilities of smart devices (DPWS devices or devices accessed through DPWS-enabled gateways). Due to the distribution of functionality across smart devices, these composite systems are much more flexible and agile than the monolithic systems they replace. They perform complex control and management functions a single device could not offer by combining heterogeneous device-level services. For the connection of legacy devices that have insufficient resources to offer web services to the middleware layer, the composition layer provides dedicated controller components.

3.3 Middleware Layer

The middleware layer connects the device layer and composite layer with the application layer. It contains a device abstraction sub-layer, which primarily deals with eventing, service invocation, and device access. Interfacing with the application layer, the middleware contains a system management sub-layer that provides service lifecycle management and device administration. In the following sections, the sub-components and their contribution to the integration of web service enabled devices with enterprise services is discussed in more detail.

3.3.1 Device Abstraction

To connect all the devices present in the shop floor with enterprise applications in a transparent way, the middleware has a device abstraction sub-layer. This layer provides the required components for invoking device-level services in a synchronous or asynchronous way, propagating events and connecting to different device protocols.

Synchronous invocations (via the *Synchronous Request Processor* component) mean that applications expect an immediate response from the device, and in case that the device cannot respond, the invocation fails returning an error. This is mostly applicable for permanently connected devices. Asynchronous invocations (via the *Asynchronous Buffer*) give support to occasionally connected devices. An application would submit a request to the middleware and poll for the response periodically until (or get notified when) the device reconnects and delivers its response to the middleware (see requirement 2.5).

To support legacy devices, the abstraction from the used protocol is achieved by implementing a *Legacy System Connector* for every legacy platform. A connector is a component that translates back-end requests (web services) into the device-specific protocol. The same is true for transforming the events coming from the devices into standardized events that enterprise applications can process. *Legacy System Connectors* are only necessary for legacy systems, given that DPWS devices already support web services.

To provide a complete web service support for legacy systems, the middleware contains a *Proxy Pool* where a set of *Service Proxies* implements the service interfaces provided by legacy systems. The function of a *Service Proxy* is to allow back-end applications to access the functionalities present in legacy systems via web services. It forwards received requests to an *Invoker* component using either synchronous or asynchronous invocations. Finally, the *Invoker* forwards the requests to the *Connector Dispatcher* component, which selects the appropriate *Legacy System Connector* to perform the invocation.

It is simple to realize the improvements that an adoption of DPWS by all devices would provide. Using DPWS, there would be no need to have components to support legacy systems given that all devices are web service enabled by default (see requirement 2.9). This means the *Proxy Pool* as well as *Legacy System Connectors* would become obsolete.

In the *Device Abstraction* sub-layer, two components, which are inspired by the WS-Brokered Notification concept [15], are responsible for handling events propagated from the shop floor. The Notification Broker manages subscriptions and propagates events directly to the consumers and the *Pull Point* component provides the alternative for applications that do not support web services to

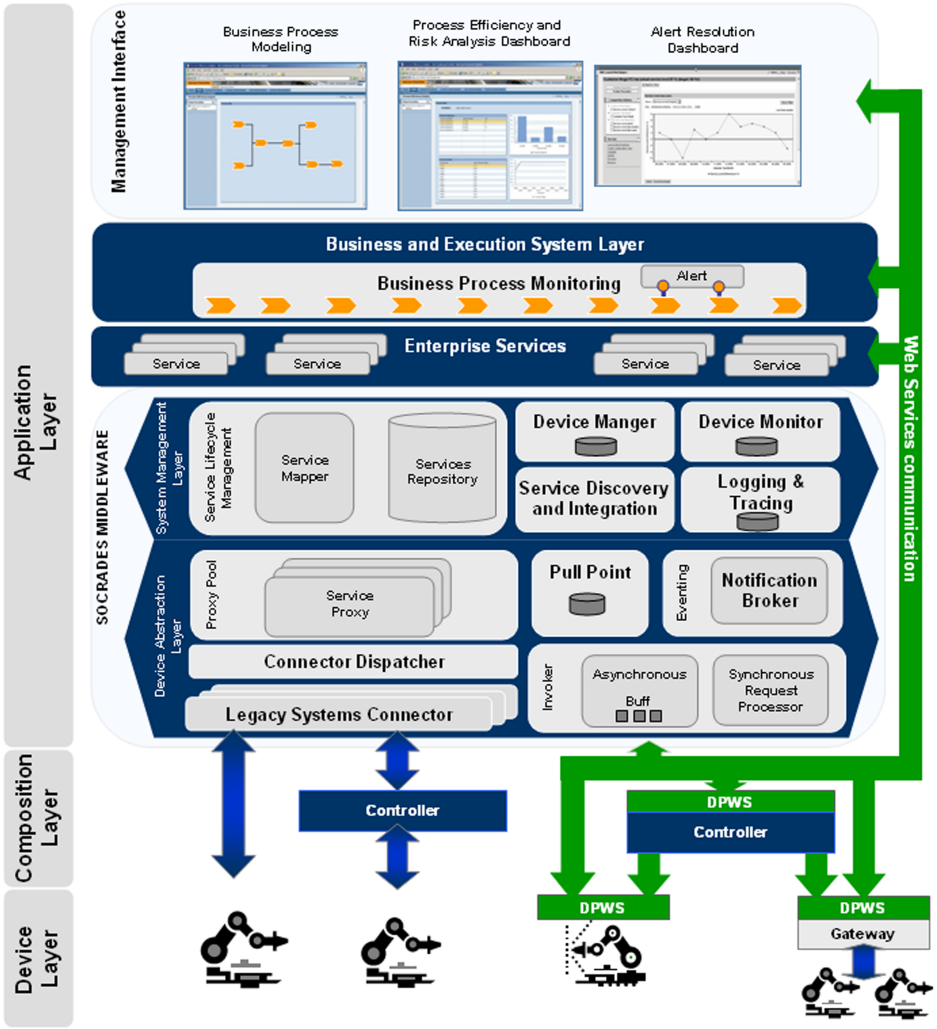

Figure 2. Integration Architecture.

retrieve their events periodically from a component (see requirement 2.2).

3.3.2 System Management

On top of the *Device Abstraction* sub-layer is the *System Management* sub-layer. This layer provides additional services that enable the management of the shop floor. *Service Lifecycle Management* is one of the additional functionalities that this layer brings. By storing a description of services into the *Service Repository,* a component called *Service Mapper* can identify the requirements of a given service and the available implementations. With this information, it is possible to identify the devices automatically that should run a given service or automatically update existing services as soon as a newer version is available. Removing, stopping, and resuming services are also functionalities that the *Service Lifecycle Management* component provides (see requirement 2.3).

Another important functionality of this layer is the ability of identifying the current status of the network, the devices that are available and their capabilities and all the events, errors and warnings that were triggered by the system. These functionalities are provided by several components that together offer a broad possibility to analyze the current situation of the shop floor. The components that support these functionalities are: *Device Manager*, *Device Monitor*, and *Logging & Tracing* (see requirement 2.10).

Given the amount of services, an efficient way of discovering available services is necessary. DPWS offers mechanisms that deal with this challenge like WS-Discovery. Nevertheless, given that legacy systems might also be present, additional mechanisms to discover services and make the accessible are provided by the *Service Discovery and Integration* component.

3.4 Application Layer

The application layer contains both applications that are traditionally strongly connected to the lower layers, such as supervision, as well as applications that do not have until now direct access to those layers, such as business applications (ERP level) or operational management (MES level) like maintenance. The exposure of device functionalities as services and the use of standard commu-

nication protocols allow such applications to access any device on the network seamlessly. Based on device-level web services, business applications are able to invoke services or access device resources directly or through enabling gateways.

3.4.1 Enterprise Services

On top of the middleware layer, several *Enterprise Services* are available to give support to enterprise applications. An example of an *Enterprise Service* is the *Maintenance Control* service, which contains a history of all activities performed on a machine for either maintenance or production. Based on this information and a set of rules, the service can determine when the next maintenance will take place. Additional techniques can be applied to improve the control of maintenance work. For instance, based on sensor readings, a *Maintenance Control* service can predict machine breakdowns and trigger counter-measures (see requirement 2.8).

3.4.2 Business Process Engine

In this sub-layer, business processes are executed and monitored. In addition, the layer deals with the orchestration of utilized services, which can run either in the back-end or on the shop floor devices. Given that the middleware together with the DPWS stack provide a transparent integration of devices and back-end systems, the monitoring and execution of processes using web services becomes possible.

During the modelling of business processes, milestones are defined that indicate stages and rules that the system must achieve to be considered correct. The *Business Process Monitoring* is responsible for analyzing these milestones and generating alerts if an irregular situation occurs (see requirement 2.6). These alerts can be distributed to several management applications including the *Alert Resolution Dashboard*.

3.4.3 Management Interfaces

The supervision of the shop floor is realized through management applications that are placed at the top layer of the architecture. These applications include for example the *Process Efficiency and Risk Analysis Dashboard*, the *Alert Resolution Dashboard,* and further *Business Process Modelling* tools.

The *Process Efficiency and Risk Analysis Dashboard* analyzes the current situation of a process indicating performance issues and providing input regarding risk evaluation. The *Alert Resolution Dashboard* receives events generated in the lower layers of the system and presents it to the process supervisor guiding him through the process of solving the situation. In a machine breakdown situation, this dashboard gives detailed information about the machine, guides the user through the process of requesting appropriate maintenance and also supports requesting a review of the ERP to consider the amount of hours the machine will be unavailable (see requirement 2.7). For cross-layer interoperability, the usage of standards such as the Common Alerting Protocol (CAP) [13] and Emergency Data Exchange Language (EDXL) [5] is implied.

Finally, with *Business Process Modelling* tools, process engineers indicate how their processes interact in order to produce the desired output. They also indicate additional analysis that must be performed during the process to ensure that the output is in accordance with the specifications. Once the process is defined, it is partially deployed to the shop floor and partially deployed into the *Business Process Engine* (see requirement 2.4).

4 Discussion

Most commercially available solutions that have integration between an ERP system and the shop floor assume the existence of an MES system. MESs that explicitly offer extended integration capabilities include the SIMATIC [3] solution from Siemens, and the Integrated Architecture from Rockwell [12]. Our goal is to integrate the production process with the ERP system, but rather have a distributed approach instead of establishing a central MES system. The aggregation of fine-grained device services should be achieved via service orchestration, creating composed, higher-level services that offer functionality at the high granularity ERP systems require. Therefore, all research (business and technical) that describes requirements, economics, and technical aspects of ERP to MES integration, will also be applicable to the "distributed MES", consisting of a set of composed services.

Currently, the widely used industry standard describing an interface for the coupling of ERP systems and MES systems is ISA-95 (also IEC/ISO 62264) defined by the Instrumentation, Systems, and Automation Society (ISA) [1]. It was elaborated by the SP95 working group, which has members of major automation manufactures (like ABB, GE, Rockwell, and Siemens) and major business software vendors (like Microsoft and SAP). It separates business processes from manufacturing processes and focuses on the interaction patterns between them. It lays out a common terminology and an object model of the parties involved in the production process.

The ISA-95 standard offers a high-level view of production integration that leaves many degrees of freedom for the actual implementation. This ambiguity discourages the adoption of the standard and could lead to several incompatible implementations. The standard B2MML (Business to Machine Mark-up Language) [16] defined by the non-profit organization WBF [4] eliminates this disadvantage by describing XML document formats implementing the ANSI/ISA standard ISA-95. It provides a direct serialization of the objects and object attributes described in the standard, which is needed for message exchange.

Both on the lower tiers of the SOA (where device ser-

vices will be composed into machine services) and the middle tiers (where machine services will be composed to MES-like services) the composition should be supported and automated as much as possible. Semantic annotation of the services as described in [8] could both reduce the risk of errors in manual service composition and, if annotation is detailed enough, could also allow for automatic composition of the system at initial setup or recomposition when the system setup is changed (e.g. one device is exchanged by another device from a different vendor).

The OPC-UA [2] is based on widespread web standards, including XML, WSDL, SOAP and some WS-* specifications and provides a homogeneous and generic information model and a set of Web Services to represent and access both structure information and state information in a wide range of devices. Although we have not fully investigated its impact on the proposed integration architecture, it could act as a possible extension of the existing DPWS stack, i.e. as an alternative to WS-Management, as both approaches try to provide a generic protocol to access device resources.

Finally yet importantly, the work on device SOA done in the EU Project SIRENA [5], showed the feasibility and benefit of enabling production devices with web services [7]. DPWS is a standard that enables the devices to express their functionality in a web service-oriented way while taking into account the limited capabilities they have. Initial efforts [6] have shown the feasibility and some slight changes [17] would enhance its functionality even more. The EU-funded project SOCRADES (www.socrades.eu) will further investigate the applicability of DPWS for a wide range of devices and their integration in a service- oriented infrastructure that strongly couples shop-floor and business layer activities.

5 Conclusions and future work

As we move towards the real-time enterprise, the strong coupling of all layers between shop floor and top-floor can lead to increased efficiency. In order to achieve this, the shop-floor devices have to be easily integrated in a service-oriented way with modern enterprise services. We have examined the requirements of a device-to-business integration infrastructure and based on these findings proposed a service-oriented architecture for the coupling of enterprise services with networked embedded devices. The proposed architecture tries to overcome shortcomings of existing frameworks, and allows the holistic integration of legacy and web-service enabled devices, effectively closing the existing integration gap between shop-floor activities and enterprise systems. The application of a service-oriented paradigm allows for an increased flexibility and reusability of device-level functionality, and the utilisation of DPWS simplifies the integration of device-level data into business processes.

Although parts of the proposed architecture have been implemented separately, a challenging task will be to actually implement, evaluate, and refine the proposed integration architecture as a whole in real-world environments, i.e. in the domains of manufacturing and process automation. Non-functional issues such as performance, scalability, communication overhead, and security will need to be closely monitored and evaluated. We plan also to focus more on the integration of non-web service enabled devices in order to enable an easy migration from legacy infrastructures. The role of OPC-UA and its relation to DPWS will be also further investigated. Finally, the information model of the approach needs to be extended and standards have to be used wherever applicable.

6 Acknowledgements

The author would like to thank the European Commission and the partners of the European IST FP6 project "Service-Oriented Cross-layer infRAstructure for Distributed smart Embedded devices" (SOCRADES; http://www.socrades.eu), for their support.

References

[1] Instrumentation, systems, and automation society. http://www.isa.org.

[2] Opc unified architecture (opc-ua) specifications. http://www.opcfoundation.org.

[3] Simatic it for interoperability. http://www.automation.siemens.com/mes/simatic-it/.

[4] World batch forum. http://www.wbf.org.

[5] H. Bohn, A. Bobek, and F. Golatowski. Sirena - service infrastructure for real-time embedded networked devices: A service oriented framework for different domains. In *International Conference on Mobile Communications and Learning Technologies*, April 2006.

[6] François Jammes, Antoine Mensch, and Harm Smit. Service-oriented device communications using the devices profile for web services. In *nternational Workshop on Middleware for Pervasive and Ad-Hoc Computing*, November 2005.

[7] François Jammes and Harm Smit. Service-oriented paradigms in industrial automation. *Parallel and Distributed Computing and Networks*, pages 716–723, 2005.

[8] Ivan M. Delamer and Jose L. Martinez Lastra. Self-orchestration and choreography: Towards architecture-agnostic manufacturing systems. In *IEEE 20th International Conference on Advanced Information Networking and Applications*, pages 573–582, 2006.

[9] Shannon Chan et al. Devices profile for web servicesdevices profile for web services. http://schemas.xmlsoap.org/ws/2006/02/devprof/, February 2006.

[10] E. Fleisch and F. Mattern. *Das Internet der Dinge: Ubiquitous Computing und RFID in der Praxis: Visionen, Technologien, Anwendungen, Handlungsanleitungen*. Springer, Berlin, 2005.

[11] Organization for the Advancement of Structured Information Standards. Oasis standards and other approved work. http://www.oasis-open.org/specs/index.php.

[12] ARC Advisory Group. Rockwell automation process industry strategies. *ARC white paper*, October 2006.

[13] Paulo Leitao, Armando W. Colombo, and Francisco J. Restivo. Adacor: A collaborative production automation and control architecture. *IEEE Intelligent Systems*, 20(1):58–66, January 2005.

[14] David S. Linthicum. *Next Generation Application Integration: From Simple Information to Web Services*. Addison-Wesley Professional, 2003.

[15] OASIS. Web services notification tc. http://www.oasis-open.org/committees/wsn/charter.php.

[16] WBF. B2mml v03 xml schemas and documentation. http://www.wbf.org/associations/2718/files/B2MML-V03-ProductDefinition.do, August 2005.

[17] Elmar Zeeb, Andreas Bobek, Hendrik Bohn, and Frank Golatowski. Service-oriented architectures for embedded systems using devices profile for web services. In *2nd International IEEE Workshop on Service Oriented Architectures in Converging Networked Environments*, May 2007.