Web-Service Enabled Wireless Sensors in SOA Environments

Domnic Savio and Stamatis Karnouskos
SAP Research
Vincenz-Priessnitz-Strasse 1, D-76131, Karlsruhe, Germany
{domnic.savio, stamatis.karnouskos} @sap.com

Abstract

Enterprise applications support business activities in companies, so that they can manage complexity and be more effective. The service oriented architecture (SOA) concepts empower modern enterprises and provide them with flexibility and agility. These concepts nowadays expand towards the shop-floor activities, down to the device level. By implementing web services on the devices natively, we are able to push down at item level SOA concepts. In this work we focus on implementing web services on the SunSPOT wireless sensor nodes and coupling it with enterprise level services. The presented methods could be used as experimental platform to evaluate new SOA-based paradigm for factory automation and enterprise computing. The goal is to seamlessly integrate heterogeneous hardware on the shop floor and the business intelligence via the SOA approach.

1. Introduction

Web services are used mainly in enterprise environments to support interoperable e.g. machine to machine (M2M) interaction while hiding the details of the implementation at each end-point. Enterprise applications use web services as basic blocks to create more sophisticated services e.g. to glue together cross-organizational functionality. Several standards exist, but most of them do not assume embedded systems as an implementation platform. Therefore, when first drafts of implementing web services on devices came up, a consortium led by Microsoft had the home and office automation hardware in mind when they proposed the Device Profile for Web Services (DPWS [13]).

One domain struggling with heterogeneity and proprietary protocols, while thousands of embedded devices exist, is that of factory automation; thus it appeared appealing to try to use web services on devices in this domain [8]. The DPWS was picked up by a number of research efforts, most notably projects such as SIRENA (`www.sirena-itea.org`), SODA (`www.soda-itea.org`) and SOCRADES (`www.socrades.eu`). The key idea is to provide the same interoperability and easiness of integration of devices, focusing exclusively on the functionality they offer at the shop floor and not on the device-specific implementation as such. As a result, any device could be broken down to some basic services depicting its basic functionality used by other devices or entities. These services would then be offered as web services from the device itself, to other devices or entities in higher layers.

It is clear that this approach could create a new paradigm at the shop floor, where integration efforts focus on the abstract functionality offered and are not bound to a specific device. This would encourage not only the development of new devices on the automation industry that have web services embedded [9][17], but would kick-start collaboration at the lowest level i.e. among devices themselves, offering new opportunities and effectively connecting the vendor-locked isolated islands of today.

Bringing such flexibility at device layer is coupled with several non-trivial challenges. One of these is the quest for common understanding of the functionality the services offer as well as the correct interpretation of the offered data. The temperature data that comes from a microwave oven and the same data coming from a temperature sensor mounted on a robotic arm have different meanings and usability based on the context they operate. Not only the consequent actions are different, the speed of the incoming data, payload, time needed to act - all have significant meaning and are mission critical on the shop floor rather than in the home kitchen. As such ontologies describing the produced data, as well as timely evaluation of it is needed.

When different services are then orchestrated to form a complex process flow, WSDL and SOAP make it comfortable to achieve, although these services might have been implemented using different languages. This leads to an increasing number of services that can be realized by relying on very simple ones that are implemented on-device, even on those with constraint resources such as computing capability. As the shop floor gets populated with web services coupled to the functionality of the existing devices, enter-

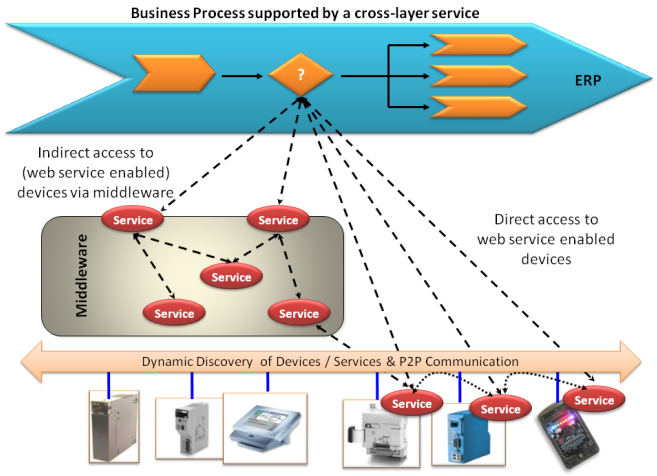

Figure 1. A cross-layer web-service mashup

prise services could easily, directly and on-event basis get critical information needed e.g. to plan the next set of production orders more efficiently.

In this paper, we focus on aforementioned concepts and show in practice how this can be realized. We take wireless sensor nodes and industrial programmable logic controllers (PLCs) and offer some of their services as web services. These services are then used to orchestrate a process flow in which an Enterprise Resource Planning (ERP) software is acting as a consumer of the services to provide business logic. It must be pointed out that our efforts as depicted in the rest of the paper refer mostly to the management plane while control is tackled but only in an indirect manner i.e. no hard real-time control issues are considered.

2. Business Implications

As we are moving towards the "Internet of Things" [18], where millions of devices will be interconnected, provide and consume info available on the network and cooperate, new capabilities are opened. As these devices need to interoperate, the service-oriented approach seems to be a promising solution i.e. each device offers its functionality in a service-oriented method, while in parallel it is possible to discover and invoke new functionality from other services on-demand. By considering the set of intelligent system units as a conglomerate of distributed, autonomous, intelligent, pro-active, fault-tolerant and reusable units, which operate as a set of co-operating entities, a new dynamic infrastructure that is able to provide a better insight to its components to the higher levels and flexibly react to dynamic business changes can be realized.

The convergence of solutions and products towards the SOA paradigm adopted for smart embedded devices, contributes to the improvement of the reactivity and performance of industrial processes, such as manufacturing, logistics and others. This will lead to information being available as it happens, on an event-driven basis, and in business-level applications that are able to use high-level information for various purposes, such as diagnostics, performance indicators, traceability, etc. These future vertical integration capabilities will also help to reduce the effort required for integration of the affected systems in the sense of the given business scenario.

We assume that in the future, web services might be the common denominator everywhere. As such, the factories will transform towards web service mashups, where composed services can be created in a cross-layer way. Business processes running at ERP level, will be able to interact and take timely decisions with services running at enterprise, network or even device level. This will effectively lead to the realization of high-resolution enterprise, increasing not only the visibility in all layers but also the collaboration among them.

As depicted in Figure 1, the functionality of multiple devices will be available either as a service abstraction via a middleware (classical connectivity efforts today) or by providing their functionality as a set of web services themselves. Furthermore it will be possible to communicate in a peer-to-peer way among them. Sophisticated services can be created at any layer (even at device layer) taking into account and based only on the provided functionality of other entities that can be provided as a service.

Having such a dynamic infrastructure has a significant effect on the way we design and implement enterprise services and applications. Increasing amount of information coming from the shop floor can enhance the efforts such as remote diagnosis, maintenance etc. Actions such as starting, stopping and pausing automation devices are also entering the shop floor [2, 3]. With the increased usage of pervasive and ubiquitous computing devices, automation in the shop floor is getting more intelligent, IP-networked, and hosted services report events happening in the their environment to subscribers on remote locations [4]. As a result, software applications like the Enterprise Resource Planning (ERP) which reside normally on the top of the automation IT pyramid, could benefit from real time events on the shop floor and provide meaningful timely reports that could be used to plan production more accurately [1, 3, 5]. This adds more flexibility and supports the on-demand nature of the market.

To execute and maintain devices on the shop floor, Manufacturing and Execution Systems (MES) play a dominant role, often acting as a middleware gateway, reporting directly to the enterprise resource planning software such as the SAP ERP. Due to globalization and high demand of flexibility, enterprise applications change their building blocks

by breaking them into further small components, empowering them with service oriented paradigm and strongly gluing them (e.g. the SAP portfolio is based on enterprise SOA approach). As the trend shifts more to SOA, with devices in the shop floor hosting a variety of services, the business software components benefit themselves by consuming information from the shop floor in a timely manner.

3 Web Services on Devices

In the past there have been efforts (e.g. Jini, UPnP) to integrate devices into the networking world and make their functionality available in an interoperable way. The latest one, coming from UPnP and attempting to fully integrate with the web-service world, is DPWS, which defines a minimal set of implementation constraints to enable secure web service messaging, discovery, description, and Eventing on resource-constrained devices. DPWS is an effort to bring a web services on the embedded world taking into consideration its constrained resources. Several implementation of it exist in Java and C (`www.ws4d.org`, `www.soa4d.org`), while Microsoft has also included a DPWS implementation (WSDAPI) by default in Windows Vista and Windows Embedded CE.

In the SOCRADES project, we have used it on automation devices like programmable logic controllers (PLC) and on the SunSPOT wireless sensor nodes. Although our efforts are based on implementing it natively at device level, or creating a gateway for devices that cannot host it due to resource mangle, an appealing idea is also to consider implementing the DPWS Stack on an ASIC chip as a hardware. The DPWS stack supports the following Web Service Standards: WSDL 1.1, XML Schema, SOAP 1.2, WS-Addressing, WS-MetaDataExchange, WS-Transfer, WS-Policy, WS-Security, WS-Discovery and WS-Eventing. As a result, dynamic device and service discovery can be realized, while the metadata exchanged can provide detailed information about the devices and its functionality. This is well supported in DPWS with the inclusion of the main data discovery and transfer protocols such as WSDL, SOAP, WS-Transfer etc. Therefore not only custom made device drivers can be eliminated to a large extend, but also these devices can now be easier and better used by ERP applications via widely used technologies such as web services.

As the shop floor gets rapidly populated with SOA-ready devices, to manage them is a fluctuating task load on the ERP system, depending upon the number of devices and their reporting frequency. Another important task is to provide the ERP system with meaningful business relevant data and not raw info e.g. just temperature information in a tag format. We have developed DPWS-based extensions to the SAP Manufacturing Intelligence and Integration (MII) product, that realize exactly this part of subscribing to devices on the shop floor that support web services and collecting data from them. MII then encapsulates the data with proper syntax that can be clearly understood by SAP ERP products like Supply Chain Management (SCM). This offers the shop floor manager a comprehensive overview about which production order is getting executed on a particular machine and the status of components on that machine. Moreover correlated information can be extracted from manufacturing analytics and performance factors are provided by the SAP MII [10].

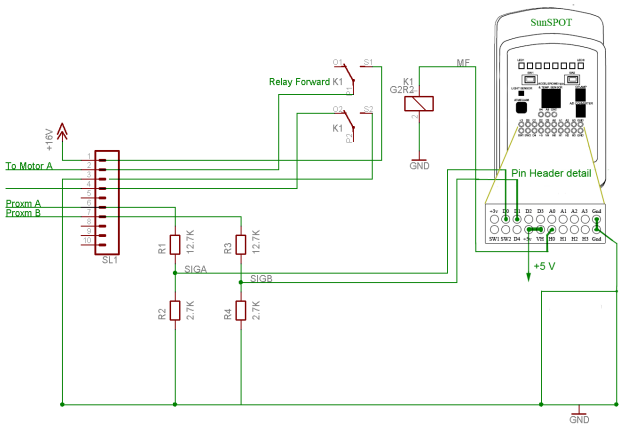

Figure 2. SunSPOT acting as a gateway controlling other devices

Sensor Networks are seen as one of the most promising technologies that will bridge the physical and virtual worlds enabling them to interact. This effectively leads to the avoidance of media breaks especially between the real and the enterprise world. Sensors not only have information about themselves but more importantly about their environment. In conjunction with the actuators, they can also act on their environment, controlling/managing it. We also witness a high number of sensors not as standalone devices but as part of more complex machines. As such it is expected that in future infrastructures at work, home, factory etc, thousands of sensors will be available, eventually being able to sense and act with respect to their context and capabilities. As the sensors get Internet access (via a gateway or the complex device that hosts them), they can be easily coupled with enterprise systems, eventually allowing them to monitor and manage almost in near real-time the infrastructure. This brings us one step closer to the so called ”real-time” enterprise that minimizes or avoids any media gaps, and is agile. The soft “real time” data acquisition and evaluation in this context offers transparency to the shop floor by integrating the devices or process execution systems and the ERP Connectivity frameworks like SAP MII

which act as middleware systems.

4. Experimental Setup

For the integration of web services and wireless sensor networks in this paper, we used the SunSPOTs from Sun Microsystems [11]. The SunSPOTs consist of a base station which is terminated to a normal PC and connected to it via an USB port. Several remote nodes use this base station to interact with applications running on the PC and get Internet access via it. The remote node hosts a set of sensors such as temperature, light and accelerometer. It also has 8 GPIO pins and a set of high current drivers to turn on 125 mA relays. As such SunSPOTs can be used as controlling devices for more simple ones e.g. proximity sensors etc and act as a gateway for their functionality. As a proof of concept for the gateway role of the SunSPOT we have integrated it with a servo motor and two proximity sensors, all of which (including the native functionality of the sensors e.g. temperature sensor) are offered by the SunSPOT as separate web services to any other entity.

As depicted in Figure 2, a SunSPOT remote node was connected to a servo motor over a 6V DC relay. The motor needs a 6VDC and 220mA. Since this was a heavy load for the battery operated SunSPOT, separate power supplies were given to the motor and the power was switched on using a 6V relay which can handle an input voltage of 2.5V. The wire connecting the SunSPOT and the relay was terminated to the high current pin of the demo board connector. To enable the high current output the pins +5V and the VH have to be shorted to enable the high voltage output pin of the ARM. In this way, setting the H0 pin would energize the relay and trigger the output terminals that would connect the power to the motor. Hence the motor could be turned on by the SunSPOT. The motor could be also made to turn in the reverse direction if the power terminals are interchanged.

To assist a controlled behavior, a Pepperl-Fuchs NPN proximity sensor was also terminated to the input pins of the SunSPOT. The proximity sensor is activated when a metal object is present near its surface. This sensor is powered by a separate power supply and his output was sent to a voltage divider. The voltage divider makes sure that the output voltage does not exceed 2,5V DC which is still under the maximum limits of the SunSPOT input pin. The programmable logic hosted on the SunSPOT assumes that when H0 is turned ON, the input pin D0 to which proximity sensor is terminated is constantly checked for voltage. When voltage is present on this pin, H0 is immediately turned down. This could be used in a scenario of a conveyor belt where, the conveyor could move on the torque of the motor and when the object to be moved had reached the destination, the sensor could turn off the conveyor.

We have shown how a SunSPOT can act as a gateway for other devices and sensors; now we will focus on showing how they can cooperate with other devices and be integrated in enterprise scenarios. To demonstrate this, a remote SunSPOT is mounted on a robotic arm and is programmed to report the temperature of it to the base station over the wireless link telemetrically. An industrial based programmable logic controller (PLC) from Schneider Electric, controls the movement of this robotic arm. The PLC is terminated to the network over the Industrial Ethernet connection to a normal PC which hosts web services that encapsulate the commands that the PLC needs to control a process. The base station is also enhanced with web services to subscribe and report temperature information from the remote SunSPOT. As such both the wireless sensor and the PLC appear as web service enabled devices on the network, offering their functionality as such.

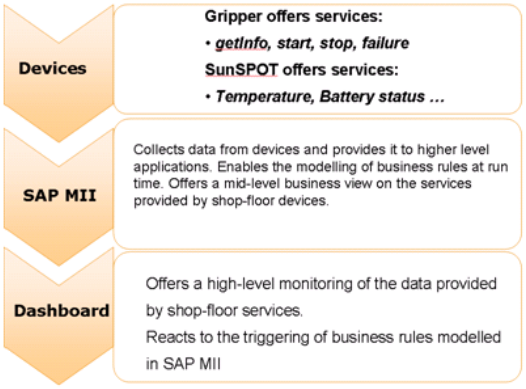

Figure 3. Components and their interactions vertically

5. Middleware Integration

The functions of individual components are shown in Figure 3. DPWS clients discover the devices using WS-Discovery. Once the devices on the shop floor are discovered, the WSDL which describes the services and operations hosted by the device servers are available. Using the WSDL, service proxies were developed to consume these services. For this experiment, certain generic interfaces like start, stop, reset and failure (as event) were defined on the device server. These operations invoke the service and then write data to a network predefined database table. For example, the event, failure is initiated by the device server. The client subscribes to this event and upon receipt it writes this event to the database with the time stamp when the event was received. The database is visible in the SAP MII system which acts as a middleware between the devices and the ERP System. There is a latency in milliseconds here

from the time when the actual event had occurred till, the time it was registered by the client in the database. This is due to the fact that in the meantime the event goes thru a series of steps like, construction of the SOAP message, identifying subscribers and sending it to the end point references. The server would in turn need to decode the SOAP message and call the appropriate function which consumes the event and in this case it is a write to the database. During the whole process, there are several non-deterministic factors involved like, network traffic, SOAP construction, size of the SOAP Message and the bandwidth of the network, processing of the SOAP at the client and retrieval of the data from the SOAP message. In between the data from the devices have to be converted from binary to ASCII. This time interval which contributes a major part to the latency of the events and service invocation can be reduced by standardizing transfer protocols like SOAP for embedded devices. Binary SOAP is a good candidate for further experimentation as its applicability in other domains shows [14]. Once the data is available in the database tables, they can then be fetched by the SAP MII that queries the database for any events or updates from these devices periodically e.g. each 1 second. The time for doing a polling was further enhanced by deploying an http call to SAP MII from the DPWS Client once the event was received (so we don't have to unnecessarily poll the DB if no events occur). Other standards e.g. OPC DA are widely used as current practice in the industry.

The use of wireless devices gives the possibility to place the devices remotely and monitor their performance even under hazardous conditions often seen on the shop floor. The overview of the software components and their associated hardware is seen in Figure 4. In this experiment only a few devices were considered. However in previous work approx 25000 devices were simulated and integrated in the similar fashion to evaluate the performance of device discovery and service invocation [16]. Increasing the number of devices on the shop floor would increase the complexity of managing the services hosted on the devices. To handle this complexity WS-Management could be used. The specification is based on DMTF open standards and Internet standards for web services [15]. WS-Management uses SOAP to manage systems that include web services.

The software used to program the SunSPOTS is written using Java on the NetBeans IDE (`www.netbeans.com`). The SunSPOT development kit also comes with a SDK, that provides APIs that ease the access to the GPIO ports and the temperature sensor e.g.:

```
private ITemperatureInput tempSensor =
EDemoBoard.getInstance().getADCTemperature();
```

The acquired data can then be transmitted using methods like,

```
Radiogram rdg = xmit.newDataPacket(PING_REPLY);
rdg.writeInt(linkQuality;
xmit.send(rdg);
```

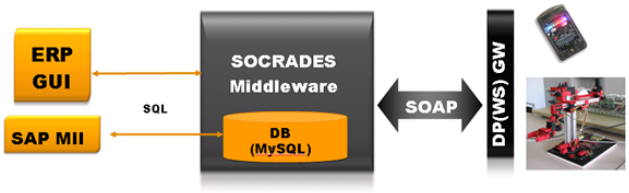

Figure 4. Overview of the Architecture using SunSPOTS and PLC

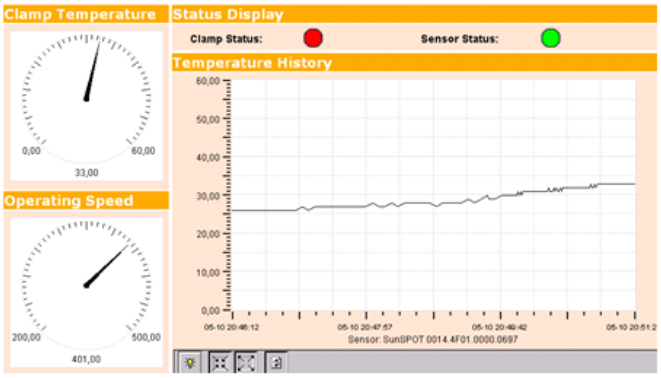

Figure 5. The dashboard at the SAP MII

Where, `PING_REPLY` is a telemetry command that both the base and remote nodes can understand.

6. Services at Enterprise Level

The actual sensing algorithm on the SunSPOTs is written using Java. The web services are deployed on top of the base station class files on a host PC. Since the web services are exposed over the network and accessible using SOAP messages, SAP MII should communicate to web services by constructing SOAP messages to make requests and parse them to read the responses from them [6]. Hence a middleware to discover and subscribe to these web services was developed, that also constructs the SOAP messages to make invocation and parses them when events are received. The received data is then deposited in the database which is then queried by the SAP MII. In this way, the SAP MII is fully aware of the changes happening on the shop floor. Any faults or completion of tasks are updated as status messages to the SAP MII. Figure 5 shows a snapshot of the graphical user interface on the shop floor.

The temperature sensor from the SunSPOT periodically reports the temperature information to the base station which is then sent to the Web Service client. This data is then retrieved by SAP MII and then compared if this data is beyond certain critical value (according to the business

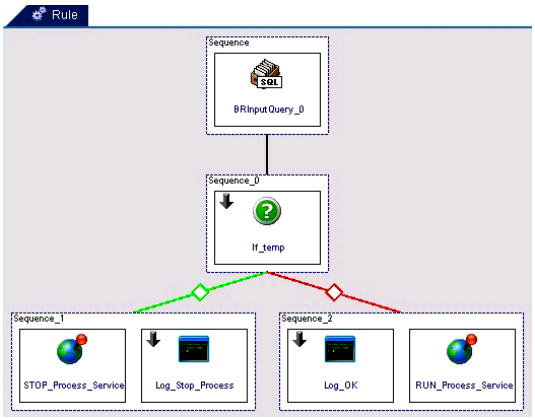

Figure 6. Business rule modelling in SAP MII

logic running in SAP MII). If the data is beyond a critical value, it writes back to the database to stop the whole process as the temperature is too high. The web service client then constructs a SOAP message and sends to the PLC to stop the operation of the robotic arm. SAP MII provides this possibility of modeling the business rule using its Business Logic Editor. A snapshot of the specific business logic modeled in this editor Figure is depicted in 6

The scenario described here can be easily extended to any complex automation process. ERP applications like Supply Chain Management (SCM) Production Planner can use the status of the shop floor to plan further new orders coming from different locations of its sales organization. Hence a distributed SOA architecture can enhance the production planning process of a multi-located manufacturing facility with ”just-in-time” data from the shop floor. Additionally the services in shop floor are also correlated in SAP MII with business data so that the shop floor manager is fully aware of which production order is currently being executed and which ones are on the pipeline. Further manufacturing analytics can be performed on shift bases to identify performance of different locations on near real time basis.

7 Final Thoughts and Conclusions

Providing web services on the shop floor is effectively bringing the business logic closer to it . The use of sensor data on the shop floor serve as parameters to make timely decisions on the next operation of the processes with or without human intervention. As such visibility in the whole enterprise is increasing, as now events happening are quickly propagated to the relevant services and applications that affect them.

However, when the number of devices and services increases on the shop floor, it is often difficult to have an overview on them. Increased communication may result to difficulty into subscribing to a particular set of events. Furthermore if too many entities subscribe to services offered by a device, this can lead to bottlenecks and overloading of the device itself. Scalability needs to be checked; especially due to the fact that we usually deal with embedded hence resource constrained devices. Semantics pose another significant challenge, as the understanding of the data delivered by web service enabled devices needs to be automated and universal. Additionally these services have to be governed and regulated to reflect the process flow of an operation which calls for more research in this area.

As of today, most sensors are connected via a gateway to Internet applications. However, with the emergence of IP as a common denominator, it is a matter of time until it is fit for embedded devices. IETF's ongoing work on IPv6 over Low power WPAN (6lowpan) [19], tackles exactly this category of devices. Extreme low power (such that they will run potentially for years on batteries) and extreme low cost (total device cost in single digit dollars, and riding Moore's law to continuously reduce that price point) are seen as essential enablers towards their deployment in networks. Having these devices offering their functionality via a web-service interface would lead to a new generation of device-aware applications. Up to now ArchRock (`www.archrock.com`) and Sensinode (`www.sensinode.com`) have implemented 6lowpan stacks in their sensors, while this is expected to be also available in the future for SunSPOTs.

Being able to directly access via a unified way e.g. web services over 6lowpan any embedded device, and couple it directly to enterprise services, opens new opportunities for enterprise applications, but also major challenges that need to be addressed before these concepts become widely commercially available.

8 Acknowledgments

The authors 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 - www.socrades.eu), for their support.

References

[1] Luciana Moreira Sa de Souza, Patrik Spiess, Moritz Koehler, Dominique Guinard, Stamatis Karnouskos, and Domnic Savio, “SOCRADES: A Web Service based Shop Floor Integration Infrastructure”, Internet of Things 2008 Conference, March 26-28, 2008, Zurich, Switzerland.

[2] J. Bollinger et al, "Visionary Manufacturing Challenges for 2020", National Research Council Report, National Academy Press, Washington, D.C., 1998.

[3] Manufuture: A vision for 2020, Report of the High Level Group, November 2004, Directorate-General for Research, European Commission, Brussels, Belgium.

[4] F. Kordon, L. Pautet, "Toward Next-Generation Middleware?", IEEE Distributed Systems Online, Vol 6(3), March 2005.

[5] D. Lea, S. Vinoski, W. Vogels, "Asynchronous Middleware and Service", IEEE Internet Computing, Jan-Feb 2006, pp. 14-17.

[6] P.T. Eugster, P.A. Felber, R. Guerraoui, A.-M. Kermarrec, "The Many Faces of Publish/Subscribe", ACM Comp. Surveys, 35(2), 114-131.

[7] H.R. Motahari Nezhad, B. Benatallah, F. Casati, F. Toumani, "Web Services Interoperability Standards", IEEE Computer, May 2006, pp. 24-32.

[8] F. Jammes and H. Smit, "Service-oriented paradigms in industrial automation", IEEE Transactions on Industrial Informatics, 1:62-70, 2005.

[9] Automatic device detection using Microsoft Windows Embedded CE 6.0 R2 and Beckhoff technology, Beckhoff News, 16th November 2007, http://www.beckhoff.de/english.asp?press/news0207.htm

[10] SAP Manufacturing Integration and Intelligence, http://www.sap.com/solutions/manufacturing/manufacturing-intelligence-software/

[11] Sun Small Programmable Object Technology (Sun SPOT) Theory of Operation, Part No. 820-1248-10, Revision 1.0.8, May, 2007

[12] J.L. Martinez Lastra, "Reference Mechatronic Architecture for Actor based Assembly Systems", Doctoral Thesis. Tampere University of Technology, Finland, 2004.

[13] S. Chan and C. Kaler and T. Kuehnel and A. Regnier and B. Roe and D. Sather and J. Schlimmer and H. Sekine and D. Walter and J. Weast and D. Whitehead and D. Wright, "Devices Profile for Web Services", Microsoft Developers Network Library, May 2005, http://specs.xmlsoap.org/ws/2005/05/devprof/devicesprofile.pdf

[14] U. Niedermeier, J. Heuer, A. Hutter, W. Stechele, and, A. Kaup., "An mpeg-7 tool for compression and streaming of xml data", in proceedings of the 3rd IEEE International Conference on Multimedia and Expo, pages 521-524, 2002.

[15] WS-Management Specification, http://www.dmtf.org/standards/published_documents/DSP0226_1.0.0.pdf

[16] Stamatis Karnouskos, Mian Mohammad Junaid Tariq, "An agent-based simulation of SOA-ready devices", 10th International Conference on Computer Modeling and Simulation, 1-3 April 2008, Cambridge, England.

[17] WSD: Plug-and-play for building automation, Beckhoff News, 9th May 2008, http://www.beckhoff.de/english.asp?press/news0408.htm

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

[19] N. Kushalnagar and G. Montenegro and C. Schumacher, "IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs): Overview, Assumptions, Problem Statement, and Goals", RFC 4919, IETF, Aug 2007, http://www.ietf.org/rfc/rfc4919.txt