Adaptation of Resource-oriented Service Technologies for Industrial Informatics

Steffen Prüter[1], Frank Golatowski[2], Dirk Timmermann[1]
[1]University of Rostock, 18055 Rostock, Germany
[2]Center for Life Science and Automation, 18119 Rostock, Germany
{Steffen.Prueter | Dirk.Timmermann}@uni-rostock.de Frank.Golatowski@celisca.de

***Abstract*-In the domain of industrial automation the interoperability of devices gains importance. One major trend is the usage of Ethernet based solutions to connect devices at all levels of a company. Through a common interface technology in all levels of execution, a central control mechanism or business flows can control an entire workflow. This paper describes a new RESTful oriented device architecture, called RODA. RODA will be compared against Web Services based service-oriented devices architecture. It describes how Web services technologies can be adapted to fulfill the requirements of automation, embedded, and real-time domains. The development process in these domains can gain an advantage, with the usage of middleware components based on well-known internet technologies. Furthermore, the created interfaces can be easily reused in other applications and merged with other interfaces to combine a bundle with an additional benefit. The applicability of RODA has been verified on a mobile robotics testbed, where robots are communicating via different wireless interfaces.**

I. Introduction

Industrial informatics plays an increasing role in a wide application area. The field reaches from the automation domain with industrial plants to the home domain with robotics and home automation. The amount of networked embedded devices is steadily increasing. The huge numbers of devices which can be involved in a single application require a new kind of adaptable interfaces. Today, some large manufactures of automation solutions already provide technologies for these issues [Simatic, OPC-UA], which are often based on proprietary protocols and specific hardware. Furthermore, they are mostly not accessible without using a complete solution from one manufacturer. These solutions are incompatible with solutions from other manufactures and do not allow a common interface technology in all domains. Due to these issues there is no technology available that can be used generally. Only some manufacture driven solutions dominating the market, which causes significant extra cost to provide gateway or proxy concepts to bridging the gap between products from different manufacturers.

Web-oriented technologies claimed as state of the art to connect business execution layers as well as networking devices. For the machine to machine communication Web services (WS) will become the most important technology. Thus, it is easily possible to encapsulate functional blocks and reconnect or bundle them with standardized interface descriptions. The task processing is no longer bounded to local machines, the interfaces can also create a world-wide structure. Additionally, the code verification and the fault-tolerance of software can be managed in a new manner. Separated software blocks can be verified and tested, not only complete applications. Based on these interfaces already verified software can easily reused in other scenarios and also be provided by third parties. The software development process can be much finer divided and better planned. Further advancements are the abstraction of complex communication infrastructures, so the developers can focus on developing applications and not solving communication or interoperability issues between devices and applications.

For the adaptation of Web services technologies to fulfill the requirements in the automation domain some general points need to be solved. The Web services technology abstracts from the used hardware. This is not possible in the automation domain with the actual *message size* and *timing requirements*. Web services are based on TCP/IP and have no *real-time capabilities*, because of the fundamental resend algorithm are not deterministic, if errors are appearing. Furthermore, also small failures in the automation systems have influence to the proper work of the overall system. Often complete production lines need to be stopped, which causes high maintenance costs. Thus, Web services in the automation domain must grant *reliability*. For the integration of all company layers also a *seamless interconnectivity* is very important.

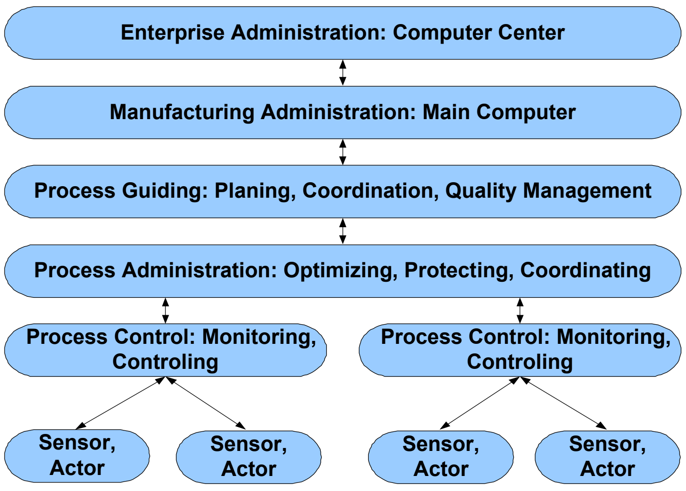

Enterprise Administration: Computer Center
Manufacturing Administration: Main Computer
Process Guiding: Planing, Coordination, Quality Management
Process Administration: Optimizing, Protecting, Coordinating
Process Control: Monitoring, Controling
Process Control: Monitoring, Controling
Sensor, Actor
Sensor, Actor
Sensor, Actor
Sensor, Actor

Figure 1: Hierarchical Process Layer Structure

This paper gives an overview on universal communication technologies in the automation domain. The presented solutions are compared and conclusions for an own software architecture are proposed. This architecture is based on well-known internet protocols and standards. Further it uses only Open Source or public domain technologies so this usage is free of additional costs. Thus, the further development of these architectures is granted because of the wide Open Source community and the participation of industrial companies. As a brief conclusion the paper shows, based on an example scenario, the different messages sizes and timing behavior for selected key technologies to allow an overview of the state of the art in the automation domain.

II. Related Work and Technologies

In this chapter a brief summary of the state of the art for communication frameworks in industrial automation technologies is described. Also the problems for the integration of service-oriented architectures in the automation domain are highlighted.

A. *Distributed Component Object Model*

Today a common technology for interconnecting devices and business process structures is the Distributed Component Object Model (DCOM) [4]. An example is the SAP DCOM Component Connector [5], which is a remote procedure call (RPC) system basing on the Distributed Computing Environment (DCE) Standard. This technology has strong limitations, but only *specific operating systems* can apply this model. Furthermore, it has *barely security aspects* because the heavy implementation does not apply in the embedded world.

B. *Extensible Markup Language Remote Procedure Call*

One further technology is the Extensible Markup Language Remote Procedure Call (XML-RPC) [6] which is called a forerunner of Web services. XML-RPC is easy to understand but has only a few features implemented. The most important problems in the domain of industrial informatics are the *lack of device discovery and eventing mechanisms*.

C. *Common Object Request Broker Architecture*

The Common Object Request Broker Architecture (CORBA) [7] is an advanced technology, which is similar to Web services. The major differences of CORBA are the need of an Object Request Broker (ORB) library and the communication with the General Inter-ORB Protocol (GIOP), which is blocked in the most firewalls. Furthermore, this architecture is *very complex* and the different implementations (COM on Windows or J2EE on Java) that cannot fulfill the *requirements of embedded devices*.

D. *D-Bus*

As an advancement of the complex CORBA the D-Bus [13] messaging bus system was developed. D-Bus is a lightweight interprocess communication mechanism designed for *local desktop environments* and system messages. The Universal Plug and Play (UPnP) [8] specification is applicable for *small networks only*. The lack of security handling and the missing service proxy make Web services technologies preferable, when using larger networks or thin devices.

E. *OPC Unified Architecture*

The OPC Unified Architecture (OPC UA) [12] is a wide-spread technology in the automation domain, but it has very limited capabilities for interconnectivity with other application domains. The old DCOM [4] based interface is replaced with Service-oriented Architectures (SOA). OPC UA implements two interfaces to solve the issues between rich client Web services applications and automation requirements. On the one hand a limited and strict defined binary protocol without XML parser and on the other hand a Web services SOAP interface. This solution does not solve the problem rather then it works as a *standardized gateway* between these two domains.

F. *Devices Profile for Web Services*

The Devices Profile for Web Services (DPWS) [2] is a new standard, which could be the next state of the art in the industrial domain. DPWS has been standardized within OASIS WS-DD Web Services Discovery and Web Services Devices Profile technical committee together with Web Services Dynamic Discovery (WS-Discovery), SOAP Over User Datagram Protocol (UDP). OASIS is still working to extend these standards. DPWS is based on several Web services standards defined by the World Wide Web Consortium (W3C) [9]. The Web services protocols using SOAP [10] for the data transportation and the Extensible Markup Language (XML) for data representation. The WS protocols are designed for rich client applications over IP based networks. Therefore, Microsoft defined the Devices Profile for Web Services (DPWS) [11].

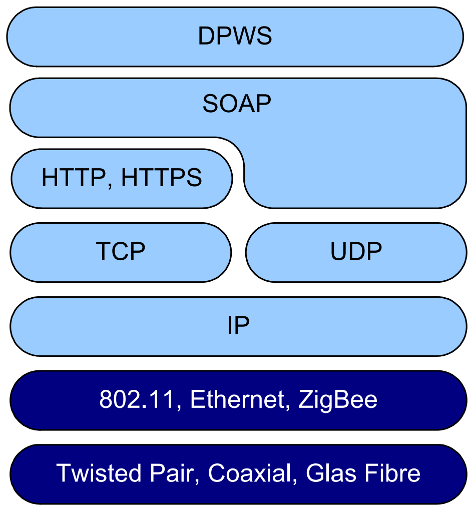


Figure 2: DPWS Protocol Stack

DPWS was developed to enable secure Web service capabilities on resource-constraint devices, using only necessary WS protocols. Additionally to the loose coupling of Web services, the useful WS-Eventing algorithm and the dynamic discovery of devices is specified. DPWS can be used for inter-machine communication. However, the latter requires the devices to feature peer functionality as well as a specific DPWS client implementation, in order to use the corresponding services hosted on other devices. It employs similar messaging mechanisms as the Web Services Architecture (WSA) with restrictions to complexity and message size.

DPWS is integrated in Microsoft's operating system Vista. For many companies, this was the reason for developing new interfaces for their products basing on this protocol. Even in Automation Industry the infiltration of Service-oriented Architectures like DPWS is already going on.

The usage of DPWS enables a device-centric SOA on the level of devices and embedded systems. This has several advantages, such as providing plug-and-play capability for networks devices, fault-tolerant services, and standardized interfaces. Furthermore, in Figure 2 the IP based protocol stack with HTTP connections is shown, which allows the usage in a various environments. Thus, the wide-spread Ethernet based networks can be used, also when firewalls are installed. Due to the fact that the most firewalls do not block HTTP traffic for the usage of the World Wide Web or company based intranets.

The OASIS technical committee has started in August 2008 to standardize the "Web Services Discovery and Web Services Devices Profile" (WS-DD) [1]. This work is based on the former DPWS specification, which was defined by Microsoft, Intel, Ricoh, Lexmark, and other companies in 2004. WS-DD defines a lightweight subset of the Web Services protocol suite that will make it easy to find, share, and control devices on a network. In the midyear 2009 the WS-DD standard v1.0 should be published, which allows manufactures to create interoperable interfaces and also attend the requirements of embedded devices.

The DPWS toolkit stack used in the following scenarios is a work of the WS4D Initiative [16]. It allows a fast and easy development of services based on the upcoming WS-DD standard. Enhancements for the applicability in the automation domain are described in further papers of the authors [17][18][19]. All described solutions have the drawback of getting incompatible with the DPWS standard or they cannot grant real-time capabilities, because the required TCP interface during the discovery process.

III. Resource Oriented Device Architecture

The Resource Oriented Device Architecture (RODA) is based on the Representational State Transfer (REST) work of Roy Fielding [14], the dissertation of Andreas Bobek [20], and the thesis of Thorsten Schulz [21]. The REST architecture has several advantages, because of this it is used as basis for RODA. It is a similar architecture than the World Wide Web (WWW). In this system each data is handled as a resource and each resource is an atomic data unit. Only a strong limited number of methods for the manipulation of a resource are specified, and each method works in a fixed functionality. The two most common methods are GET for the request of a resource and POST for the set of a resource. Furthermore, REST is a stateless architecture. This means, each resource can be handled and manipulated over different states and these states must be handled by the service user. Therefore, the implementation of a service can be more simple and lightweight than in other architectures. REST is not developed for machine to machine interaction, because of this some parts are missing that are necessary for Service-oriented architectures.

A. Discovery

Each service must announce itself when joining a new network, also a service must respond to search request that fit to the service description of the device. This functionality is important for a dynamic service infrastructure and the seamless connectivity of devices.

The RODA architecture using the established Zeroconf [22] architecture to obtain IP addresses and find and announce services in the local subnet. The main advantages of Zeroconf are:

- Using well designed and open source protocols for name resolution and address allocation.
- Allow service announcement and discovery without central service broker.
- Already widespread usage and different implementations, especially in the printer domain.

B. Eventing

In the automation domain it is important that services provide an eventing mechanism. Modern automation facilities have a lot of sensors, but the sensor data are often rarely required only.

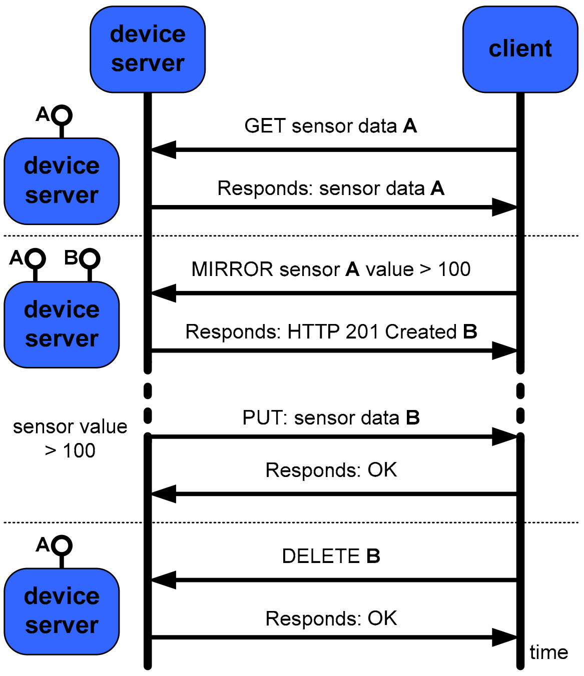


Figure 3: Messages between device/server and client in GET and Eventing scenarios for RODA

As Example a sensor value is needed only when it pass over a specified security level, in other cases no information from this sensor are required. Eventing mechanisms allow a client the subscription of an event with specific roles. If this roles match an event is fired, than the sensor directly informs the subscribed clients about the event. Thus, the clients do not need to poll the sensor in periodic intervals to stay up to date. Therefore, the network load is reduced and the client computing resources are preserved.

The concept of a communication initiated by the device/server to the client is new in the domain of web-oriented and resource constrained architectures. Therefore, the RODA architecture using the new MIRROR concept, described in the dissertation of Andreas Bobek [20]. The message flows for a simple data request and the subscription of an event are shown in Figure 3. Basically this concept introduces two new commands based on the well-known HTTP methods for the WWW. The first command is MIRROR, which is send from a client to a device. This command includes the roles when the client wants to be connected from the device. The device creates a new resource for each subscription, which allows the control of the subscription. The second command is the reuse of PUT, which is in this case send from the device to the client. This command is equally to the GET responds in a normal sensor data request, except that it is the start of a communication and not the answer. The answer in this communication is an acknowledgment from the client.

C. *Description Language*

Dynamic application generation at runtime is possible with a universal description language only. Each device and client must be able to communicate with each other, also when the at the development time of one part the other part not known. The description language explains how to use a client or device and which functionality it has.

The RODA architecture specifies the usage of the Web Application Description Language (WADL) [23] as description language. The java.net project has published this specification in 2006 but it is still in development. WADL allows the same work for REST frameworks like WSDL for Web-oriented services. It is easy extendable, allows modularization, and the automated code generation for the RODA architecture.

D. *Overall architecture*

In Figure 4 the general protocol stack for the RODA architecture is shown. Like Web-oriented architectures all communication is handled by an IP network, which is very common in a lot of new technologies and also in modern automation facilities. The REST part is responsible for the data exchange, the MIRROR part for Eventing, and Zeroconf for the dynamic and seamless discovery.

For the comparison of the different communication technologies and the new RODA architecture an example scenario with mobile robots is used for the evaluation. DPWS and RODA are implemented, because DPWS seems to be the most promising technology for new automation technologies to challenging with. The scenario describes which respond times, message sizes, and data processing is analyzed and what are the important points.

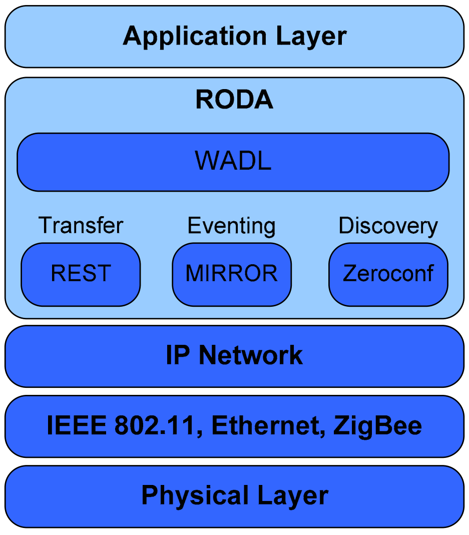


Figure 4: RODA Protocol Stack

IV. Example Scenario

A. *General Scenario Setup*

The mobile robot platform is part of the international RoboCup event, in the Small Size League (SSL) teams with five autonomous mobile robots playing soccer against each other. The robots are controlled by an external PC and a camera above the field. For the control loop of the robots the latency time from capturing the image to the execution of the commands on the robot is the most important part. To avoid collisions between the robots and allow an accurate movement it is important that new commands can be executed on the robot in predictable timeslots. The overall control loop is shown in Figure 5.

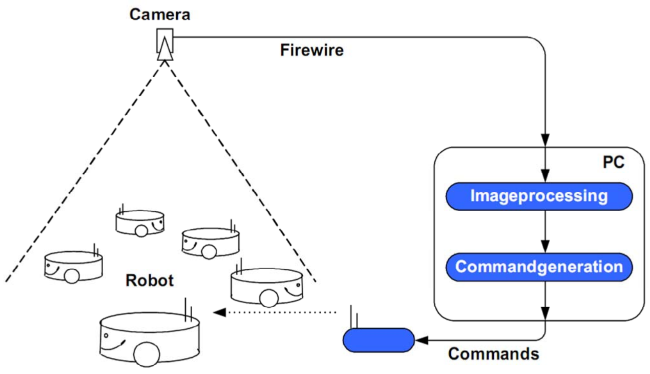


Figure 5: RoboCup Example Scenario Global Control Loop

The control loop starts with the capturing of an image of the field. This image is send to a PC outside the field. Image processing software extracts the position and alignment of objects on the field. Based on this information commands for each robot of the own team are generated and send to the robots. The used communication technology can be switched between the wireless WLAN, Bluetooth, and ZigBee on the one hand and on the other hand wire based Ethernet. The robot commands are virtual target coordinates and the actual global robot positions. The robots use the global coordinates to update their own local and imprecise coordinate tracking. Therefore, the actual positions send with the highest priority and in defined timeslots. If the robots cannot correct the local movement the robots calculate incorrect positions, because of slip and friction. This result in an uncontrollable status and the robot must be stopped to avoid collisions.

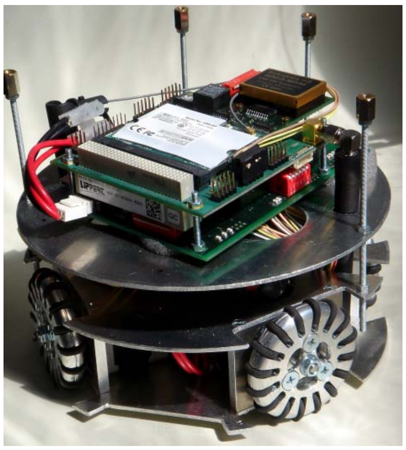


Figure 6: RoboCup SSL Robot

These real-time requirements with a parallel running communication system make the robot scenario an ideal test bench for our implementations. Furthermore, the main board is designed as a sensor network gateway. Without the robot hardware it is used to collect data from sensor nodes, over plain radio communication technologies. Thus, the power consumption of a robot that do not drive is also very important in the following tests.

B. Robot Hardware

A robot without chassis is shown in Figure 6. The robot is equipped with an embedded Linux board and an ARM7 controller board. The embedded Linux board is a Cool Mote Master (CMM) from LiPPERT, with an AMD Alchemy AU 1550 processor [15]. The CMM handles the communication and high level robot behavior, the ARM7 controlling the motor drivers. The additional ARM7 board is only necessary because of the special stepper motors, which need a high-frequency pulse-width-modulated controlling signal. The possible communication interfaces for the robot are WLAN, Bluetooth, and ZigBee.

V. Measurements and Tests

This chapter describes how which measurements are done and which restriction we have applied. In the scenario one mobile robot shown above and three different radio technologies are tested. The server was a standard PC with 3.3 GHz and 2GB RAM. The measurements include the maximum communication data rate which could be achieved. Furthermore, the power consumption of the embedded board for the communication and processing of the commands is included. Also the time from the start of sending a message until the robots sends an acknowledgement is recorded and compared.

Because of the radio technologies we cannot grant a free medium for the communication. Based on that the test series is aborted when a single responds time value is higher than 200% of the average value. One test series includes 1500 values and the server and the robots are set with 1m space. Figure 7 shows the recorded data for the first test, with the comparison of the communication technology only.

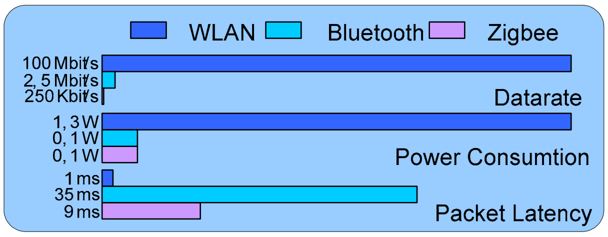
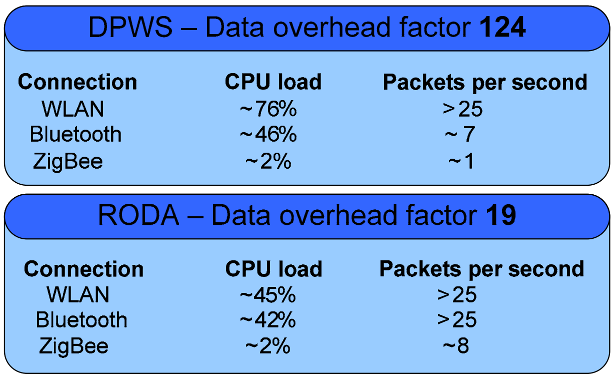


Figure 7: Overview of evaluated physical communication technologies

The measurement show that, also for this resource constrained devices, WLAN has the best power to data factor. But WLAN also has a high static power consumption which makes this technology only applicable when the bandwidth is needed. Also the latency for the communication scenario is only useful with WLAN all other technologies will be too restricted in most automation applications.

If a higher latency time possible in the automation scenario the comparison of RODA and DPWS is interesting. Figure 8 shows the factor for communication data and payload with both architectures, also the average CPU load of the embedded device during the test series, and the packets per second. As you vision system has a maximum rate of 25 frames per second this is the maximum possible value. The control loop does not need to be fast to control the robot in an accurate way.

This measurement shows that RODA has a signification lower data overhead factor. Therefore it does not need so much computing resources for the data conversation and communication. This allows also the accurate robot control over Bluetooth and WLAN. Furthermore, the implementation footprint for RODA is only 0.25 MB, DPWS need 2.2 MB for the footprint.



DPWS – Data overhead factor 124

Connection	CPU load	Packets per second
WLAN	~76%	>25
Bluetooth	~46%	~7
ZigBee	~2%	~1

RODA – Data overhead factor 19

Connection	CPU load	Packets per second
WLAN	~45%	>25
Bluetooth	~42%	>25
ZigBee	~2%	~8

Figure 8: Compared Data Overhead, CPU load, and Packets per Second in the Example Scenario

VI. Conclusion

RODA allows a lightweight device implementation and therefore a fast data processing. The specified protocols like Zeroconf are well defined, open source, and width speeded. This allows an easy implementation of the RODA architecture. Based on the assumption, that in the automation informatics the most important data streams are small size data packets that are transmitted very often, it should be noticed that RODA has a small data overhead factor than DPWS. Therefore, is saves important bandwidth and respond times. The open source community grants a further advancement of the used technologies and stacks.

Further development and enhancements of the RODA architecture can be easily applied, based on the modularized structure. For the automation informatics the next step should be the real-time capabilities, which can be achieved with the specification of HTTPU interfaces. These interfaces can be embedded in real-time networks that can grant real-time capabilities for UDP data communication.

Acknowledgment

This work has been achieved in the European ITEA project uSERVICE and has been funded by the German Federal Ministry of Education and Research under contract number 01|S0902F. Furthermore, thanks to Andreas Bobek and Thorsten Schulz for their previous work.

References

[1] OASIS Web Services Discovery and Web Services Devices Profile (WS-DD) TC, www.oasis-open.org/committees/ws-dd/, 2008.
[2] Microsoft, Intel, Ricoh, Lexmark. Device Profile for Web Services Specification. http://specs.xmlsoap.org/ws/2006/02/devprof, 2006.
[3] E. Zeeb, A. Bobek, H. Bohn, S. Prüter, A. Pohl, H. Krumm, I. Lück, F. Golatowski, and D. Timmermann. WS4D: SOAToolkits making embedded systems ready for Web Services. In 3rd International Conference on Open Source Systems, Limerick, Ireland, 2007.
[4] Microsoft Corporation. DCOM: Distributed Component Object Model, http://msdn.microsoft.com/de-de/library/ms809311(en-us).aspx, 1997.
[5] Microsoft Corporation. SAP DCOM Component Connector, http://msdn.microsoft.com/en-us/library/ms809349.aspx, 1997.
[6] UserLand Software. Extensible Markup Language Remote Procedure Call, http://www.xmlrpc.com, 1998.
[7] Object Management Group, Common Object Request Broker Architecture, http://www.corba.org/, CORBA V3.0 – 2002.
[8] UPnP-Forum, Universal Plug and Play, http://www.upnp.org/, 1999.
[9] World Wide Web Consortium. Web Service Architecture Specification. Technical report, World Wide Web Consortium, http://www.w3.org/TR/ws-arch/, 2007.
[10] WorldWideWeb Consortium. Simple Object Access Protocol Specification. Technical report, World Wide Web Consortium, http://www.w3.org/TR/soap/, 2007.
[11] Microsoft Corporation. DPWS Specification. Technical report, Microsoft Corporation, http://specs.xmlsoap.org/ws/2006/02/devprof, 2006.
[12] OPC Foundation. OPC Unified Architecture, http://www.opcfoundation.org/, 2006.
[13] Freedesktop. D-Bus, http://www.freedesktop.org/wiki/Software/dbus, 2002.
[14] Fielding, Roy T.: Architectural Styles and the Design of Network-based Software Architectures, University of California, Irvine, Department of Informatics, DISSERTATION, 2000.
[15] LiPPERT: PC solutions for rugged industrial applications, Cool Mote Master Board, Technical report, http://www.lippert-at.com/, 2009.
[16] WS4D: Web Services for Devices C Stack, University of Rostock, http://www.ws4d.org, 2009.
[17] G. Moritz, S. Prüter, F. Golatowski, and D. Timmermann: Real-Time Service-oriented Communication Protocols on Resource Constrained Devices: IMCSIT2008, Proceedings on, vol. 3, pp. 695 – 701, ISBN: 978-83-60810-14-9, October 2008.
[18] G. Moritz, S. Prüter, F. Golatowski, D. Timmermann: Web services on Deeply Embedded Devices with Real-Time Processing: ETFA 2008, ISBN: 978-1-4244-1505-2, September 2008.
[19] S. Prüter, G. Moritz, E. Zeeb, F. Golatowski, D. Timmermann, R. Salomon: Applicability of Web Service Technologies to Reach Real Time Capabilities: ISORC, ISBN: 978-0-7695-3132-8, May 2008.
[20] A. Bobek, Service-oriented Infrastructures for network Services and embedded Devices, University of Rostock, Dissertation, 2008.
[21] T. Schulz, Development of a sensor gateway for embedded devices, University of Rostock, Thesis, 2008.
[22] The Internet Engineering Task Force, Zero Configuration Networking (Zeroconf), http://www.zeroconf.org/, 2003
[23] Java.net Project, Web Application Description Language (WADL),: https://wadl.dev.java.net/, 2009