# SIRENA - Service Infrastructure for Real-time Embedded Networked Devices: A service oriented framework for different domains

Hendrik Bohn, Andreas Bobek, Frank Golatowski
University of Rostock
Institute for ME and CE
18051 Rostock, Germany
{hendrik.bohn, andreas.bobek, frank.golatowski}@uni-rostock.de

## Abstract

*The SIRENA project started in 2003 to leverage Service Oriented Architectures (SOA) to seamlessly interconnect (embedded) devices inside and between four distinct domains - the industrial, telecommunication, automotive and home automation domain. A framework was developed to achieve this aim as well as to assure interoperability with existing devices and extensibility of networks based on SIRENA technology. The core of the framework is the Devices Profile for Web Services (DPWS) which will play an important role in the upcoming Microsoft Windows Vista platform. The first DPWS stack worldwide for embedded devices developed by the SIRENA consortium is described and its operation evidenced in several demonstrators. This paper presents the results of the SIRENA project, provides insight into the technologies used and presents tools, components and services for advanced development, deployment and maintenance of devices.*

## 1. Introduction

Market research from IDC predicted that the annual sale of network-enabled (non-PC) devices will reach $44 billion in 2005 and Forrester Research forecasts a market expansion for network-connected devices to 14 billion units by 2010 [2]. These figures highlight the need for new approaches to ease the integration and interaction of devices in today's networks.

The *SIRENA* (Service Infrastructure for Real-time Embedded Networked Applications) project was started in the framework of the European premier cooperative R&D program "ITEA" in 2003 to define an innovative *Service Oriented Architecture* (SOA) framework to seamlessly connect heterogeneous (resource constrained) devices from the industrial, automotive, telecommunication and home automation domains [1]. 15 partners from Germany, France and Spain formed the consortium. Among them were companies like Siemens Business Services (D), Schneider Electric (F), EADS Defence and Security (F), Capgemini (F), Materna (D), Robotiker (E) and other small and medium size companies, research institutes (e.g. Fraunhofer FIRST) and several universities. It was required to support plug-and-play connectivity, to base on open standards, to apply down to sensor and actuator level and to be technology neutral regarding programming languages, operating systems and network media.

This paper describes the developed *SIRENA Framework* and shows its feasibility and advances in various demonstrators and tools. Section 2 presents an overview of evaluated technologies and points out the advances of the *Devices Profile for Web Services* (DPWS) chosen for the SIRENA base technology. In section 3 the SIRENA Framework is described in detail which builds the foundation for the demonstrators - section 4. Finally a conclusion is drawn and future work is discussed in section 5.

## 2. Evaluation of base technology

The *Service Oriented Architecture (SOA)* approach provides a standardized view to networked devices and applications. These entities are called *Services*. The functionality of a service is entirely and exclusively defined by the interface (or "contract") that the service exposes. The implementation is totally hidden. A service interface is described in a standardized format (*Service Description*), which is used to integrate, announce, find and use the functionality offered. Furthermore, services can subscribe to other services to get informed about state changes. Although SOAs should be platform and programming language independent in order to be widely applicable, they still rely on some standards (e.g. most SOAs built up on TCP/IP and more and more XML is used as a standard information format).

Several service-oriented middleware approaches have been identified and evaluated as potential SIRENA base technology: Open Service Gateway Initiative (OSGi), Home Audio/Video Interoperability (HAVi), Java Intelligent Network Infrastructure (JINI), Universal Plug and Play (UPnP), Web services and Devices Profile for Web Services (DPWS), respectively.

The *OSGi* specification defines a service platform that serves as a common architecture for service providers, service developers and software equipment vendors who want to deploy, develop, and manage services [13]. The specification is based on the Java platform and application independent thereby enabling the easy integration of existing technologies. Deployed services are called *bundles* and are plugged into the framework. An OSGi service is a simple Java interface but the semantics of a service are not clearly specified. The drawback for SIRENA is the reliance on Java.

*HAVi* was developed for and is still restricted to the home domain, in particular IEEE1394 networks [15]. It offers plug-and-play capability as well as Quality-of-Service (QoS) support (relying on IEEE1394 QoS capabilities). Recent research demonstrates that HAVi can be combined with other middleware such as UPnP and OSGi [7]. The SIRENA consortium rejected HAVi technology due to its restrictions and its rather low availability on the market.

*JINI* was developed by Sun Microsystems for spontaneous networking of services and resources based on the Java technology [14]. Services/devices are registered and maintained at a centralized meta service called *Lookup Service* but carry the code (proxy) needed to use them. This code is dynamically downloaded by clients when they wish to use the service. Each service access has to be performed by using the lookup service. JINI was not chosen by the SIRENA consortium due to its reliance on Java and the need of a centralized service registry.

*UPnP* is a simple, easy-to-use SOA for small networks [16]. It supports ad-hoc networking of devices and interaction of services by defining their announcement, discovery and usage. Programming languages and transmission media are not assumed. Only protocols and interfaces are specified instead. The UPnP specification divides the device interaction patterns into six phases: *Addressing*, *Discovery* and *Description* specify automatic integration of devices and services, *Control* (operating a remote service/device), *Eventing* (subscribing to state changes of a remote service/device) and *Presentation* (URL representation of a service/device) specify how to use them. UPnP was a choice for SIRENA basic technology but has the disadvantage of supporting only smaller networks. With an increasing amount of services/devices the amount of broadcast messages grows exponentially in a UPnP network. Furthermore UPnP supports IPv4 only.

The *Web service* architecture provides a set of modular protocol building blocks that can be composed in varying ways to create protocols specific to particular applications [18]. They address networks of any size and provide a set of specifications for service discovery, service description, security, policy and others. Web services are self-describing using *WSDL* (Web Services Definition Language). The implementation is entirely hidden from their interfaces and may be changed at run-time. Unfortunately, Web services do not bring Plug and Play capabilities and a sufficient solution (spec and guideline) for device integration. This is overcome by DPWS.

*DPWS*, first published in May 2004 and revised in October 2005, is a profile identifying a core set of Web services that enables dynamic discovery of, and event capabilities for Web services [12]. The profile arranges several Web service specifications such as *WS-Addressing*, *WS-Discovery*, *WS-MetadataExchange*, and *WS-Eventing* for devices, particularly. In contrast to UPnP, it supports discovery and interoperability of Web services beyond local networks. DPWS is intended to become the successor of UPnP. However, the replacement is hindered by the fact that DPWS is not compatible with UPnP. DPWS is therefore most easily adopted in environments with no ”legacy” UPnP devices, such as industrial automation.

| | OSGi | HAVi | JINI | UPnP | WS | DPWS |
|---|---|---|---|---|---|---|
| Plug and Play | - | X | X | X | - | X |
| Device support | X | X | X | X | - | X |
| Programming Lang independent | - | X | - | X | X | X |
| Network media independent | - | - | X | X | X | X |
| Large scalability | X | - | X | - | X | X |
| Security | X | X | X | - | X | X |
| High market acceptance | X | - | X | X | X | X |

**Figure 1. Excerpt from comparing evaluated technologies for device integration.**

Figure 1 shows an excerpt of the evaluation results. The requirements for the SIRENA project reduced the choice to either UPnP, Web services and DPWS. UPnP fulfills most requirements but does not have security aspects included and does not support larger networks. Web services only support software services. The SIRENA consortium decided for DPWS as the basic technology for the SIRENA Framework. Due to the late release of the DPWS proposal and the existence of ”legacy” UPnP devices some SIRENA developments focus on UPnP with the aim of replacing it later by DPWS.

## 3. SIRENA framework

The SIRENA Framework is displayed in figure 2. Although DPWS is designed in such a way that any kind of transport protocol could be used, IPv4 and IPv6 form the underlying protocols in the SIRENA project because of their wide acceptance. The SIRENA Basic Framework defines the basic service oriented technology used for device integration and interaction. All SIRENA enabled devices should comply with the Basic Framework. The SIRENA Framework Enhancements are a set of tools, components and services making the development, deployment, maintenance and lifecycle management of devices and services easier. Devices and services from other SOAs may be attached to the SIRENA Framework by using the SIRENA Framework Extension Interface.

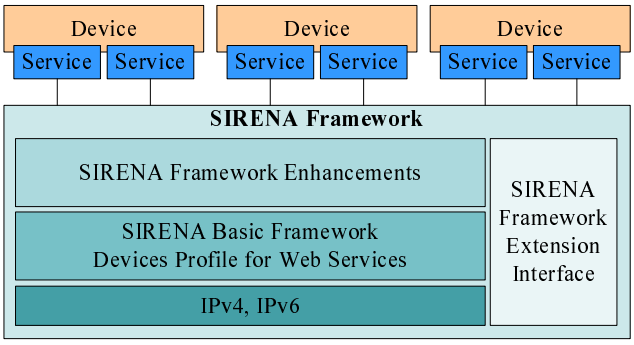


**Figure 2. The SIRENA Framework.**

### 3.1 SIRENA Basic Framework

The SIRENA Basic Framework defines the basic architecture to seamlessly connect heterogeneous devices and services offered by such devices. At the time when the SIRENA consortium was working on a specification, the DPWS proposal became available. DPWS fulfilled most of the SIRENA requirements and a high market acceptance was anticipated due to the alignment with Web services.

**DPWS:** Web services specifications are a set of protocols that can be used independently or that can be reused by combining or profiling them to form a new specification to meet different services requirements. DPWS is such a profile combining the following protocols - shown in figure 3.

*WS-Addressing* introduces two new concepts to the Web services world: endpoint references and message information headers. An endpoint reference is a transport-neutral mechanism to address Web services, e.g. instead of widely used HTTP-depending URLs the more general Uniform Resource Identifier (URI) can be used which possibly can not

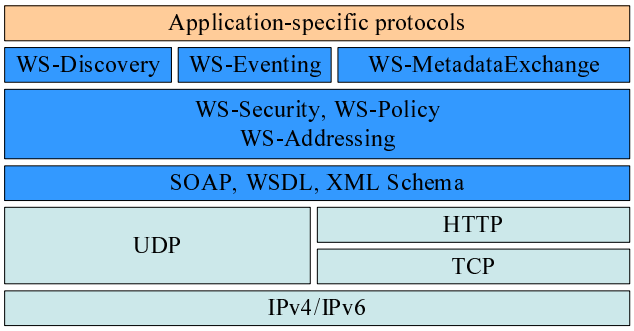


**Figure 3. The Devices Profile for Web services.**

be resolved into an accessible physical address. Moreover, message information headers enrich messages with meta-information. Messages can be marked with sender, response and error receiver endpoint references, and with message IDs that uniquely identify messages. The combination of these concepts enables Web services to communicate asynchronously: they no longer need to rely on synchronous HTTP message exchange only.

*WS-Discovery* allows dynamic detection of Web services by defining a specific multicast group. For that purpose it introduces three different endpoint types: target service, client and discovery proxy. The target service is a service container which offers services by announcing them to the network. Clients can search for target services and discover them. Discovery proxy is an optional component. If available, all discovery messages are exchanged with the proxy via unicast. Furthermore WS-Discovery defines how transport-neutral addresses of endpoint references can be resolved to transport-specific addresses.

*WS-MetadataExchange* defines two operations to obtain metadata from endpoint references. Metadata is static data (e.g. WSDL, XML schema data) which describe endpoint references in such a way that they facilitate communication with endpoint references.

*WS-Eventing* allows a Web service (subscriber) to subscribe to another Web service (event source) to obtain event notification messages when state changes occur.

The devices profile puts these protocols together and adds some device-oriented constraints and usage rules, e.g. it defines a dialect called "ThisModel" for describing device characteristics accessed using the WS-MetadataExchange specification.

**SIRENA-DPWS-Stack:** The SIRENA-DPWS-Stack was developed by Schneider Electric and is the first DPWS implementation worldwide [10]. It is based on the open source package gSOAP [17] and is implemented in C. Main features of the stack are service invocation using the SOAP 1.2 engine and WS-Addressing specification, au-

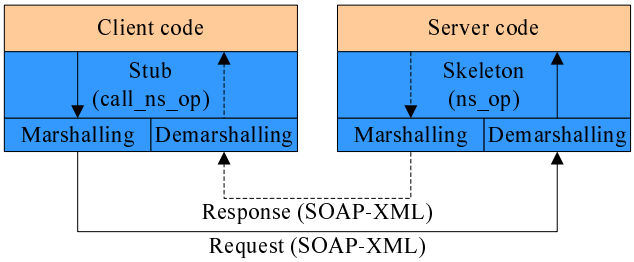


**Figure 4. Remote operation invocation in gSOAP/DPWS.**

tomated discovery via WS-Discovery, partial support of WS-MetadataExchange and event management using WS-Eventing. The stack includes a toolkit which follows the same approach as gSOAP - the automated code generation for mapping SOAP-XML-messages and C/C++ structures back and forth. In other words, it is responsible for generating the *stub* (DPWS client interface) and *skeleton* (DPWS server interface) and the marshalling and demarshalling of messages.

Figure 4 illustrates the operation of the DPWS Stack. Only the server and client code must be implemented by a service developer. The SIRENA-DPWS-Stack is not a stand-alone implementation. It is statically bound to the client and server program, respectively. Thereby it is ideal for small embedded devices, as execution times are optimized. The stack was ported to ThreadX, Linux, Solaris, Windows CE and XP, VxWorks and Quadros (Schneider Electric Ethernet FIRE brick) and has less than 200 kBytes of footprint. Its development will be continued and commercial support is provided by Schneider Electric.

The University of Rostock currently develops a Java DPWS protocol stack which is based on the upcoming Apache Axis Architecture 2.0 [3]. It will be finished in 2006.

**UPnP support:** Devices relying on Java are attached to the SIRENA network by using UPnP (until a Java DPWS stack is available). Those devices are DPWS-ready. They have an abstraction layer integrated allowing simple replacement of Java UPnP by Java DPWS.

### 3.2 SIRENA Framework Enhancements

The SIRENA Framework Enhancements have been developed to ease the development, integration, deployment, maintenance and (lifecycle) management of devices and services in a SIRENA based network (figure 6). It is a set of tools, components and services, described in more detail below.

**DPWS-Stub-Skeleton-Generator:** The SIRENA-DPWS-Stack includes a toolkit for automated generation of stub (proxy) and skeleton for service developers to provide transparent access to the remote operations from a client. The developer specifies the WSDL or SOAP description of the service being developed as shown in figure 5. It involves operations and data types used as parameters and return values to those operations. Only the implementations of the server operations and client remote operation calls are left to the developer.

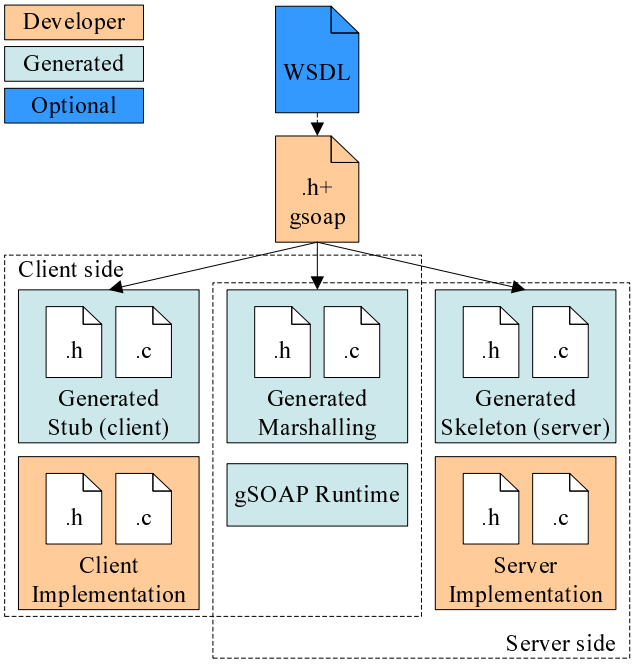


**Figure 5. Design flow for service development using DPWS.**

**WebWSDL:** The WebWSDL tool was developed by the University of Rostock and is a Web based tool to simplify the definition of a WSDL for a device using a Web front end. WSDL data types are stored in a database and can be simply selected. When the Web editing process is finished the WSDL file is generated and passed to the SIRENA-DPWS-Stack. All files generated by the stack can be downloaded by the developer. WebWSDL allows collaborative development.

**Device Lifecycle Management:** The Device Life-Cycle Management (DLCM) Service has been developed by Fraunhofer FIRST. It can be used to control the life-cycle of local device software such as installing, starting, stopping, updating and removing devices in a UPnP network. The DLCM service is based on the OSGi Service Platform (open-source implementation "Oscar") and therefore running as a UPnP service in a Java run-time environment (using the Siemens UPnP stack). The DLCM is DPWS-ready.

**MoBaSeC:** The Model Based Service Configuration

tool enables developers to visually model the design of configurations for management services and generates those management services automatically [9]. The model is policy based. MoBaSeC results from a joint effort of University of Dortmund and Materna (D). A model of the service-oriented system is created visually and incorporates all devices and services including their relevant parameters. Created management services supervise running services and their environments, can react on changes and reconfigure managed services at runtime. Using management services metatasks can be performed by combining heterogenous services - *Service Orchestration*. This unfolds the whole potential of SOAs. A service-oriented abstraction layer allows the support of DPWS and UPnP in MoBaSeC.

**GINGER:** GINGER is a process management system developed in Java by kachel GmbH (D) for the integration of applications and systems based on configurable workflow definitions [5]. It provides an open and flexible plug-in mechanism for accessing systems and/or applications, where plug-ins are used to couple an activity to a system and/or application. For configuration and administration of workflow-based applications GINGER provides tools to support definition-time as well as run-time. The University of Paderborn and Siemens Business Services implemented an abstraction layer making GINGER UPnP and DPWS enabled. A slideshow as a workflow scenario using UPnP was presented at the CeBIT'05 computer fair in Hanover (D).

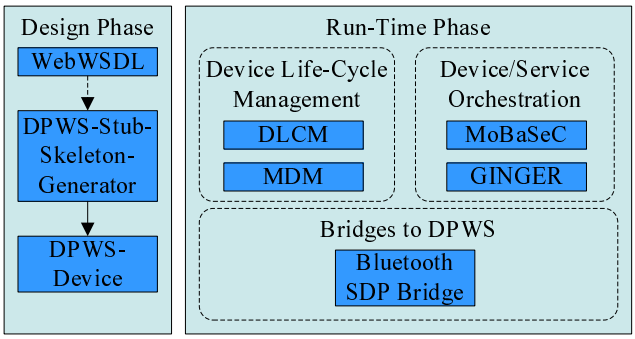


**Figure 6. Overview of DPWS tools in the SIRENA Framework.**

**Device metadata management:** The Manage Device Metadata (MDM) service was developed by Capgemini in cooperation with Schneider Electric. This service extends DPWS with management capabilities by requesting and changing metadata information of remote devices. Metadata is used in DPWS to describe device characteristics and discovery information (e.g. device type). The MDM service can manage the device's life-cycle by adjusting metadata information of remote services.

**UPnP AV Stack:** The UPnP Audio/Video Java stack of the University of Rostock is a software component that can be used to develop AV applications (*MediaServer*, *MediaRenderer* or *AV Control Points*) [4]. The AV interfaces are the core of this component. They provide a layer to access AV devices, AV services and their actions. AV interfaces hide the complexity of *UPnP AV Architecture* and peculiarities of specific underlying UPnP stacks. Vendor AV stacks implement AV interfaces. Vendor AV stacks are accessible by specific factory methods in a vendor factory class only (a Factory Method is a well known design pattern for creating an instance of an unknown class that implements a specific interface) [8]. AV applications use the factory methods and AV interfaces only. Therefore, applications written to this paradigm are independent of a specific vendor UPnP stack. Currently, we provide AV stack prototype implementations for the Siemens UPnP stack and the Cyberlink UPnP stack.

### 3.3 SIRENA Framework Extension Interface

The SIRENA Framework Extension Interface describes the requirements for integrating non-SIRENA devices into the SIRENA Framework. The SIRENA consortium has written tutorials for a SIRENA-OSGi bundle, an abstraction layer to replace a Java UPnP stack by DPWS and a DPWS-Bluetooth-Bridge.

**DPWS-Bluetooth SDP bridge:** The DPWS-Bluetooth SDP bridge was developed by the University of Rostock to connect resource constrained devices to a SIRENA network [6]. Bluetooth Service Discovery Protocol (SDP) is a service-oriented approach for Bluetooth devices including description and discovery. SIRENA Bluetooth devices can be discovered by their specific SIRENA description, which is mapped to a corresponding DPWS metadata description. Therefore a central device - the *Bluetooth Device Manager* (BDM) - is required in a Bluetooth network in order to manage connected Bluetooth devices and to map their description. The BDM is a DPWS device in the SIRENA network and all services from the Bluetooth devices are offered as services of the BDM.

## 4. SIRENA demonstrators

Several demonstrators were developed to show the integration and interaction of described devices, components, stacks and tools. One demonstrator was designed for each domain and one that presented domain crossing applications.

**Industrial demonstrator:** The industrial demonstrator is a fully functional model of a production chain component - a dose maker [11]. Its purpose is to fill granules from a tank into small bottles. It includes a motor to move the granules from the tank via a trap to the bottles. Various

sensors observe the state of the dose maker. The dose maker components are SIRENA-DPWS devices and entirely communicate via DPWS.

**Telecommunication demonstrator:** The telecommunication demonstrator presents SIRENA-DPWS based management of cellular network equipment. It enables dynamic configuration of network devices and cells, reaction on failures in networking equipment as well as supervision of end-user services (e.g. Short Message Service). Plug-and-play capabilities of attached devices provide a significant advantage in this context.

**Automotive demonstrator:** A central device in the car (UPnP and DPWS enabled) forms the core of the automotive demonstrator. MoBaSeC manages the integration of Bluetooth-enabled temperature sensors. A real-time device offers information from the *Controller Area Network* (CAN) bus for emission-reduced driving and a *Digital Audio Broadcast* (DAB) device plays back audio media on the central device using UPnP.

**Home demonstrator:** The home demonstrator realizes media distribution, streaming and replication as a testbed for network-connected devices. It uses UPnP for legacy devices and DPWS for new devices. The legacy devices are made DPWS-ready.

**Cross-domain demonstrator:** Cross domain capabilities of SIRENA devices were demonstrated by combining usage scenarios from the home domain in the automotive domain such as the UPnP AV scenario - playing music from a notebook on the central device in the car.

## 5. Conclusion

The presented paper has pointed out the need for new technologies for device integration in heterogeneous domains, which was the motivation for the SIRENA project. DPWS was selected as the best choice to achieve this aim and the first stack worldwide for embedded devices was developed. Several tools, services and components for DPWS ease the development, deployment and integration of devices and services, and support legacy devices from other service-oriented technologies. The demonstrators show the feasibility of the SIRENA approach.

In April 2005 the SIRENA DPWS stack was successfully presented to Microsoft. It also successfully participated in the first DPWS interoperability workshop held in California in September 2005. Microsoft is integrating DPWS technology into their new Windows Vista platform. In November 2005 the SIRENA project was successfully evaluated by ITEA, German and French public authorities.

The SIRENA technology is developed further - far in excess of the SIRENA project which will be finished in March 2006. A Java stack will be available in 2006 and real-time functionality will be provided by a real-time IP stack that manages *Quality of Service* (QoS) aspects in the SIRENA Basic Framework. Schneider Electric is currently developing such a stack and QoS framework enhancement, as well as bringing the SIRENA technology to the market. Fraunhofer FIRST plans to develop a DPWS-OSGi bundle to attach DPWS devices to the OSGi framework.

The ITEA project *Service Oriented Device Architectures* (SODA) will continue the research and development around device integration.

## References


[1] SIRENA. `http://www.sirena-itea.org`, year = 2005.

[2] Forrester Research: The Rebirth Of European Telecoms. `http://www.forrester.com`, 2001.

[3] Apache Software Foundation. *Axis Architecture Guide 2.0*, 2005.

[4] A. Bobek, H. Bohn, and F. Golatowski. UPnP AV Architecture: Generic Interface Design and Java Implementation. In *PDCN'05*, Innsbruck, Austria, 2005.

[5] A. Bobek, H. Bohn, F. Golatowski, G. Kachel, and A. Spreen. Enabling Workflow in UPnP Networks. In *INDIN'05*, Perth, Australia, 2005.

[6] H. Bohn, A. Bobek, and F. Golatowski. Bluetooth Device Manager Connecting a Large Number of Resource-Constraint Devices in a Service-Oriented Bluetooth Network. In *ICN'05*, St. Gilles Les Bains, La Reunion, 2005.

[7] M. Ditze, G. Kaemper, I. Jahnich, and R. Bernhardi-Grisson. Service-based Access to Distributed Embedded Devices through the Open Service Gateway. In *INDIN'04*, Berlin, Germany, 2004.

[8] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley, 1995.

[9] S. Illner, H. Krumm, I. Lueck, A. Pohl, A. Bobek, H. Bohn, and F. Golatowski. Management of Embedded Service Systems - An Applied Approach. Submitted to AINA 2006, 2005.

[10] F. Jammes and H. Smit. Service-Oriented Architectures for Devices - the SIRENA View. In *INDIN'05*, Perth, Australia, 2005.

[11] F. Jammes, H. Smit, J. L. M. Lastra, and I. M. Delamer. Orchestration of Service-Oriented Manufacturing Processes. In *ETFA'05*, Catania, Italy, 2005.

[12] Microsoft. *Devices Profile for Web Services*, 2005.

[13] OSGi Alliance. *OSGi Service Platform Release 4 CORE*, 2005.

[14] Sun Microsystems. *Jini Architecture Specification Version 1.2*, 2001.

[15] J. Teirikangas. HAVi: Home Audio Video Interoperability. Technical report, Helsinki University of Technology, 2001.

[16] UPnP Forum. *UPnP Device Architecture v.1.0.1*, 2003.

[17] R. A. van Engelen and K. A. Gallivan. The gSOAP Toolkit for Web Services and Peer-To-Peer Computing Networks. In *CCGrid'02*, Berlin, Germany, 2002.

[18] W3C. *Web Services Architecture*, 2004.