A Concept for a Medical Device Plug-and-Play Architecture based on Web Services

Stephan Pöhlsen
University of Lübeck
Institute of Telematics
Ratzeburger Allee 160
23538 Lübeck, Germany
poehlsen@itm.uni-luebeck.de

Stefan Schlichting
Drägerwerk AG & Co. KGaA
Research Unit
Moislinger Allee 53–55
23542 Lübeck, Germany
stefan.schlichting@draeger.com

Markus Strähle
Dräger Medical AG & Co. KG
Moislinger Allee 53–55
23542 Lübeck, Germany
markus.straehle@draeger.com

Frank Franz
Drägerwerk AG & Co. KGaA
Research Unit
Moislinger Allee 53–55
23542 Lübeck, Germany
frank.franz@draeger.com

Christian Werner
University of Lübeck
Institute of Telematics
Ratzeburger Allee 160
23538 Lübeck, Germany
werner@itm.uni-luebeck.de

ABSTRACT

Medical device interoperability is still an issue. Standards exist only for specific areas like HL7 and DICOM, or have not been widely adopted like ISO/IEEE 11073 except for the domain information model at the semantic level. An approach that covers interoperability below the semantics is proposed. It is based on Web services which are widely accepted outside the medical device application domain. In particular the architecture is build on the upcoming Device Profile for Web Services (DPWS). It is a collection of existing Web services specifications for service discovery, interface description, event notification, and security. It is designed for resource-constrained devices and thus seems to be suitable as a basis for medical device plug-and-play.

1. INTRODUCTION

Interoperability is defined as "ability of two or more systems or components to exchange information and to use the information that has been exchanged" [6]. According to Lesh et al. [8] the interoperability continuum distends from the least complex endpoint of physical interoperability to the most complex of data interoperability. To achieve data interoperability not only the capability to exchange information without an error has to be given, but also correct interpretation of the information to use it in an algorithm. Benefits of medical device interoperability range from less development time for data-driven clinical decision support algorithms or medical device safety interlocks to improved patient safety. Many initiatives like the MD PnP Interoperability program [10] or IHE PCD [7] also address this problem.

The ISO/IEEE 11073 is a mature standard addressing the interoperability issue. Especially the provided domain information model is used frequently and enables medical devices to exchange data on the semantic level. The lower layers are complex and do not support current technologies like Ethernet or TCP/IP.

MediCAN is a solution proposed by McKneely et al. [9] and is described as a vendor-independent network architecture for interfacing medical devices based on proprietary protocols that are built on top of UDP/TCP as well as CAN-Bus. There is no direct communication between the devices and the access to the closed network of medical devices is controlled by a proxy server.

Another approach establishes interoperability between components of imaging systems built from OEM-components using special purpose CANopen device profiles. There is also a CANopen based application profile addressing acute care systems [25].

1.1 Service-oriented Architecture

The idea of service-oriented architectures (SOA) is coming from the business environment. The operation and maintenance of computer systems there is complex and thus very costly. Each time business processes change, new computer systems have to be integrated in the existing infrastructure. Often, these new computer systems have new interfaces and an integration with old systems is problematic.

A SOA solves this problem by introducing a new standardized interface technology for all systems, called "service". All

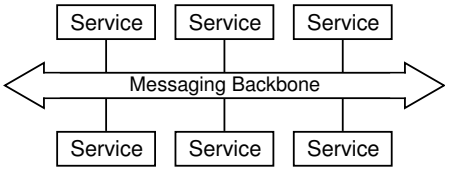

Figure 1: In a SOA all services communicate with each other over a common messaging backbone. The realization of this abstract backbone depends on the concrete SOA implementation. It can be a network or a central communication server for example.

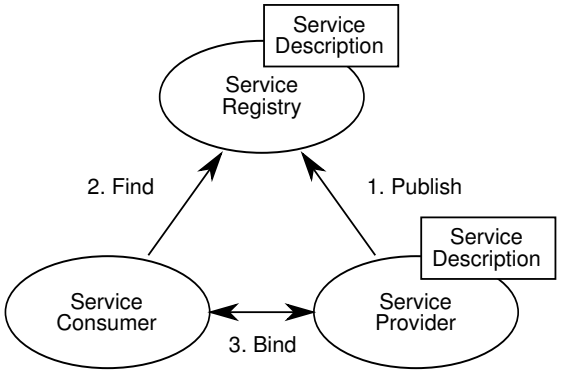

Figure 2: The three primary roles in a service-oriented architecture with the interaction among each other

services are now communicating with their service interface in a common messaging backbone according to Figure 1.

A SOA does not dictate a specific technology, it is just a concept how to build such an architecture. Figure 2 depicts that concept with the three primary roles: service provider, service registry, and service consumer. A service provider publishes its service description to a service registry. Then, a service consumer is able to find a corresponding service in the registry. Afterwards, the service consumer binds to the service provider to use that service.

Web services are a realization of a SOA with Internet technologies [15]. The benefit of Web services in contrast to other SOA solutions is that it uses well-known technologies and a vendor lock-in can be avoided.

The fundamental and widely accepted standards on which Web services are build are *SOAP* [23] and the *Web Services Description Language (WSDL)* [22]. WSDL covers the service description in the role model, while SOAP is an XML-based messaging transfer format. Together WSDL and SOAP cover the roles of the service provider and the service requester.

The registry role is not covered by these two standards, but by the *Universal Description Discovery & Integration (UDDI)* [14] that specifies an interface for such a registry. However, the standard is not so widely accepted as WSDL and SOAP. This is owed to the fact that the main idea behind UDDI to build one global service registry failed.

1.2 First Architecture Concepts

The architecture developed in the FUSION project, funded by the German Federal Ministry of Education and Research, is built upon Web services to realize a SOA. Two approaches have been explored: a complete centralized and a more decentralized approach. The latter to overcome the problem of a single point of failure and due to performance issues.

The centralized approach utilizes an enterprise service bus (ESB) to provide a common data-centric middleware structure for reliable transport and traceability of synchronous and asynchronous messages [19]. The distributed approach does not depend on a central structure, but more on standards from the Web service domain. Lookups of services are handled by a UDDI server. Web services of a provider are invoked directly by a consumer without sending the request/response through the ESB. Furthermore, events are transported using the WS-Eventing specification: either point-to-point or via an event broker. In most cases, point-to-point event notification will be used, if the event is safety critical and could not be send via a centralized event broker. This approach yields more flexibility, less configuration effort, and – by far the most important – communication between devices will not break down, even if one of the basic service components crashes. With regard to semantic interoperability, both approaches employ an XML vocabulary based on the ISO/IEEE 11073 domain information model for hemodynamic and respiratory data as well as HL7 for ADT data.

The problem with both solutions – even if reduced for the latter one – is the usage of centralized structures and hence the introduction of a single point of failure as well as scalability issues. Furthermore some effort has to be expended for configuration of the medical devices with the result that no plug-and-play feeling comes up.

For that reason, an architecture based upon current Web service standards is presented that is locally de-centralized and only needs central components if communication over subnet boundaries is needed. This concept is described in the following section.

2. PROPOSED ARCHITECTURE

The architecture proposal for medical device connectivity leverages the upcoming *Devices Profile for Web Services (DPWS)* standard [11] that defines services for discovery, interface description, messaging, event propagation as well as secure information transmission. It is out of scope of the proposed architecture to define the semantic side of interoperability. This can be handled using data models from mature standards like HL7 or ISO/IEEE 11073.

DPWS – becoming an OASIS standard in June 2009 – is a minimal set of Web services specifications for resource-constrained devices. It includes discovery and description of Web services as well as the possibility for event propagation. The origins of DPWS are in the consumer electronics domain where it is used in modern network printers or image scanners to allow plug-and-play. To sum up, DPWS achieves the same ease of use for consumer electronics for Ethernet as USB does for serial connected devices. It is pushed forward by Microsoft for future printer integration and consequently

Microsoft Windows Vista, Windows Server 2008, and Windows Embedded CE 6.0 R2 already have a native DPWS stack (called WSDAPI) on-board.

The services of the proposed architecture are discussed in detail below.

2.1 Dynamic discovery

Web Services Dynamic Discovery (WS-Discovery) [13] is a service localization protocol and will be standardized in the context of the DPWS standardization process. By default it operates in an ad-hoc mode without any configuration. Therefor it uses a predefined multicast group to reach all services within the same sub-network. Multicast is a transmission mode where the underlying network distributes the data to all subscribed nodes. The discovery functionality is similar to the probably known discovery procedure in SoHo (Small Office Home Office) firewall appliances or in media center systems that distribute music and video in home networks. In the managed mode the discovery process uses a centralized component called discovery proxy that caches all information. With this proxy the discovery process scales to a larger number of endpoints since the usage of multicast is reduced to a minimum in contrast to the ad-hoc mode.

The decentralized nature of the ad-hoc mode is applicable to the application domain of medical devices, since it avoids a single point of failure. It is important to have a robust discovery system in case of network failures outside the currently interacting devices. Medical devices in an operation room for example should function well in case of network failures outside their room. This scenario forbids a sole centralized registry for discovery.

In WS-Discovery service providers announce themselves when they join the network. It avoids the need for polling in service consumers for the same service periodically. With this feature it is possible to automatically update lists of available services or directly respond in case a new service appears in the network.

In Figure 3 two discovery procedures are shown. First, the service consumer on the left side is already running while the service provider starts up. The service consumer receives the hello announcement of that service provider and thus is able to directly start the interaction without the need for periodically polling for new services. In the second discovery sequence with the service consumer on the right, the service provider started before the consumer. Thus, the consumer missed the hello announcement and must search for that service.

WS-Discovery is designed for use within a single subnet. The multicast announcements and search requests do not scale very well, because all messages have to be sent to all WS-Discovery nodes in the network. Also the specification restricts the usage to a single subnet. Thus, this specification seems to be impractical for use in enterprise or hospital networks. In such large networks it is much more efficient to have a centralized component for discovering services.

On the one hand the reliability of the decentralized multicast discovery is required on the other hand an enterprise scale discovery is required. An approach to solve this conflict is to use both variants in combination with each other. For the managed mode WS-Discovery specifies a so-called discovery proxy. Originally it is used to cache all available information from the local service providers and response to search requests from service consumers within the same subnet. Instead of using this discovery proxy to reduce multicast traffic in subnets it can be used as a central component to search for services hospital wide – outside the current subnet. A benefit of using a WS-Discovery proxy server instead of a UDDI server is that the discovery process is more consistent for the architecture.

The proposed feasible 2-layer discovery architecture [17] is shown in Figure 4. The WS-Discovery compliant decentralized multicast discovery within a subnet is as robust as possible against network failures. It is sufficient, that a network connectivity between service provider and consumer exist. Practically this is no restriction, since the two services cannot communicate further. To discover services in the whole network it is necessary to query a centralized discovery proxy.

Different approaches are possible to reach the centralized discovery proxy. In [17] the DHCP protocol is used to transfer the IP address of the discovery proxy server to the service consumers. DHCP offers the possibility to add vendor specific data on the server side. Then DHCP clients are able to request this data. Another approach [16] is to use the domain name system. Here, the clients query the name servers for the IP address of the discovery proxy. The DNS based solution has the benefit of being easier to implement platform independent and has to be configured only in one name server instead of all subnet limited DHCP servers.

2.2 Event-driven Architecture Concepts

Web services started with typical request-response scenarios, where a client requests a service from the Web service provider and gets a response. A typical example for this are use cases from travel agencies where a flight and a hotel must be booked for a customer. Events are disregarded in this scenario.

In the medical application domain a lot of communication is event driven. For example, alarms were sent and real-time data is transmitted. That means that publish-subscribe has to be supported in addition to the typical request-response pattern for device control.

In IP-based networks two different solutions for publish-subscribe exists in general. The first one uses a dedicated point-to-point connection for each subscriber; while the second solution utilizes the functionality from the underlying network using UDP multicast.

For the first solution with n point-to-point connections for n subscribers different Web services specifications exist that were pushed by different companies: WS-Eventing (Microsoft) [1], WS-Notification (IBM) [5], and WS-Events (HP) [2]. In March 2006 they released a joint white paper to harmonize the existing specifications [3]. For event propagation it results in a new specification called WS-EventNotification that builds on WS-Eventing and has to

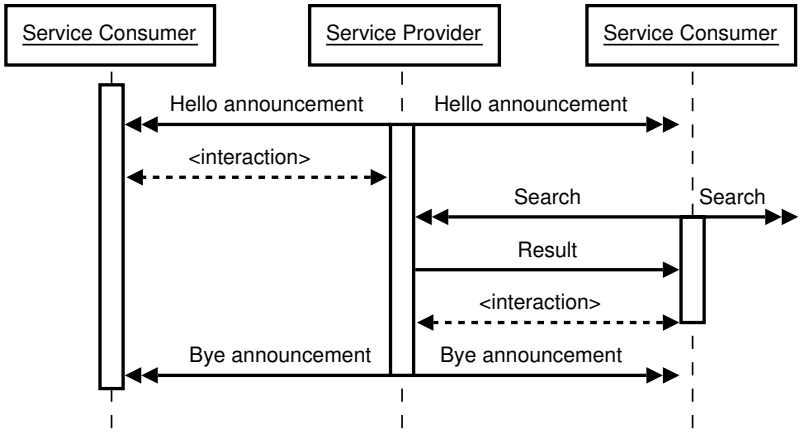

Figure 3: WS-Discovery sequence diagram for two discovery procedures: The service consumer on the right searches for services, while the consumer on the left waits for announcements.

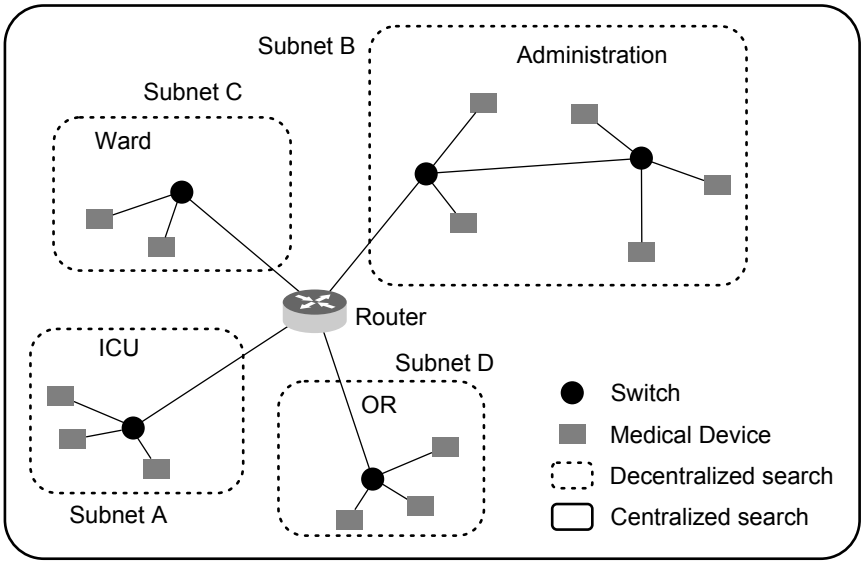

Figure 4: Two different discovery layers: Within the same subnet services can be found decentralized, while in the whole network a centralized scheme is necessary to go across subnet boundaries.

be specified in the future. It seems to be best practices to use WS-Eventing. It will also be referenced in the DPWS standard. In WS-Eventing the event source can delegate the management of subscriptions to and distribution of events to another Web service. This is practical for scenarios where the event source cannot or should not handle the list with all subscriptions.

The second solution is based on the multicast functionality from the underlying network. In this case, messages are sent via UDP multicast, which results in only one message transmission from the publisher to the multicast address. The distribution to the recipients is managed by the network routers. Multicast is designed from the group up for such distribution scenarios that is why it scales better than point-to-point transmission. The required specification to employ multicast messaging in Web services is SOAP-over-UDP [12] which is also included in the OASIS standardization process of DPWS.

2.3 Security

For sensitive data and control commands information security is important. In the context of medical device connectivity data integrity, confidentiality, availability, and non-repudiation is interesting for risk management.

Data integrity refers to the validity of data. The integrity can be harmed by accident or maliciously. Confidentiality means that the data is only accessible to those that are authorized to have access. Availability stands for a system that has all information available that are needed to serve its purpose. In the considered context of medical device interoperability, it is important to have a correctly functioning communication channel. Non-repudiation is required to ensure that a party in a dispute cannot repudiate a message that it sent out. This might be interesting for control commands.

Information security can be achieved on transport layer or at

message level. For transport layer security a secure channel is established for end-to-end communication. This secure channel provides data integrity and confidentiality as long as both endpoints authorize each other at the beginning. HTTPS is an example of transport layer security that uses the TLS (Transport Layer Security) protocol which is the successor of SSL (Secure Socket Layer) [4].

For message level security each message is secured individually instead of sending unsecured messages through a secure channel. The benefit of message level security is that the message can be passed through insecure nodes. Furthermore, non-repudiation is achieved on the message level utilizing digital signatures. In this case the signed messages must be logged for later proof of message transmission However, the message level security is more complex in contrast to the transport layer security.

For transport as well as message security a public key infrastructure (PKI) is a basic requirement for medical device authentication. A centralized approach with user name and password is inapplicably. It contradicts to the plug-and-play nature, introduces a single point of failure, and nevertheless requires a certificate to authenticate the server to the clients.

Availability for Web services highly depends on the underlying network. Ethernet is a best effort network that supports prioritization. Anyhow, a network node with unsocial behavior might disturb other data transmission and important data packets might get dropped in overloaded switches. For hard real-time constraints proprietary Ethernet modifications exists. These are not compatible with Web services. A tradeoff between normal Ethernet and proprietary solutions can be a fair network switch [18] that equally shares the available data rate between sending nodes. Thus the effect of unsocial nodes can be reduced and a minimal data rate can be guaranteed.

3. FIRST FEASIBILITY EVALUATIONS

First feasibility evaluations have been made with demonstrator programs on standard computer hardware. For WS-Eventing a few performance measurements were made to gain experience for multiple point-to-point transmissions. They were done with the DPWS toolkits WS4D-gSOAP and WS4D-JavaME [24] with enabled debugging/logging. The results are summarized in Table 1.

The performance measurements showed for a WS4D-gSOAP device, that for one WS4D-gSOAP listener it takes 3.0 ms to transmit the event and receive an acknowledgement. For two listeners the event is delivered after 5.7 ms. This time does not only depend on the devices implementation it also depends on the listeners. For one Java listener implemented with the WS4D-JavaME toolkit it took 3.8 ms instead of the 3.0 ms with the WS4D-gSOAP listener.

Table 1: Exemplary performance results for event distribution according to WS-Eventing for different toolkits

Event source	Listener	Total delay
WS4D-gSOAP	1 × WS4D-gSOAP	3.0 ms
WS4D-gSOAP	2 × WS4D-gSOAP	5.7 ms
WS4D-gSOAP	1 × WS4D-JavaME	3.8 ms

Table 2: Serialization overhead in WS4D-gSOAP for normal messages and messages with a digital signature for different payloads on PC hardware

Payload	Normal messages	Signature w/o certificate	Signature with certificate
1 Integer	1.3 ms	6 ms	8 ms
50 Integer	2.4 ms	12 ms	14 ms

For measurement of the overhead especially for message level security the WS4D-gSOAP toolkit was used. It is an add-on for the gSOAP toolkit [20] that has support for digital signatures. Table 2 summarizes first results for the serialization time in different scenarios with debug logging enabled by default. From the second column it can be concluded that the overhead for the SOAP envelope and the DPWS compliant SOAP header takes 1.3 ms. The serialization of an additional integer takes merely 22 μs. In the third column the whole body of a SOAP message is signed according to WS-Security with a 1024 bit RSA key and the SHA1 hash algorithm. The certificate for the RSA key is not included because it is already known to recipient in this scenario. An additional integer in a signed SOAP message takes 122 μs which is 5 to 6 times the overhead according to the unsigned message. This overhead solely results from the SHA1 hashing in the signature procedure. From the last column it can be concluded that an embedded certificate leads to an overhead of 2 ms for serialization and hashing.

Van Engelen et al. [21] did further performance measurements concerning security overhead. They compared HTTPS transmissions with symmetric and asymmetric message level security.

4. RESULTS

A concept for a plug-and-play architecture was introduced. It is build on current Web services standards and covers the technical layers of interoperability. It is an infrastructure architecture and thus semantics is out of scope and existing standards should be used. To discover other devices an enhanced version of WS-Discovery is used to offer the possibility to reach services outside the current sub-network. For event propagation two different solutions WS-Eventing and SOAP-over-UDP multicast coexist with different advantages. Information security is covered at the transport layer as well as at the message level. First performance tests showed a timing overhead that results from the use of Web services. If security is needed due to safety reasons, this adds of course an additional overhead that cannot be avoided.

5. DISCUSSION

In the medical application domain the proposed DPWS-based architecture seems to provide a basis for device connectivity. It is designed to achieve full plug-and-play capabilities with devices from different vendors. It runs on resource-constrained devices as well as on high-end computer systems. Web services in general are one of the widest

accepted building blocks for a SOA that probably has the most tool support. The reasons for this are supposably its platform and programming language independency in combination with the use of well-known Internet technologies such as HTTP.

Even, if there some reservations concerning the performance of Web services for medical device connectivity. The performed measurements show that they could be used for communication even if low transmission latency is needed.

For event and real-time data propagation SOAP-over-UDP multicast should scale better since multicast was designed for such purposes in contrast to WS-Eventing that tried to add the required functionality on top of existing Web services standards. The exemplary measurements for WS-Eventing support that assumption. The delay depends on the listener implementation and the scheduling strategy of the sender. From a devices perspective the listener implementation can not be influenced. The sender is able to inform the listener in sequential order or concurrently. When transmitting the messages in sequential order the last subscribed listener will be informed with a large delay. In contrast to that a programmatically concurrent transmission is more resource expensive, since multiple threads are required.

Also the UDP multicast has a drawback. It does not support a reliable transport. It is assumed that all transmitted data is received by the listener. Thus, the decision between WS-Eventing and SOAP-over-UDP multicast is not a Web service specific decision.

Anyhow, the major problem in both cases is: What to do with failed transmissions? Queue them and try it x minutes later or just fire and forget an event? All these different aspects lead to the conclusion that it depends on the specific event (i.e., alarm or real-time data) which solution might be more practical.

In case of information security this decision is even more complex. Van Engelen et al. [21] propose to use HTTPS whenever it is possible since it performs much better then message level security. It is only possible to use HTTPS in conjunction with WS-Eventing, because of the nature of multicast transmissions that require message level security.

A de-centralized plug-and-play architecture like the proposed one might have the best chances to smoothly migrate into the existing infrastructure since it does not require complex components – two modern devices are sufficient.

6. REFERENCES

[1] D. Box, et al. Web Services Eventing (WS-Eventing), Mar 2006.

[2] N. Catania, et al. Web Services Events (WS-Events) Version 2.0, Jul 2003.

[3] K. Clineand, et al. Toward Converging Web Service Standards for Resources, Events, and Management, Mar 2006.

[4] T. Dierks and E. Rescorla. The Transport Layer Security (TLS) Protocol Version 1.2. RFC 5246 (Proposed Standard), Aug 2008.

[5] S. Graham, et al. Web Services Notification, Mar 2004.

[6] IEEE. *IEEE standard computer dictionary: a compilation of IEEE standard computer glossaries.* IEEE Computer Society Press, New York, NY, USA, Jan 1991.

[7] IHE Patient Care Device Domain. URL `http://www.ihe.net/pcd/`.

[8] K. Lesh, et al. Medical Device Interoperability-Assessing the Environment. In *Proceedings of the 2007 Joint Workshop on High Confidence Medical Devices, Software, and Systems and Medical Device Plug-and-Play Interoperability (HCMDSS-MDPNP '07)*, pages 3–12. IEEE Computer Society, Los Alamitos, CA, USA, Jun 2007.

[9] P. K. McKneely, F. Chapman, and D. Gurkan. Plug-and-Play and Network-Capable Medical Instrumentation and Database with a Complete Healthcare Technology Suite: MediCAN. In *Proceedings of the 2007 Joint Workshop on High Confidence Medical Devices, Software, and Systems and Medical Device Plug-and-Play Interoperability (HCMDSS-MDPNP '07)*, pages 122–130. IEEE Computer Society, Los Alamitos, CA, USA, Jun 2007.

[10] MD PnP Program. URL `http://mdpnp.org/MD_PnP_Program.html`.

[11] OASIS. Devices Profile for Web Services Version 1.1 – Public Review Draft 01, Jan 2009.

[12] OASIS. SOAP-over-UDP Version 1.1 – Public Review Draft 01, Jan 2009.

[13] OASIS. Web Services Dynamic Discovery (WS-Discovery) Version 1.1 – Public Review Draft 01, Jan 2009.

[14] OASIS Open. UDDI version 3.0.2, 2004.

[15] M. P. Papazoglou. *Web Services: Principles and Technology.* Pearson Education, 2008.

[16] S. Pöhlsen, C. Buschmann, and C. Werner. Integrating a Decentralized Web Service Discovery System into the Internet Infrastructure. In *Proceedings of the 6th IEEE European Conference on Web Services (ECOWS '08)*, pages 13–20. IEEE Computer Society, Los Alamitos, CA, USA, Nov 2008.

[17] S. Pöhlsen and C. Werner. Robust Web Service Discovery in Large Networks. In *Proceedings of the 2008 IEEE International Conference on Services Computing (SCC '08)*, volume 2, pages 521–524. IEEE Computer Society, Los Alamitos, CA, USA, Jul 2008.

[18] S. Pöhlsen, et al. Fair-Queued Ethernet for Medical Applications. In *Proceedings of the 2008 Seventh IEEE International Symposium on Network Computing and Applications (NCA '08)*, pages 152–159. IEEE Computer Society, Los Alamitos, CA, USA, Jul 2008.

[19] M. Strähle, et al. Towards a Service-Oriented Architecture for Interconnecting Medical Devices and Applications. In *Proceedings of the 2007 Joint Workshop on High Confidence Medical Devices, Software, and Systems and Medical Device Plug-and-Play Interoperability (HCMDSS-MDPNP '07)*, pages 153–155. IEEE Computer Society, Los Alamitos, CA, USA, Jun 2007.

[20] R. A. van Engelen and K. A. Gallivan. The gSOAP Toolkit for Web Services and Peer-to-Peer Computing Networks. In *Proceedings of the 2nd IEEE/ACM International Symposium on Cluster Computing and*

the Grid (CCGRID '02), pages 128–135. IEEE Computer Society, Los Alamitos, CA, USA, May 2002.

[21] R. A. van Engelen and W. Zhang. An Overview and Evaluation of Web Services Security Performance Optimizations. In *Proceedings of the 2008 IEEE International Conference on Web Services (ICWS '08)*, pages 137–144. IEEE Computer Society, Los Alamitos, CA, USA, Sep 2008.

[22] W3C. Web Services Description Language (WSDL) 1.1, Mar 2001.

[23] W3C. SOAP Version 1.2, Apr 2007.

[24] E. Zeeb, et al. WS4D: SOA-Toolkits making embedded systems ready for Web Services. In *Proceedings on the Second International Workshop on Open Source Software and Product Lines 2007 (OSSPL07)*. Limerick, Ireland, Jun 2007.

[25] R. Zitzmann and T. Schumann. Interoperable Medical Devices due to Standardized CANopen Interfaces. In *Proceedings of the 2007 Joint Workshop on High Confidence Medical Devices, Software, and Systems and Medical Device Plug-and-Play Interoperability (HCMDSS-MDPNP '07)*, pages 97–103. IEEE Computer Society, Los Alamitos, CA, USA, Jun 2007.