An Approach to Integrate Distributed Systems of Medical Devices in High Acuity Environments

David Gregorczyk[1], Stefan Fischer[1], Timm Busshaus[1], Stefan Schlichting[2], and Stephan Pöhlsen[3]

1 University of Lübeck
Institute of Telematics
Ratzeburger Allee 160
23562 Lübeck
{gregorczyk,fischer,busshaus}@itm.uni-luebeck.de

2 Drägerwerk AG & Co. KGaA
Smart Software Solutions
Research Unit
Moislinger Allee 53-55
23558 Lübeck, Germany
texttttstefan.schlichting@draeger.com

2 Dräger Medical GmbH
Research & Development
Moislinger Allee 53-55
23558 Lübeck, Germany
texttttstephan.poehlsen@draeger.com

Abstract

This paper presents a comprehensive solution to build a distributed system of medical devices in high acuity environments. It is based on the concept of a Service Oriented Medical Device Architecture. It uses the Devices Profile for Web Services as a transport layer protocol and enhances it to the Medical Devices Profile for Web Service (MDPWS) to meet medical requirements. By applying the ISO/IEEE 11073 Domain Information Model, device data can be semantically described and exchanged by means of a generic service interface. Data model and service interface are subsumed under the Basic Integrated Clinical Environment Specification (BICEPS). MDPWS and BICEPS are implemented as part of the publically available openSDC stack. Performance measurements and a real world setup prove that openSDC is feasible to be deployed in distributed systems of medical devices.

1998 ACM Subject Classification J.3 Life and Medical Sciences, C.2.2 Computer-Communication Networks

Keywords and phrases Integrated Clinical Environment, Devices Profile for Web Services, ISO/IEEE 11073

Digital Object Identifier 10.4230/OASIcs.MCPS.2014.15

1 Introduction

Modern operating rooms (ORs) and intensive care units (ICUs) are equipped with numerous medical devices delivered by different manufacturers. While the amount of devices continuously increased over time, interoperability has not been adapted in the same way [14]. However, interconnecting interoperable medical devices can improve patient safety and optimize clinical workflows by providing the right information at the right time, in the right amount, at the right location and in the necessary quality [21]. As a result it makes the clinical work much easier and saves money.

Typical use cases are characterized by medical device safety-interlocks, remote control or data exchange with clinical information systems (CISs). Safety-interlocked devices perform a reciprocal monitoring to increase patient safety. For example, stopping an infusion at a predetermined blood pressure value or to prevent intra-abdominal CO2 insufflation if the heart rate and blood pressure are unmonitored [7]. Remote control means that designated device parameters can be controlled by remote devices. For example, muting a monitoring device's alert system or regulating the power of an ultrasound cutting device from a central OR cockpit. CIS communication comprises on the one hand a medical device providing data for, and on the other hand a medical device consuming data from a CIS or any electronic medical record system. Here, the most prominent example is the aquisition of patient demographics and other patient related data to be used during surgery.

The IEEE refers to interoperability as the ability of two or more IT systems to exchange information and to utilize the information that has been exchanged [9]. As a consequence, to achieve interoperability it is both important to assure error resistent data transmission and correct data interpretation. Assurance of data interpretation is twofold. Syntactic interpretation offers consistent data exchange according to an underlying specification, whereby semantic interoperability is the ability to interpret information exchanged with other systems, and to make effective use of it [8].

Unfortunately, in current distributed systems of medical devices interoperability is commonly achieved by using proprietary, vendor-dependent communication protocols and middleware. Products like Storz OR1 [12] or Olympus ENDOALPHA [17] provide fully integrated ORs. However, these systems are restricted to specific vendors and product models. Furthermore, integration after deployment at the Point-of-care (PoC) is a cumbersome task since in medical applications there is typically no system integrator with expert-level technical knowledge available [13]. In summary, realizing the aforementioned use cases is cost-intensive and will lead to isolated applications.

To enable *heterogeneous* interoperability, we introduce a future-proof, open and efficient architecture, protocol stack and middleware which is designed to satisfy functional and non-functional requirements on distributed systems of medical devices. The remainder of this paper is structered as follows. The second section illustrates related technologies and research projects. Section 3 describes the underlying conceptual model, requirements and a brief protocol overview. The overall conceptual model is described in Section 4. Sections 5 and 6 introduce the implementation and evaluation of the system. Our work is concluded in Section 7 which also gives an outlook on future work.

2 Related Work

In the past substantial standardization effort and several research projects have been carried out on medical device interoperability. The most popular standard is the ISO/IEEE 11073 (x73) [10]. It is separated into series 11073-1xxxx to 11073-7xxxx, of which the first three are the most important ones. The first part defines fundamentals for all subsequent parts, containing language elements, a nomenclature and an object-oriented Domain Information Model (DIM). The second part describes message exchange patterns between medical devices referring to the upper application layers of the ISO/OSI model. Physical interfaces are described as part of the third serie. However, most often only the DIM and nomenclature part are referenced due to the fact that underlying transport protocols do not

Table 1 Functional requirements (left) and non-functional requirements (right).

Plug & Play	**Risk Management**
Discovery & binding	Safe communication
Device capability description at runtime	Access control
Standardized protocols and open data access	Trust establishment between participants
Communication (1-1, 1-n, n-m)	**Privacy of patient-related data**
Event notification	**Latency in milliseconds**
Data reporting	
Remote control	

support the needs of current distributed systems of medical devices. As of today, x73 is rarely implemented by medical device manufacturers.

Another important work was done by Goldman et al. from the United States as part of the Medical Device "Plug-and-Play" Interoperability Program (MDPnP). They have published the ASTM F2761-1:200 Integrated Clinical Environment (ICE) standard [3]. It defines functional elements for PoC related IT systems, especially focusing on communication of patient data and on equipment command and control. Though the ICE standard gives sophisticated information on conceptual system design, no concrete technical specifications and implementations have been created yet.

There is also much research done in Germany. All roads run together in the OR.NET project [19] that began in September 2012. It combines and consolidates the concepts of predecessor projects to develop a proposal for a new standard called Open Surgical Communication Protocol (OSCP). Foundational predecessor projects are SOMIT FUSION [23], SOMIT OrthoMIT [24], Smart.OR [22], TiCoLi [26], TeKoMed [25] and DOOP [5]. All of these projects propose an architecture based on the idea of a Service-oriented Medical Device Architecture which is described in Section 3. The OASIS standard *Devices Profile for Web Services* serves as the fundamental transport protocol providing TCP/IP and UDP transport bindings, decentralized service discovery and eventing capabilities.

3 SOA for Medical Devices

The middleware presented in this paper follows the principles of the well-known Service-oriented Architecture (SOA) paradigm [15]. Service providers offer their capabilities by means of machine-readable service descriptions and publish them to a service directory. Service consumers can discover these services by using the service directory. Afterwards, they dynamically bind to suitable service providers and invoke their operations.

If SOA principles are applicated on device communication, it is called a Service-oriented Device Architecture (SODA) [4]. In a SODA, services encapsulate both a device's functionality and physical user interface. They are then called device services. In addition to the regular SOA request-response model, publish-subscribe systems are used to transmit information between service providers and consumers. In publish-subscribe systems the communication direction is reversed such that service providers start communicating with service consumers.

If a SODA is applied to a distributed system of medical devices, it is defined as a Service-oriented Medical Device Architecture (SOMDA). Device services are then called medical device services. Middleware systems which implement a SOMDA have certain functional and non-functional requirements to satisfy. Requirements have been acquired by Dräger and are listed in Table 1. Since SOMDA is an abstract concept, implementation directives are

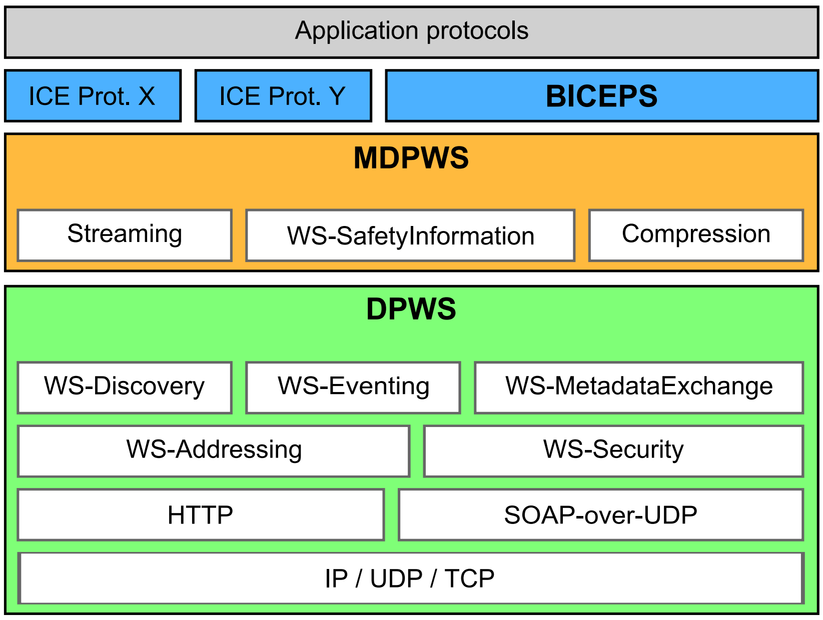


Figure 1 The openSDC protocol stack.

required to get a corresponding middleware up and running. The middleware introduced in this paper is based on Web Services and is called openSDC [18]. Figure 1 shows the protocol stack that is implemented in openSDC.

The most general protocol is the Devices Profile for Web Services (DPWS) [6]. A Web Services profile contains a certain set of specifications and defines appropriate constraints to eliminate protocol ambiguities. DPWS was first invented by Microsoft to interconnect network devices and PCs in a plug-and-play-like fashion. In particular, it was proposed to automatically detect network printers and download suitable drivers. DPWS comes with decentralized service discovery, eventing capabilities and is based on open standards. Hence, it meets the features of a SODA and commonly fulfills the requirements *Plug & Play* and *Communiction (1-1,1-n,n-m)* listed in Table 1.

DPWS is not sufficient to meet the complete set of requirements for a SOMDA. Therefore, openSDC implements a special DPWS dedicated to medical software: the Medical Devices Profile for Web Services (MDPWS). It provides streaming capabilities, error-resistent data transmission and compression options. Besides MDPWS, the Basic Integrated Clinical Environment Protocol Specification (BICEPS) provides a domain-specific protocol to offer generic and extensible device access and modeling. MDPWS and BICEPS are described in detail in the next section.

4 Conceptual Design

A central aspect of the work presented in this paper is the capability to be extensible for future communication needs. For this purpose a layered protocol stack has been developed. As shown in Figure 1, it is separated into three different layers. In conjunction with DPWS, MDPWS builds the transport layer to securely transmit messages. BICEPS introduces an extensible message information model (MIM) and access services to facilitate medical device interoperability. With changing network conditions, MDPWS can be replaced by upcoming, improved or redundant transport protocols, without altering the domain and message model.

4.1 MDPWS

MDPWS is based on DPWS version 1.1 [6]. DPWS became an OASIS standard in June 2009 is used by openSDC to provide Web Services-based features like:

- Service-oriented interaction
- Service discovery using WS-Discovery [16]
- Service description using WSDL [2]
- Publish-Subscribe using WS-Eventing [1]

One major requirement of openSDC is to be extensible for future communication needs. DPWS (and the underlying Web Services technology) is designed for extensibility. But it is an enabling technology rather than a comprehensive communication specification. Hence, DPWS only provides basic features which are constrained and enhanced by MDPWS to enable safe communication in distributed systems of medical devices in high acuity environments. The extended features of MDPWS are explained in the following subsections. In some cases, fundamental knowledge of DPWS may be beneficial.

4.1.1 Liveliness

To assure that a medical device is still active, MDPWS defines to send a directed Probe message as specified in DPWS.

4.1.2 Streaming

A typical use-case in PoC scenarios is the transmission of vital parameters by using waveforms. To enable efficient waveform transmission using Web Services, MDPWS includes SOAP-over-UDP and WS-Streaming. WS-Streaming is part of the MDPWS specification and has not been published yet. It defines a policy to embed descriptive information on streams provided by a Web Service. WS-Streaming does not explicitly define stream management and transport, but provides the capability to announce the existence and type of a stream.

4.1.3 Safety and Security

Any communication protocol dealing with distributed systems of medical devices has to avoid impairment of patient safety. Therefore, remote control mechanisms have to be at least single fault safe. Moreover, another risk control measure is the exchange of a remote invocation context.

Single fault safety can typically be met by providing dual channel transmission. To remotely modify device parameters, invocation information is sent redundantly over two independent channels. The second channel might be separated in time or representation. In time means that the second channel is established temporally after the first channel has been transmitted. The drawback of separation in time is that it requires stateful services and double transmission costs. Separation in representation is achieved by providing both channels in a single message. The service provider detects failures, for example, by means of an invalid checksum. Figure 2 illustrates the dual channel transmission in accordance to separation in representation. Data is given to the middleware of the service consumer by means of two input objects. The middleware serializes and deserializes one input object and compares the result to the second object. This implicates that parser end serializer are working correctly. The serialized object is transmitted to the service provider. On the

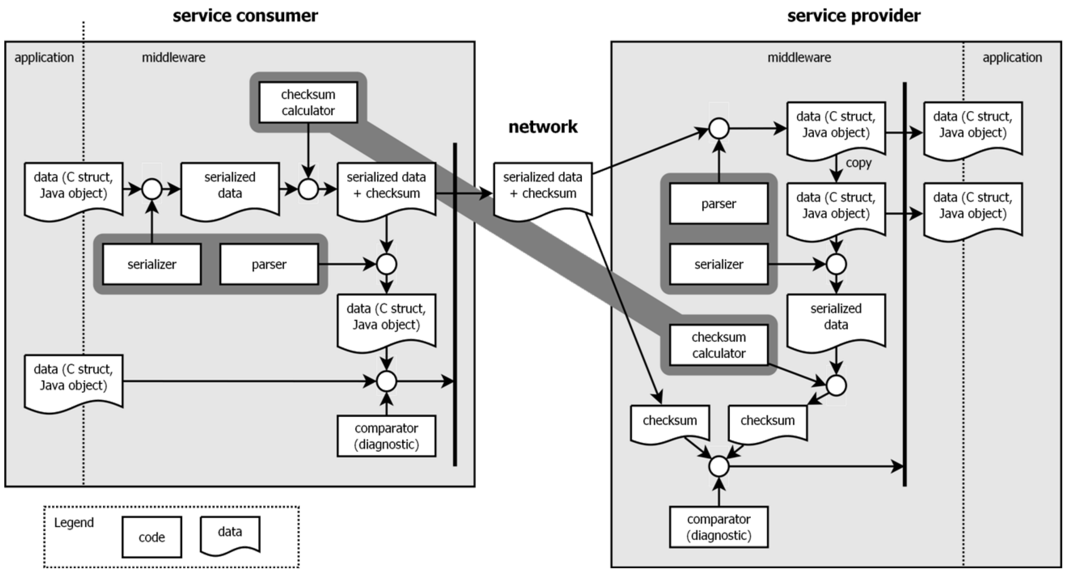


Figure 2 Dual channel transmission using separation in presentation.

provider side, the parsing process and the data duplication generating the second channel are controlled. In summary, this guarantees that data is not compromised when being transmitted from one process to another. More information on dual channel transmission is given in [20].

A remote invocation context is also called a safety context. The service consumer needs to know which information is required by the service provider to transmit a remote invocation context. If this knowledge is available to the service consumer, it can append context information to the service invocation message. An example for a safety context is the value of a setting an operation is applied to. To enable dual channel transmission and safety contexts, WS-SafetyInformation has been specified as part of MDPWS. It is not standardized yet.

4.1.4 Security

Besides patient safety, access control, integrity and confidentiality are major security goals. Only authorized service consumers should have remote access to devices. Patient data should be secured by using encryption. MDPWS provides capabilities to meet the aforementioned requirements. It includes a Public Key Infrastructure (PKI) to gain authorization capabilites using X.509 certificates. If access control is needed, it is handled on the level of individual security principles using the PKI. MDPWS specifies that a service provider may control access to a service by HTTPS with mutual authentication. This also applies to WS-Eventing services.

MDPWS specifies that a X.509 certificate is issued to a service provider's Universally Unique Identifier (UUID) [11]. The same is prescribed to service consumers. Authorization is enabled by using a Device Access Control List (DACL) which contains subject identities of security principals and associated granted access rights. In order to be able to handle groups of security principals, DACL provides group subject identities which are used to grant access to, for example, device types or devices of certain manufacturers. In order to ensure decentralization, devices have to maintain root certificates of the manufacturers for every device they are communicating with. Another solution is to get a separated DACL which is signed by the clinical operators to verify digital signatures of communication partners. So far, devices of a dedicated manufacturer trust every other decvice of the same manufacturer.

To ensure confidentiality and integrity of messages, MDPWS states that HTTPS should be applied. SOAP-over-HTTPS may also be used. Since SOAP-over-UDP is used to transmit anonymous streaming data, no security mechanism has been defined by MDPWS.

4.2 BICEPS

BICEPS specifies a MIM and access services for the domain of distributed medical devices. It is based in large parts on the x73 DIM as described in Section 2 and forms a minimal set of generic functionality and messages to facilitate interoperability and extensibility. A core component of x73 and even BICEPS is the Medical Device Information Base (MDIB). It represents an object-oriented model to encapsulate managed medical objects. Managed medical objects are best known as physiological patient data, device configuration data, or remote invocation operations.

4.2.1 MIM

Communication messages exchanged in distributed systems of medical devices contain state data about clinical measurements of a patient or the device associated with a patient. Moreover, remote invocation commands might also be transmitted. To enable interoperability, medical devices have to exchange meta-information about state data as well as contextual information that describes in which context state data has been acquired. Such information is described in the BICEPS MIM. It provides two parts: communication message definitions and the MDIB component. For the sake of brevity the communiction messages are not described in this paper. Basically, for every service request and response, and even for every notification, BICEPS-MIM defines a dedicated message payload.

The MDIB in turn is divided into a descriptive part and a state part. The descriptive part holds information on a device's structure, services and metrics, and provides coded values. Coded values allow to characterize object types by referencing a coding system with a version and code identifier. By means of coded values communication partners are able to interpret transmitted data, for example, the unit of a measurement value. Since every managed medical object is equipped with a coded value, they can be semantically interpreted. On the other side, the state part holds the content data that a medical device can deliver. It should be noted that both parts of the MDIB can change during runtime.

A top level overview of the BICEPS-MIM MDIB descriptive part is given in Figure 3. A Medical Device System (MDS) is an abstract representation of a physical device that exposes its capabilities as a medical device service. In accordance to x73 it is depicted as an HydraMDS and may contain multiple Virtual Medical Devices (VMDs). A VMD is a representation of a sub-system of a MDS. It may in turn contain multiple channels. Channels refer to a group of metrics, whereby metrics are abstract representations of measurement values, settings or status items. BICEPS-MIM specifies by default numeric values, textual values and sample arrays of numeric values. Nevertheless, it can be extended to any type of value. MDSs, VMDs and channels can be assigned with an optional alert system. It detects alert conditions and may signal them by means of alert signals. Alert signals are in most cased displayed visually or acousticly.

Another part of the model in Figure 3 is the Service Control Object (SCO). It comprises remote invocation capabilities. This includes affected objects and Quality of Service (QoS)

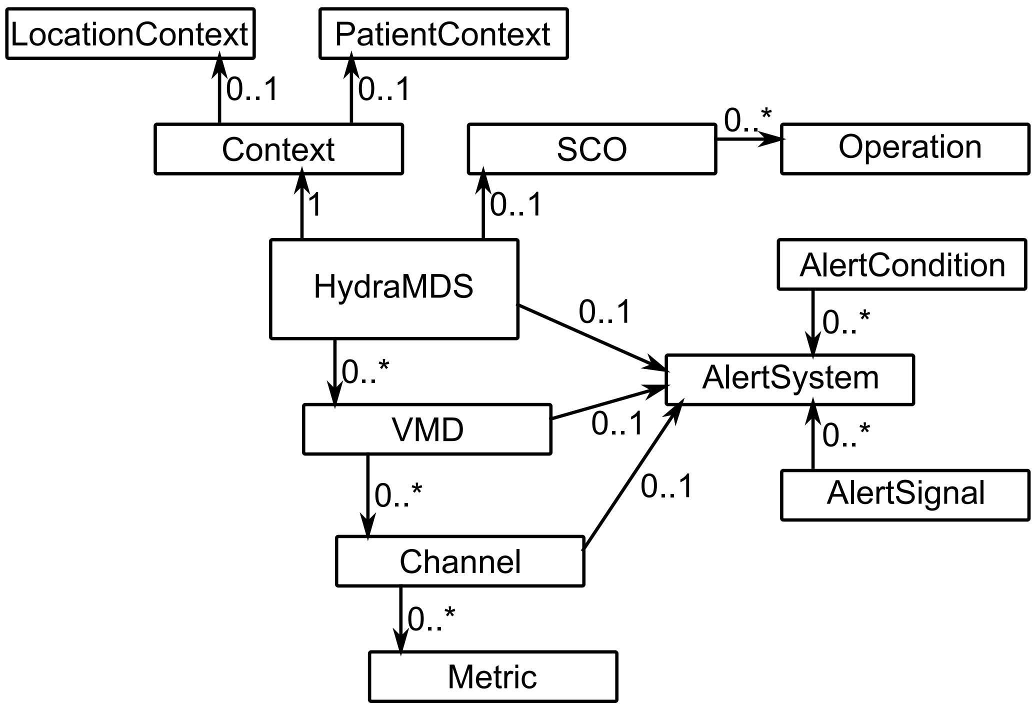


Figure 3 Top level overview of the BICEPS-MIM MDIB descriptive part.

parameters. There are different operation types to offer several set methods, non-generic method calls or remote control calls. The Context element represents the context the underlying MDS is currently working in. This context is designated by patient or location information.

4.2.2 Service interface

To get remote access to managed medical objects, x73 specifies a generic service interface called CMDISE. In a slightly different way this interface is also provided by BICEPS. It contains a generic get service to request data, a generic set service to manipulate data, a dedicated waveform service to transmit, for example, vital signs, a protected health information (PHI) service to request or set patient related information in a secured manner, and an event report service to enable publish-subscribe data retrieval. Figure 4 depicts the differences between CMDISE and BICEPS services. While x73 separates SET and ACTION, BICEPS merges them to a single service. BICEPS omits CREATE and DELETE since they provide extended functionality in terms of allowing to manipulate device memory. If CREATE and DELETE are required in sophisticated scenarios, they can later be implemented by BICEPS' extensibility mechanism. Waveform and PHI are extra services provided by BICEPS. Finally, EVENT REPORT is mapped from x73.

Figure 4 illustrates a coarse-grained service interface. BICEPS uses numerous operations grouped together to build GET, SET, Waveform, PHI and EVENT REPORT. GET is separated into:

- **GetMDIB:** retrieval of the full MDIB including descriptive and state part
- **GetMDDescription:** retrieval of the MDIB descriptive part
- **GetMDState:** retrieval of the MDIB state part

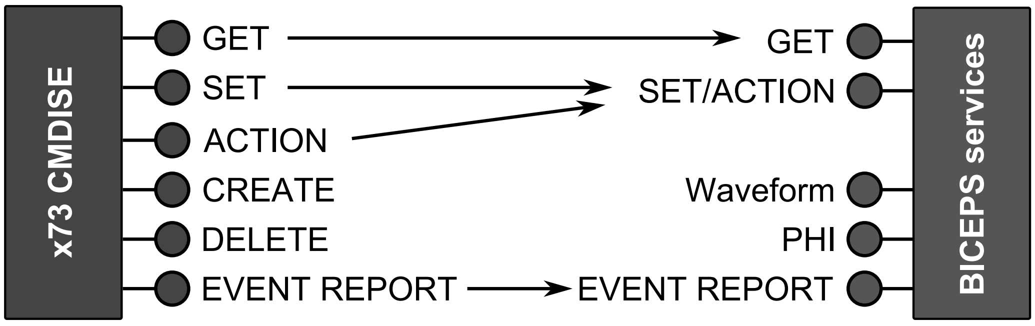


Figure 4 Coarse-grained mapping of Common Medical Device Information Service Element (CMDISE) and BICEPS service interfaces.

SET is characterized by:

- **SetValue:** sets a numeric metric
- **SetString:** sets a string metric
- **SetRange:** sets the range of a numeric metric
- **Activate:** executes remote control

EVENT REPORT is the capability to retrieve notifications about either changes of the descriptive part or the state part. The EVENT REPORT service is divided into

- **MetricReports:** notification of metric changes
- **AlertReports:** provides alert events
- **ContextReport:** notifies when context changes
- **OperationInvokedReport:** since operation calls are queued, this notification provides operation progress information
- **MDSCreatedReport:** notifies if a MDS appears to be available for access
- **MDSDeletedReport:** notifies if a MDS disappears and is no longer active
- **ObjectCreatedReport:** notifies on any object creation events
- **ObjectDeletedReport:** notifies on any object deletion events

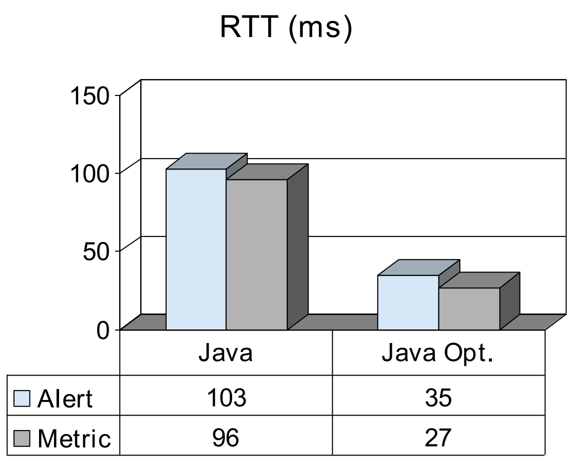
There is no definition on how to subscribe to notifications. BICEPS states that the underlying transport protocol has to support subscription management.

Waveform is an optional service that defines an interface to retrieve a stream for real-time sample array metrics. There is only a waveform stream manifested in BICEPS even without specifying means to subscribe to it. BICEPS states that the underlying transport protocol has to support subscription management.

Like Waveform, PHI is also an optional service. It allows retrieval and modification of patient data. Due to privacy reasons this service is separated from other BICEPS services. Hence, it can separately be protected.

Besides the generic service interface medical devices might offer remotely invokable operations that are not defined as part of the BICEPS-MIM / BICEPS services. For this purpose a non-generic operation descriptor can be defined in the descriptive part of the MDIB. It facilitates non-generic, proprietary service calls to be represented by a coded value. By using the non-generic operation descriptor, service calls are made accessible through the MDIB.

Finally, BICEPS does not define any QoS requirements, but provides extensibility points to embed QoS requirements for a transport protocol.



	Java	Java Opt.
▢ Alert	103	35
▨ Metric	96	27

Figure 5 Maximum round trip time performance measurements for openSDC using JRE6's default (Java) and optimized (Java Opt.) environment.

5 Implementation

BICEPS and MDPWS have been released as part of openSDC [18] and can be downloaded and used for free. OpenSDC serves as a reference implementation for vendors of medical devices, but is not intended to be used in clinical trials, clinical studies or in clinical routine.

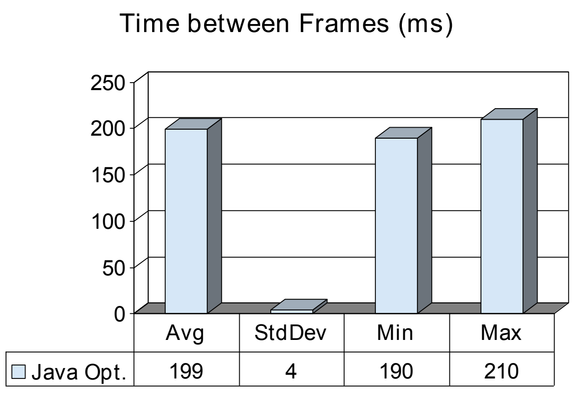
The openSDC project uses a modified version of the WS4D JMEDS stack [27] to gain DPWS functionality. The modified JMEDS stack is called JDPWS. The remaining components of the openSDC stack implement the BICEPS and MDPWS protocols. It allows either a contract-first approrach by defining the MDIB in a XML file and load it into the framework, or a code-first approach by building the MDIB in Java code.

6 Evaluation

To get an impression on how stable and powerful openSDC is, performance measurements have been made using a Java JRE6. A client (2x2 GHz) requests metric values and alerts twice a second. The device (1x1.1 GHz) responds with 276 metric values and 80 alerts. Additionally, 10 waveform frames containing at least 20 values are sent out at 200 ms intervals.

The first measurement is the maximum round trip time (RTT) to deliver an alert and a numeric metric. It is depicted in Figure 5. When using Java with its default settings, openSDC causes a maximum RTT of 103 ms to post an alert and 96 ms to post a metric. By optimizing the JRE using Java's environment properties, open SDC is on average 69 percent faster. Figure 6 shows the average and maximum framerate when transmitting a waveform. In this scenario, only optimized Java has been tested. With an average framerate of 199 frames per second, openSDC's Web Service Streaming is fast enough to build continuous 1-dimensional signals.

In addition to the previously mentioned simple performance measurements, openSDC has been tested with real world medical devices as part of a public project workshop in December 2013. Figure 7 gives an idea of the demonstration setup. Participating devices were a Möller-Wedel operating microscope, a Localite navigation system, a Dräger monitoring device, a Söring ultra sound surgery device and an Olympus documentation system.



	Avg	StdDev	Min	Max
□ Java Opt.	199	4	190	210

Figure 6 Maximum waveform framerate for openSDC using JRE6's optimized (Java Opt.) environment.

Different use cases have been introduced to prove the practical effect of openSDC. First, patient data arrived at the documentation system by fetching HL7 ADT messages. Afterwards, this data was automatically stored at the microscope and navigation system side using DPWS. Next, BICEPS was used to transmit vital signs from the patient monitor to the operating microscope to display health information. Moreover, it was applied to remotely control the ultrasound surgery device. By sending activation requests, the operating microscope was able to control cutting. The navigation system retrieved zoom and focus information from the operating microscope. Thereby, the navigation system was able to select proper views of the area currently navigated in.

In all scenarios openSDC provides immediate network responses confirming the measurements in Figure 5. Especially, there was no remarkable delay when remotely controlling the ultra sound activation. Therefore, the system is fast enough to remotely control devices where controlling actions have to be initiated and recognized by a human.

7 Conclusion and Future Work

In this paper, we have presented openSDC, a protocol stack which enables heterogeneous interoperability of medical devices. Based on a set of standards, most prominently including the Device Profile for Web Services (DPWS), we presented in detail the MDPWS protocols for message transport in medical environments and BICEPS which is used as an application protocol for service access. An openly available implementation of openSDC exists, and first evaluations show promising results.

We see future work mostly in three directions: First, BICEPS will be enhanced by extending its functionality through providing more plugins. Second, more clinical use cases will be described and show-cased. Finally, we are working on a seamless integration of openSDC with clinical IT systems. Especially authentication and authorization will be a major issue the solution of which will make the system much easier and more secure to use.

References

1 Don Box, Luis Felipe Cabrera, Craig Critchley, Francisco Curbera, Donald Ferguson, Steve Graham, David Hull, Gopal Kakivaya, Amelia Lewis, Brad Lovering, Peter Niblett, David Orchard, Shivajee Samdarshi, Jeffrey Schlimmer, Igor Sedukhin, John Shewchuk, Sanjiva

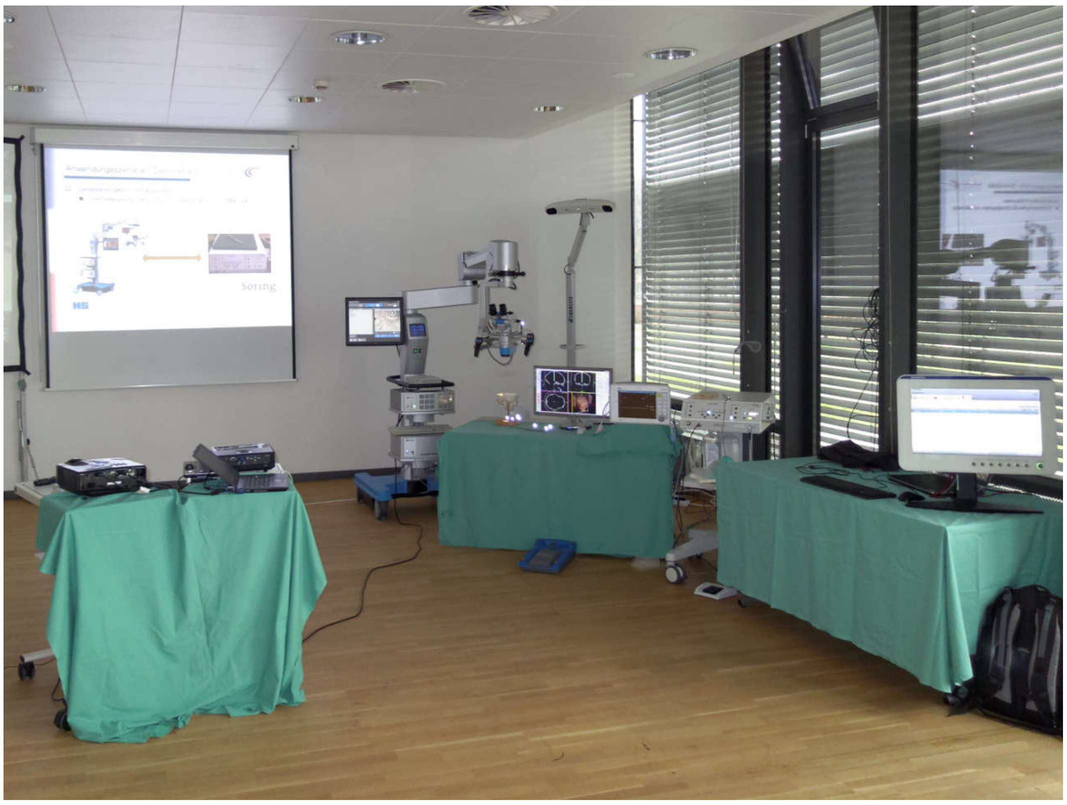


Figure 7 Image of an openSDC powered distributed system of medical devices at a project workshop in December 2013. F.l.t.r.: operating microscope, navigation system, patient monitor, ultrasound surgery device and documentation system.

Weerawarana, and David Wortendyke, editors. *W3C Member Submission: Web Services Eventing (WS-Eventing).* World Wide Web Consortium (W3C), July 2006.

2 Erik Christensen, Francisco Curbera, Greg Meredith, and Sanjiva Weerawarana, editors. *Note: Web Services Description Language (WSDL) 1.1.* World Wide Web Consortium (W3C), March 2001.

3 Committee F29.21 on Devices in the Integrated Clinical Environment. ASTM F2761 – 09 Medical Devices and Medical Systems – Essential safety requirements for equipment comprising the patient-centric integrated clinical environment (ICE) – Part 1: General requirements and conceptual model. ASTM International, 2009.

4 Scott de Deugd, Randy Carroll, Kevin Kelly, Bill Millett, and Jeffrey Ricker. SODA: Service Oriented Device Architecture. *IEEE Pervasive Computing*, 5(3):94–96, c3, 2006.

5 DOOP project. `http://www.doop-projekt.de/`.

6 Dan Driscoll and Antoine Mensch, editors. *OASIS Standard: Devices Profile for Web Services Version 1.1.* OASIS, July 2009.

7 Julian M. Goldman. Medical Devices and Medical Systems – Essential safety requirements for equipment comprising the patient-centric integrated clinical environment (ICE) – Part 1: General requirements and conceptual model. ASTM International, `http://www.mdpnp.org/uploads/F2761_completed_committee_draft.pdf`, 2008.

8 Healthcare Information and Management Systems Society – HIMSS. What is Interoperability? `http://www.himss.org/library/interoperability-standards/what-is`, April 2013.

9 IEEE Standard Computer Dictionary. A Compilation of IEEE Standard Computer Glossaries, January 1991.

10 ISO/IEEE. ISO/IEEE 11073: Health informatics – Point-of-care medical device communication, June 2004.

11 ITU International Telecommunication Union. Information technology – Open Systems Interconnection – Procedures for the operation of OSI Registration Authorities: Generation and registration of Universally Unique Identifiers (UUIDs) and their use as ASN.1 object identifier components, September 2004.

12 Karl Storz. OR1. `https://www.karlstorz.com/cps/rde/xchg/SID-949FE9D0-512F111A/karlstorz-en/hs.xsl/522.htm`.

13 B. Larson, J. Hatcliff, S. Procter, and P. Chalin. Requirements specification for apps in medical application platforms. In *Software Engineering in Health Care (SEHC), 2012 4th International Workshop on*, pages 26–32, June 2012.

14 Kathy Lesh, Sandy Weininger, Julian M. Goldman, Bob Wilson, and Glenn Himes. 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 (HCMDSSMDPnP)*, pages 3–12, Cambridge, Massachusetts, USA, June 2007. IEEE Computer Society.

15 C. Matthew MacKenzie, Ken Laskey, Francis McCabe, Peter F Brown, and Rebekah Metz, editors. *OASIS Standard: Reference Model for Service Oriented Architecture 1.0*. OASIS, October 2006.

16 OASIS. Web Services Dynamic Discovery (WS-Discovery) Version 1.1, 2009.

17 Olympus. ENDOALPHA. `http://www.olympus-europa.com/medical/en/medical_systems/products_services/systems_integration/productselector_service_solutions_8.jsp`.

18 openSDC Website. `https://sourceforge.net/projects/opensdc/`.

19 OR.NET. OR.NET | Projektwebsite. `http://www.ornet.org`.

20 Stephan Pöhlsen, Winfried Schöch, and Stefan Schlichting. A Protocol for Dual Channel Transmission in Service-Oriented Medical Device Architectures based on Web Services. In *3rd Joint Workshop on High Confidence Medical Devices, Software, and Systems & Medical Device Plug-and-Play Interoperability*, 2011.

21 Andreas Schweiger, Ali Sunyaev, Jan Marco Leimeister, and Helmut Krcmar. Toward Seamless Healthcare with Software Agents. *Communications of the Association for Information Systems (CAIS)*, pages 692–709, 2007.

22 smartOR project. `http://www.smartor.de/`.

23 SOMIT FUSION project. `http://www.somit-fusion.de/SF/`.

24 SOMIT orthoMIT. `https://www.vde.com/de/fg/DGBMT/Arbeitsgebiete/Projekte/Seiten/SOMIT-orthoMIT.aspx`.

25 TeKoMed project. `http://kosse-sh.de/projekte/tekomed/`.

26 TiCoLi. `http://www.iccas.de/ticoli/`.

27 Web Services for Devices (WS4D). JMEDS (Java Multi Edition DPWS Stack). `https://sourceforge.net/projects/ws4d-javame/`.