Please quote as: Mauro, C.; Sunyaev, A.; Leimeister, J. M. & Krcmar, H. (2010): Standardized device services - A design pattern for service oriented integration of medical devices. In: Hawaii International Conference on System Sciences (HICSS) 2010, Kauai, USA.

Standardized Device Services – A Design Pattern for Service Oriented Integration of Medical Devices

Christian Mauro[1] Ali Sunyaev[1] Jan Marco Leimeister[2] Helmut Krcmar[1]

[1]Technische Universitaet Muenchen
Information Systems
Boltzmannstrasse 3
85748 Garching, Germany
{mauro, sunyaev, krcmar}@in.tum.de

[2]Universitaet Kassel
Information Systems
Nora-Platiel-Strasse 4
34127 Kassel, Germany
leimeister@uni-kassel.de

Abstract

Service oriented device architecture (SODA) is a promising approach for enabling a continuous IT support of medical processes in hospitals. However, there is a lack of specific design patterns for realizing the concept in an effective and efficient way. This paper addresses this research gap by introducing the Standardized Device Service design pattern, as a first fundamental pattern for encapsulating devices as services. The pattern is based on both established Service Oriented Architecture (SOA) best practices as well as latest research in the field of SODA. This paper contributes to a) the extension of the IT support of medical processes by devices, b) the general concept of SODA by addressing the lack of generalized design concepts, and c) the existing catalog of SOA design patterns by introducing a first pattern for device integration.

1. Introduction

The standardization of medical treatment is an important way of improving quality of care and reducing costs in hospitals, e.g., by introducing clinical pathways [6]. From an information logistics point of view, standardized medical processes define concrete information needs of the respective actors along the treatment path. This enables the automated provision of information according to the principle of information logistics, i.e., the right information, at the right time, in the right amount, at the right location and in the necessary quality [25, 41].

In order to realize this vision of seamless healthcare with horizontally and vertically integrated healthcare processes enabled by seamless IT support, all participating IT (Information Technology) systems (i.e. software systems and medical devices) have to be integrated [41]. Process oriented IT architectures like SOA (Service Oriented Architecture) are often proposed as a way to support the integration of heterogenous software systems and with it the IT support of standardized medical processes [3, 32, 33]. However, the integration of medical devices is not addressed by these works.

An emerging idea is the application of service oriented principles to enable the integration of devices [7, 8, 21, 38, 43], which we refer to as Service Oriented Device Architecture (SODA) in this paper. Devices are encapsulated as services, analogous to enterprise services in SOA in order to create interoperable virtual devices (device services) [7, 8, 21, 38, 43], which can be used within IT supported processes [5, 15, 21, 22]. As Lesh et al. summarize, "Interoperability is an almost non-existent feature of medical devices" [27]. Thus, the SODA concept is a promising approach to overcome these interoperability issues [38, 43].

As we showed with a literature review, the research field of SODA is mostly unexplored, particularly with respect to medical devices [30]. The main subjects of the identified works are proof of concepts (e.g., [5, 21]) and process orchestration (e.g., [5, 15, 22]) for specific use cases. A structured examination of possible architectural design concepts based on existing SOA design patterns could not be found. In another work (currently in submission for publication) we analyzed existing SOA design patterns [11, 12] with regard to their applicability for SODA. It was shown that existing patterns cannot resolve all design problems.

This paper proposes a first SOA design pattern for SODA. As this research is part of a design science research project, in accordance with Hevner et al. [18], the results are based on existing research knowledge, especially on existing (SOA) design patterns. In addition, evaluation and improvement of the artifact – the new pattern - is part of the research project. However, within the scope of this paper, only the technical feasibility and a first conceptual evaluation

can be demonstrated. The evaluation of different ways of implementation, and the overall evaluation of the suggested pattern are the objectives of further research.

This paper contributes to:

a) the extension of the IT support of medical processes by devices by proposing a pattern that uncouples the device service consumer from the proprietary device interface,
b) the general concept of SODA by proposing a first design pattern that is applicable independently of a specific domain or use case,
c) the existing catalog of SOA design patterns by extending it to the integration of devices.

This paper is structured as follows. First, section 2 presents the concept of SODA. Then, the most important SOA fundamentals are described in section 3, on which the research is based. Section 4 introduces the new Standardized Device Service pattern. In section 5, an exemplary scenario, which was implemented in our laboratory, demonstrates the feasibility of the pattern. A conceptual evaluation of the pattern is presented in section 6. Section 7 summarizes the paper and calls for further research.

2. Service Oriented Device Architecture

2.1. General concept

The basic concept of SODA is the encapsulation of devices as services, analogous to enterprise services in SOA. An enterprise service is a software component that offers a business functionality on a highly semantical level by specifying the interface in a standardized way [24]. A highly semantical level refers to a service that is self-descriptive in a way that it can be consumed dynamically and loosely coupled by other components with a consistent understanding of shared data. In the medical domain, a device service, for instance, could offer functionality for measuring the current blood pressure of a patient. Based on such basic services, more complex services (like a patient monitoring system) can be realized.

The main advantage of the service oriented approach is that the manufacturer-specific device interface does not have to be known by the service consumer and by the programmer, as it is encapsulated by a standardized service interface. This enables the extension of IT supported medical processes by devices. In addition, new functionalities could be added to the device service, which are logically related to the device but not offered by the device itself (e.g., tracking & tracing functionalities); thus, the device service can be considered as a virtual device. Therefore, software maintenance becomes easier because the service interface remains unchanged in case of a device exchange or device interface changes.

2.2. State of the art

In previous work a literature review was performed to examine the state of the art concerning SODA, especially in the health care domain [30]. By using a keyword search in the most important journals and conferences in the field of information systems, computer science, economic science, medical informatics and medical science, 26 relevant articles were identified, predominantly in a technical format. The feasibility and usefulness of the general concept was shown in all examined sources, and fundamental results shown by the various authors.

However, much room for further research was identified, with the primary research gaps categorized into three layers [30]:

- *Requirements Layer*: Analyses of requirements could not be found, taking the specifics of device services into consideration.
- *Technical Layer*: Architectures, used in existing works for instantiating the SODA concept, were built from scratch (e.g., [43]). Although implementation details (like selected technologies) were part of the studies (e.g., [21]), general design concepts were not included. The use of existing SOA design patterns was not documented (or not considered), and the development of new patterns with respect to device integration was not addressed.
- *Methodology Layer*: No analyses were found that indicated whether existing SOA methodologies (like methods for service identification) were transferable to device services.

Concerning the requirements layer, analyses of the specifics of device services are part of another work that is currently in submission for publication. The results are summarized in the next section.

This paper focuses on the technical layer, and more specifically, on the general design concepts for SODA. However, technologies themselves are not examined. The research is built upon the findings of the identified existing works. Thus, further results of the state of the art analysis are presented at the respective sections.

2.3. Specific requirements for device services

Contrary to software services, device services are based on physical devices. Thus, the following characteristics of devices influence the specific requirements of device services:

Mobility: Several devices are designed to be moveable; thus, the device (and with it the service)

might temporally not be available (e.g., portable ultrasound devices [29]).

Locality: The locality of devices might be important in several cases, e.g., in hospitals the mapping of a device to a patient could be based on the locality of the device. In addition, the quick localization of a device can be important, e.g., respirators [26]. The locality, naturally, correlates with the mobility.

Manual influences: Devices can be directly influenced by human beings. Settings can be changed and, most importantly, devices can be switched on or off at any time [28].

Replacement: Devices might be replaced by others (other manufacturer, model or version) in case of failure or for reasons of maintenance [31].

Devices as resource: Devices are physical resources. Therefore, they can be reserved or they might be designed for exclusive access only [17].

Hardware interfaces: When integrating legacy software systems, proprietary interfaces are a significant challenge. When integrating devices, apart from software interfaces, hardware interfaces also have to be taken into consideration [39].

Software changeability: The internal software of devices often cannot be changed or is not allowed to be changed. For instance, medical devices are certified. Any changes, not permitted by the manufacturer, would result in a loss of certification [14].

From these characteristics of devices, the following specific requirements for device services can be deduced, that is, device services must be able to:

- dynamically handle different kinds of device interfaces, which usually cannot be influenced,
- manage the fact that devices can suddenly be not accessible at any time and
- provide functionality for handling devices as physical resource (e.g., exclusive access and locality) if required.

When developing design patterns for device services, these requirements have to be taken into consideration. In addition, they should coincide with existing SOA best practices, which are presented in the next section.

3. SOA best practices

3.1. SOA design principles

Our approach is based on following SOA design principles, which can be found in different works [2, 9, 36, 42]:

Standardized Service Contract. Service contracts define the capabilities of services. They should be defined in a standardized way, preferring the contract-first approach (i.e., defining the contract before implementing the service). This principle ensures consistent and partly reusable service contracts.

Service Loose Coupling. This principle asks to avoid negative types of coupling as Contract-to-Functional Coupling, Contract-to-Implementation Coupling, Contract-to-Logic Coupling and Contract-to-Technology Coupling (for further information about these types of coupling, see [9]).

Service Abstraction. The application of this principle turns services into a black box. The use of volatile information (which can possibly change in future, e.g., when a service is composed of other services) by the service consumer is avoided by this principle.

Service Reusability. This principle focuses on ensuring robust and generic services, which can be reused on different positions within the IT infrastructure.

Service Autonomy. This principle advocates a maximum of control of a service over its underlying runtime execution environment. For example, this means that two services should not have direct access to the same data object on a database.

Service Statelessness. The management of state information consumes system resources and should therefore be avoided as far as possible.

Service Discoverability. This principle advises that services contain communicative meta data that enable discovery and interpretation.

Service Composability. Services should be designed in a way that enables them to be effective participants in service compositions.

In general, it is not possible to fulfill all SOA design principles. For instance, task services contain specific process logic, and are therefore not reusable in most cases [9, 24]. However, for maximizing the benefits of SOA, the solution design for device integration should, as much as possible, be in accord with the principles.

3.2. SOA patterns

In this paper, we present a first SOA design pattern for SODA. The idea of patterns can be traced back to Alexander [1] in the field of architecture and Gamma [13] in the area of software engineering. Basically, a pattern consists of the following elements [4, 40]: the *context* of a given problem and its circumstances, the description of the *problem* itself (also called forces), the proposed *solution* for the problem and *references* to related patterns.

Erl [12] extends this meta-definition of a pattern by using:

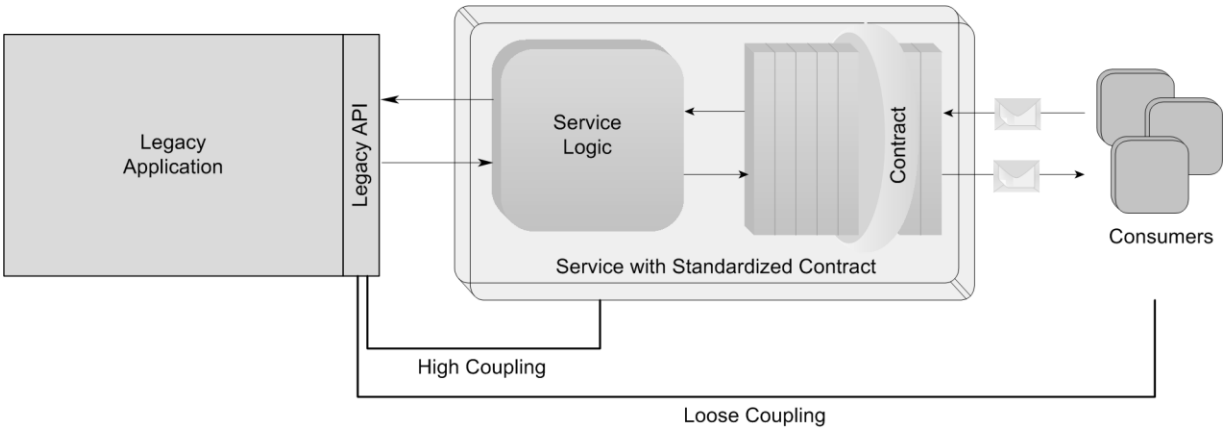

Figure 1: Legacy Wrapper pattern (according to [12])

- a *Pattern Profile* (consisting of a requirement definition, an icon, a summary table, a problem definition, a solution, an application description, impacts, relationships and a case study example) and
- a *Pattern Notation* (including specific symbols to represent different kinds of patterns, as well as different types of pattern figures to emphasize specific aspects of the pattern).

Due to space restrictions, the complete definition of the pattern profile of the Standardized Device Service pattern is not included in this paper. Instead, the pattern is introduced in the next section in a more consolidated and understandable way. Patterns contain generalized concepts for a specific problem. Thus, the Standardized Device Service pattern can be used as a guideline to design and implement device services, based on SOA best practices.

4. Standardized Device Service pattern

This section introduces the new Standardized Device Service pattern. It is a compound pattern, i.e., it is comprised of combinations of design patterns [12]. The name of the pattern is due to the fact, that realizes device services with standardized service contracts (cf. 3.1 and 4.2) The following patterns are included:

- Service Encapsulation
- Legacy Wrapper
- Dynamical Adapter
- Auto-Publishing

The former two patterns are established SOA design patterns; the latter two patterns could not be found by the authors in the examined pattern literature. These two patterns, as well as the compound pattern, are suggested by the authors as new SOA design patterns. All patterns are described within the next sections.

4.1. Service Encapsulation pattern

The Service Encapsulation pattern is the basis of any SOA and therefore the basis of SODA as well. It addresses the problem of how to make solution logic available as a resource of the enterprise when the associated application was originally not designed to be interoperable [12].

The solution, offered by this pattern, is to encapsulate the solution logic by a service. Applied to devices, the concept of SODA arises.

In order to realize the service encapsulation of devices, three different implementation concepts can be identified in the literature:

- Direct Integration: The device service is offered by the device itself (e.g., by using the Devices Profile for Web Services, as in [21]).
- Adapter Integration: The device is connected to an adapter that offers the device service (e.g., by using a XPORT-Adapter, as in [15])
- Server Integration: The device is connected (maybe by using an adapter) to a server that offers the device service (e.g., by installing device drivers and a web application server, as in [43]).

4.2. Legacy Wrapper pattern

Legacy applications are often automatically wrapped as services when integrating them into a SOA. As a result, functions of the application are directly exposed as capability of the service, using the proprietary data model of the legacy application.

The Legacy Wrapper pattern addresses the problem of negative type of coupling caused by wrapper services. It advocates the establishment of a standardized service contract when wrapping legacy systems as a service (Figure 1) [12]. Thus, the coupling of service logic to the legacy application programming interface (API) is high, but the coupling of service

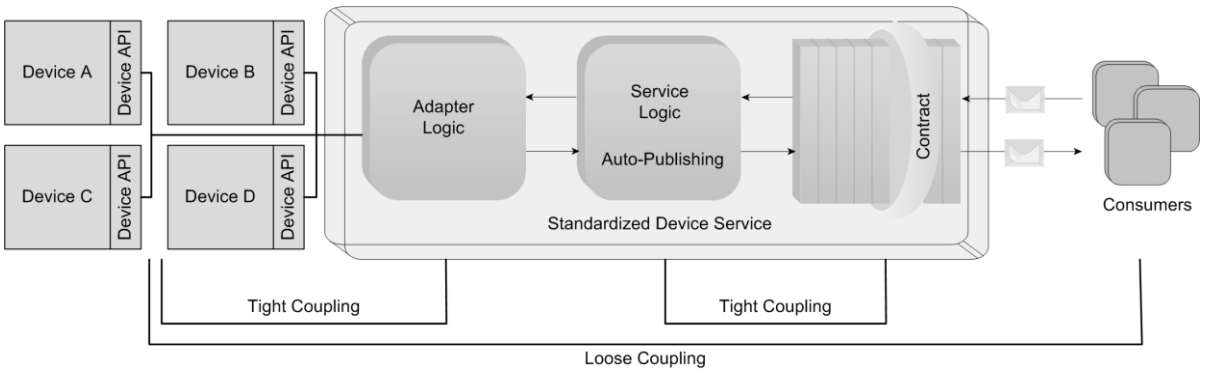

Figure 2: Standardized Device Service pattern

consumers to the implementation is loose. In this way, changes of the legacy API or the replacement of the legacy application affect the service implementation but not the service contract. Ideally, service consumers do not even notice changes behind the service contract.

Medical devices can be perceived as legacy applications. When encapsulating devices by services, the service contract may often be non-standardized, especially if the wrapper service is delivered by the device manufacturer. The application of the Legacy Wrapper pattern results in a standardized device service contract, which can be used for any device of a specific class (e.g., infusion pumps). This enables a loose coupling of consumers to devices. Irrespective of the physical device (its manufacturer, its model and its interface version) service consumers are faced with the same standardized service contract. If devices are updated or replaced, service consumers do not need to adapt their implementations.

4.3. Dynamic Adapter pattern

Wrapper services are usually responsible for encapsulating a specific legacy application in a specific version. Updates of the API or the whole replacement of the legacy application occur rarely and these tasks would be performed in an accurately planned way.

In the context of devices (especially medical devices in hospitals), the replacement or update of devices occurs much more spontaneously, dependent on the current needs (cf. 2.3). If, for instance, a medical device breaks down, the health personnel immediately replaces the device by another one of the same type, but not necessarily of the same manufacturer, the same model or same version. Existing works about SODA only support a static handling of adapters (e.g., in [15]).

The Dynamic Adapter pattern addresses this problem by inserting adapter logic into the device service (cf. 4.5). The adapter logic is responsible for recognizing the model and version of the device and for dynamically selecting an appropriate adapter that is able to communicate with the given device.

Another responsibility of the adapter logic is hiding proprietary device interfaces from the service logic. As a consequence, the addition of a new device adapter can be executed at runtime. The adapter logic only needs to know where the new adapter can be found and which device(s) it is compatible with.

4.4. Auto-Publishing pattern

Software services are usually manually registered in the service registry once. Afterwards, the entry in the service registry does not need to be updated until the service contract is modified or the service is completely shut down. In addition, the unscheduled shutdown of a service is a serious incident and can affect the operability of other services. If, for instance, a customer service were to shut down, all dependent services that need to create, update or find customers would not work properly any more.

Device services behave differently. The shutdown of a device service is a normal task and can happen almost anytime, e.g., when a device is replaced or getting switched off (cf. 2.3). Thus, the operability of a device service directly depends on the operability of the associated device. This aspect can also be found in existing research about SODA (e.g., [22]).

This problem is addressed by the Auto-Publishing pattern which extends the service logic by a mechanism that enables services to automatically publish their service contracts to the service registry. In addition, services automatically unregister themselves, when the device is no longer accessible.

4.5. Architecture

Figure 2 depicts the architecture of standardized device services. They consist of a standardized service contract, service logic and adapter logic.

The standardized service contract is the (for the

service consumer) visible part of the device service. It is independent of proprietary device APIs, and is constructed on the basis of functionalities and data that a device of the specific class offers.

The service logic is the implementation of the service and therefore realizes the service contract. Thus, there is a tight coupling of the service logic to the service contract, which is the only type of positive coupling [9]. In addition, the service logic is responsible for automatically publishing or unpublishing the service to the service registry, depending on the status of the associated device.

The adapter logic is responsible for selecting an appropriate adapter to communicate with the device. In addition, it monitors the status of the device. If the device status changes (plugged on/off, switched on/off, etc.), the adapter logic informs the service logic about the new status for the purpose of auto-publishing/unpublishing. Because the adapter logic is faced with the proprietary device API, there is a tight coupling of the adapter logic to the device.

The most important aspect is the coupling of service consumers to devices. Service consumers are at all times faced with the same standardized service contract, independent of the actual physical device. Thus, service consumers are completely decoupled from the devices.

4.6. Standardized by whom?

When using the term "Standardized", the question arises, by whom the standardization is performed. In the context of SOA, the standardization of services, service contracts or data models is related to the enterprise or at least subdomains (an overall standardization is not realizable in most cases) [9].

In the healthcare domain, several standards exist, like DICOM (Digital Imaging and Communications in Medicine), HL7 (Health Level 7) or ISO/IEEE 11073 [16]. They should be taken into consideration when designing service contracts for device services. This ensures stable and consistent device service contracts.

Within hospitals, different departments have different information needs and the use of specific medical devices depends on the specialization of the department. Thus, a single service inventory for the whole hospital might not be reasonable. Instead, the Domain Inventory pattern should be adopted. It groups services "into manageable, domain-specific service inventories, each of which can be independently standardized, governed and owned." [12].

5. Exemplary scenario

The application of the Standardized Device Service pattern is demonstrated by an exemplary scenario. This scenario was successfully prototypically implemented in our laboratory.

5.1. PCA infusion pumps

In the anesthesia and intensive care unit, infusion pumps are used to "deliver fluid and drugs to the patient in a controlled and easily manageable way" [19]. With PCA (Patient-Controlled Analgesia) infusion pumps, drugs can be delivered in a continuous fluid or in PCA mode. In this mode, the patient can demand a certain amount of drugs, e.g., by pressing a button. For safety reasons, this amount is limited by health personnel. The aim of PCA is "to reduce the dose of analgesic drug to the lowest value acceptable by the patient" [19].

For the purpose of medical documentation, the most interesting data are the medication process and (in PCA mode) especially the PCA behavior of the patient. For this reason, infusion pumps are usually provided with an interface for communication with other systems, e.g., Patient Data Management Systems (PDMS [34]). Normally, these interfaces are proprietary and therefore not standardized.

For the following scenario, B. Braun Perfusor® Space infusion pumps are used in conjunction with a B. Braun SpaceStation and SpaceCom. The SpaceStation acts as a container for several infusion pumps. The SpaceCom component is the communication center of the SpaceStation. Infusion pumps within a SpaceStation can only be accessed over the SpaceCom component by using a proprietary ASCII (American Standard Code for Information Interchange) protocol (BCC Protocol Version V.3.26).

5.2. Scenario

The following scenario is kept simple to demonstrate the concept of standardized device services. Thus, the complexity of intensive care units with lots of devices and monitors is reduced to infusion pumps. In addition, the service contract is limited to manually controlled PCA infusion pumps only. So-called closed-loop control of pumps (i.e., pumps that are controlled by computer applications) are not within the scope of the service contract. This classification is reasonable; due to legal issues (cf. [14]), many infusion pumps are not intended to be controlled over their IT interfaces.

In the scenario, a B. Braun SpaceStation with four infusion pumps is used in conjunction with the B. Braun SpaceCom (Figure 3). Two pumps are actually in use, and the other two pumps are switched off. The device service has automatically published two

services (one for each pump) to the service registry. Service customers can find such device services by sending a search request to the service registry. Afterwards, the service consumer receives all necessary information to use the service.

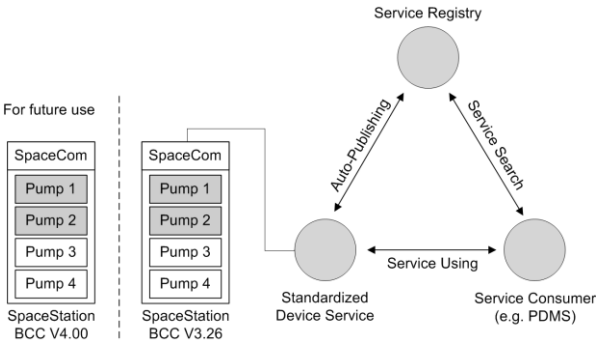

Figure 3: Medical scenario

5.3. Activity 1: on/off-switching of pumps

In addition to the two operating pumps, a third pump is switched on by the health personnel. The adapter logic recognizes the new pump and informs the service logic, which publishes a new service to the service registry. The new device service can now be found and used by service consumers.

After a while, the third pump gets switched off. Again, the adapter logic recognizes the change and informs the service logic, which unpublishes the service. Now, the device service can no longer be found or used by service consumers.

Remark: In fact, the direct communication of e.g., a PDMS to the infusion pump device services is not reasonable. The PDMS would have to take care of new device services for the considered patient and for device services that are no longer available. In this situation, the creation of another service is suggested: one that collects the medication data of all infusion pumps, and one to which specific patients are assigned. For this purpose, we developed a pattern called *Device Concentrator*, which monitors a specific set of device services (e.g., all infusion pumps associated with patient X). The Device Concentrator pattern is out of the scope of this paper and is therefore not included in the scenario.

5.4. Activity 2: device replacing

The SpaceCom with protocol version v3.26 shall be replaced by a SpaceCom with the protocol v4.00 (remark: fictive protocol version), which is not compatible with the old version. For this purpose, the old SpaceCom is completely switched off. The adapter logic recognizes the changes and informs the service logic which unregisters all affected services.

The adapter logic recognizes the new SpaceCom and selects the appropriate adapter (note: at first time, the associated adapter must be made available to the device service. This should be done by the IT department in advance, including functional tests). Subsequently, the service logic publishes services for all operating infusion pumps.

The service consumer does not notice any changes. He is faced with the same standardized service contract as before and does not need to adapt his implementation.

5.5. Service contract

The service contract for the infusion pumps, used in the presented scenario, was defined by using the Web Services Description Language (WSDL). The data model was based on the ISO/IEEE 11073 standard [19] and defined using XML Schema [44]. Details of the contract are out of the scope of this paper. Thus, only the simplified service interface is depicted in Figure 4, using the SOA Modelling Language (SoaML) [35], which is based on UML 2.0 (Unified Modeling Language). This service interface is part of the standardized service contract. Thus, independent of the specific infusion pump (manufacturer, model, version) actually in use, the service interface remains unchanged.

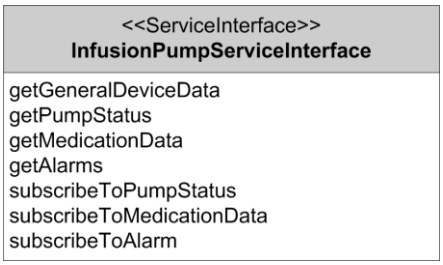

Figure 4: Device service interface

The *getGeneralDeviceData* functionality delivers general information about the device, such as the manufacturer, model version and serial number. Information, which is specific to infusion pumps, can be obtained by *getPumpStatus*, e.g., battery status, readiness for infusion, current power supply, active pumping, etc. Information about the medication, currently infused into the patient, can be gathered with *getMedicationData*. The *getAlarms* functionality delivers all current alarms, such as the battery empty alarm, pressure alarm, end of volume alarm, etc.

Changed information can also be obtained by using the Observer pattern [13], also called Event-Driven Messaging pattern in the context of SOA [12]. Thus,

the service consumer subscribes to a specific event, e.g., a specific alarm, pump status changes or current medication data every five seconds. If the specific event occurs, corresponding information is sent to the service consumer by the device service.

6. Conceptual evaluation

The Standardized Device Service pattern is based on best practice SOA as well as on existing works in the field of SODA. In addition, the specific requirements of device services were taken into account. Furthermore, the technical feasibility could be proved by a prototypical implementation. However, a pattern "provides a proven solution to a common problem" [12]. Thus, the overall evaluation of the pattern needs further research, especially practical experiences.

As a first quality criterion, in the following, we present a reflection of the eight SOA design principles, presented in section 3.1.

Standardized Service Contract. The concept realizes standardized service contracts per definition by applying the Legacy Wrapper pattern (cf. 4.2). Standardization is additional supported by the recommendation of considering existing standards when designing the service contract (cf. 4.6). Thus, services contracts are standardized in any case.

Service Loose Coupling. The concept follows the contract-first approach. Negative types of coupling are avoided by using the Legacy Wrapper pattern (cf 4.5). Thus, there can't be a tight coupling between service consumers and device services.

Service Abstraction. Device services are designed as a black box. Service consumers are not faced with interfaces of physical devices, nor are service consumers aware of implementation details, such as device identification or adapter selection. Thus, the service abstraction is very high.

Service Reusability. Device services are not restricted to specific devices, but are valid for all devices of a specific class. As an example, a device service for infusion pumps can be used for any infusion pump in a hospital, independent of manufacturer, model or version. Thus, the reusability potential is very high.

Service Autonomy. The autonomy of device services depends on the way of implementing the service. If device identification and adapter selection is encapsulated to other services, the autonomy is moderate. If these mechanisms are realized within the service itself, the autonomy is high if no other components can access the stored data.

Service Statelessness. Depending on the device class, scenarios are possible where state information has to be managed by the device service. One possible scenario is the exclusive use of a physical device by a service consumer. In this case, the device service has to store information about the consumer and the usage time / timeout limits. Another example is event managing. If service consumers are enabled to subscribe to specific events, the device has to manage subscriber information.

Service Discoverability. This depends on the technology. If web service technologies are used to realize device services, service discoverability is given by default [10].

Service Composability. Being classified as utility service (cf. 4.7), device services are highly composable if the service contract is designed well.

The results of the reflection are summarized in Table 1. Completely/Predominantly fulfilled means that the application of the pattern enables, e.g., a very high/high degree of service reusability.

Table 1: Standardized Device Service pattern: reflection of SOA design principles

SOA Design Principle	Fulfillment
Standardized Service Contract	●
Service Loose Coupling	●
Service Abstraction	●
Service Reusability	●
Service Autonomy	◕ [1)]
Service Statelessness	◕ [2)]
Service Discoverability	● [3)]
Service Composability	●

[1)] Depends on implementation, here: adapter logic intern, service logic with extern connectivity.
[2)] Depends on device/service functionality, here: only subscriber states are managed.
[3)] Depends on technology, here: web services.
● = completely fulfilled, ◕ = predominantly fulfilled

The table reflects the perception of the authors. The definition of objective criteria for each design principle is extensive work (cf. [37] for the example of loose coupling) for future research and therefore out of scope of this paper. As shown, the concept of standardized device services has the potential to completely fulfill all SOA principles. Depending on the way of implementation and depending on the specific device service functionality, it might be necessary to make concessions to the service autonomy and service statelessness principles.

7. Conclusion and future research

As has been shown in several studies (e.g., [15, 20, 23, 43]), SODA is a promising concept for overcoming

interoperability issues, especially for extending the IT support of processes to devices. In the literature, however, general design concepts are missing. This paper proposes the Standardized Device Service pattern, which is composed of two existing and two newly developed patterns. The concept is based on SOA best practices, as well as on existing works, with respect to the eight SOA principles. By implementing a prototype which realizes the new pattern on the example of infusion pumps, the technical feasibility of the concept is proven.

As a pattern, the presented concept is flexible and reusable (cf. [12]). Thus, it is not limited to specific domains, devices or IT architectures. It can be applied in situations where devices are integrated into a SOA or when devices are encapsulated as services for overcoming interoperability issues. In the healthcare domain, the Standardized Device Service pattern enables the integration of medical devices. Thus, in combination with existing SOA design patterns, the entire IT support of medical processes can be realized, based on SOA best practices.

As mentioned, further research is required. In a next step, several technologies and ways of implementation will be analyzed and compared with each other. In addition, a more complex scenario will be realized to evaluate and improve the concept. If reasonable, further patterns will be developed.

Even if SODA is a promising concept, the management of adapters for devices to enable the service encapsulation, is still an undesirable task. Thus, the efforts of standardization in healthcare should be further expanded. If the potentials of the SODA concept can be transferred and realized in practice, the adoption of an official standard for device services (analogous to ISO/IEEE 11073) would be a desirable consequence.

8. References

[1] C. Alexander, The timeless way of building. New York: Oxford University Press, 1979.

[2] R. Baskerville, M. Cavallari, K. Hjort-Madsen, J. Pries-Heje, M. Sorrentino, and F. Virili, "Extensible Architectures: The Strategic Value of Service Oriented Architecture in Banking," in European Conference on Information Systems (ECIS) Regensburg, 2005.

[3] M. W. Bridges, "SOA in Healthcare," Health Management Technology, vol. 28, pp. 6-10, 2007.

[4] F. Buschmann, R. Meunier, H. Rohnert, P. Sommerlad, and M. Stal, Pattern-Oriented Software Architecture: A System of Patterns. West Sussex: Wiley & Sons, 1996.

[5] D. Cachapa, A. Colombo, M. Feike, and A. Bepperling, "An approach for integrating real and virtual production automation devices applying the service-oriented architecture paradigm," Emerging Technologies & Factory Automation, 2007. ETFA. IEEE Conference on, pp. 309 - 314, 2007.

[6] P. L. Chang, T. M. Wang, S. T. Huang, M. L. Hsieh, K. H. Tsui, and R. H. Lai, "Effects of implementation of 18 clinical pathways on costs and quality of care among patients undergoing urological surgery," The Journal of Urology, vol. 161, pp. 1858-1862, 1999.

[7] S. de Deugd, R. Carroll, K. E. Kelly, B. Millett, and J. Ricker, "SODA: Service Oriented Device Architecture," Pervasive Computing, IEEE, vol. 5, pp. 94-96, 2006.

[8] L. M. S. de Souza, P. Spiess, D. Guinard, M. Köhler, S. Karnouskos, and D. Savio, "SOCRADES: A Web Service based Shop Floor Integration Infrastructure," in The Internet of Things. First International Conference (IOT 2008), 2008.

[9] T. Erl, SOA Principles of Service Design. Boston: Prentice Hall International, 2007.

[10] T. Erl, Web Service Contract Design and Versioning for SOA. Boston: Prentice Hall International, 2008.

[11] T. Erl, "SOA Patterns - Candidate Pattern List," 2009.

[12] T. Erl, SOA Design Patterns. Boston: Prentice Hall International, 2009.

[13] E. Gamma, Design Patterns: Elements of Reusable Object-Oriented Software. Massachusetts: Addison-Wesley Publishing Company, 1995.

[14] A. Gärtner, Medizinproduktesicherheit - Band 1: Medizinproduktegesetzgebung und Regelwerk. Köln: TÜV Media, 2008.

[15] V. Gilart-Iglesias, F. Maciá-Pérez, F. José Mora-Gimeno, and J. V. Berná-Martínez, "Normalization of Industrial Machinery with Embedded Devices and SOA," in Conference on Emerging Technologies and Factory Automation (ETFA '06), 2006.

[16] P. Haas and C. Johner, Praxishandbuch IT im Gesundheitswesen. Erfolgreich einführen, entwickeln, anwenden und betreiben München: Carl Hanser Verlag, 2009.

[17] R. Haux, A. Winter, E. Ammenwerth, and B. Brigl, Strategic Information Management in Hospitals. New York: Springer-Verlag, 2004.

[18] A. R. Hevner, S. T. March, J. Park, and S. Ram, "Design Science in Information Systems Research," MIS Quarterly, vol. 28, pp. 75-105, 2004.

[19] ISO/IEEE, ISO/IEEE 11073-10101:2004: Health informatics - Point-of-care medical device communication - Part 10101: Nomenclature, 2004.

[20] F. Jammes, H. Smit, C. Arandyelovitch, and F. Depeisses, "Intelligent device networking in industrial automation," in 2nd IEEE International Conference on Industrial Informatics (INDIN '04), Berlin, 2004, pp. 449-456.

[21] F. Jammes and H. Smit, "Service-Oriented Paradigms in Industrial Automation," IEEE TRANSACTIONS ON INDUSTRIAL INFORMATICS, vol. 1, pp. 62-70, 2005.

[22] F. Jammes, H. Smit, J. L. M. Lastra, and I. M. Delamer, "Orchestration of service-oriented manufacturing processes," in 10th IEEE Conference on Emerging Technologies and Factory Automation (ETFA 2005), 2005, pp. 617-624.

[23] S. Karnouskos, O. Baecker, L. M. S. de Souza, and P. Spiess, "Integration of SOA-ready Networked Embedded Devices in Enterprise Systems via a Cross-Layered Web Service Infrastructure," in 12th IEEE Conference on Emerging Technologies and Factory Automation (ETFA 2007), Patra, Greece, 2007.

[24] D. Krafzik, K. Banke, and D. Slama, Enterprise SOA - Service-Oriented Architecture Best Practices. Indiana, USA: Pearson Education, 2006.

[25] H. Krcmar, Informationsmanagement: Springer Berlin Heidelberg New York, 2005.

[26] J. M. Leimeister, A. Schweiger, and H. Krcmar, "Ortsunabhängiges Management von hochpreisigen mobilen medizinischen Geräten im Krankenhaus auf WLAN-Basis," in Proceedings of Informatik 2006, GI - Gesellschaft für Informatik, Ed. Dresden, 2006, pp. 220-226.

[27] K. Lesh, S. Weininger, J. M. Goldman, B. Wilson, and G. Himes, "Medical Device Interoperability – Assessing the Environment," in Joint Workshop on High Confidence Medical Devices, Software, and Systems and Medical Device Plug-and-Play Interoperability, 2007.

[28] E. Liljegren, "Usability in a medical technology context assessment of methods for usability evaluation of medical equipment," International Journal of Industrial Ergonomics, vol. 36, pp. 345-352, 2006.

[29] S.-C. Liu, W.-T. Chang, C.-H. Huang, T.-I. Weng, H.-M. Ma Matthew, and W.-J. Chen, "The value of portable ultrasound for evaluation of cardiomegaly patients presenting at the emergency department," Resuscitation, vol. 64, pp. 327-331, 2005.

[30] C. Mauro, A. Sunyaev, J. M. Leimeister, and H. Krcmar, "Service-orientierte Integration medizinischer Geräte - eine State of the Art Analyse," in Wirtschaftsinformatik 2009 - Business Services: Konzepte, Technologien und Anwendungen Wien, 2009, pp. 119-128.

[31] J. McCauley and F. D. Joseph, "Maintenance and Repair of Medical Devices," in Clinical Engineering Handbook Burlington: Academic Press, 2004, pp. 130-132.

[32] J. P. Melrose, "e-health is the way via SOA," Healthcare Financial Management, vol. 61, pp. 120-122, 2007.

[33] J. C. Naranjo, C. Fernandez, S. Pomes, and B. Valdivieso, " Care-Paths: Searching the way to implement pathways," in Computers in Cardiology Valencia, 2006, pp. 285-288.

[34] S. P. Nelwan, T. B. van Dam, S. H. Meij, and N. H. J. van der Putten, "Implementation and use of a patient data management system in the intensive care unit: A two-year experience," in Computers in Cardiology, 2007, 2007, pp. 221-224.

[35] OMG, "Service oriented architecture Modeling Language (SoaML) - Specification for the UML Profile and Metamodel for Services (UPMS)," 2008.

[36] M. P. Papazoglou, P. Traverso, S. Dustdar, and F. Leymann, "Service-Oriented Computing: State of the Art and Research Challenges," Computer, vol. 40, pp. 38-45, 2007.

[37] C. Pautasso and E. Wilde, "Why is the web loosely coupled?: a multi-faceted metric for service design," in Proceedings of the 18th international conference on World wide web Madrid, Spain: ACM, 2009.

[38] S. Pöhlsen, S. Schlichting, M. Strähle, F. Franz, and C. Werner, "A Concept for a Medical Device Plug-and-Play Architecture based on Web Services," in 2nd Joint Workshop on High-Confidence Medical Devices, Software and Systems (HCMDSS) and Medical Device Plug-and-Play Interoperability (MD PnP) San Francisco, 2009, pp. 52-65.

[39] M. I. Reynolds, "Device Interfaces," in Anesthesia Informatics, J. Stonemetz and K. Ruskin, Eds. Berlin: Springer Verlag, 2008, pp. 109-146.

[40] M. Schumacher, Security engineering with patterns: origins, theoretical models, and new applications. Berlin: Springer, 2003.

[41] A. Schweiger, A. Sunyaev, J. M. Leimeister, and H. Krcmar, "Toward Seamless Healthcare with Software Agents," Communications of the Association for Information Systems (CAIS), vol. 19, pp. 692-709, 2007.

[42] M. Stal, "Using architectural patterns and blueprints for service-oriented architecture," Software, IEEE, vol. 23, pp. 54-61, 2006.

[43] M. Strähle, M. Ehlbeck, V. Prapavat, K. Kück, F. Franz, and J.-U. Meyer, "Towards a Service-Oriented Architecture for Interconnecting Medical Devices and Applications," in Joint Workshop on High Confidence Medical Devices, Software, and Systems and Medical Device Plug-and-Play Interoperability, 2007.

[44] W3C, "XML Schema Part 0: Primer Second Edition - W3C Recommendation," 2004.