Secured SOA for the Safe Interconnection of Medical Devices (Position Paper)

Martin Leucker[1]
leucker@isp.uni-luebeck.de

Malte Schmitz[1,2]
mschmitz@isp.uni-luebeck.de

[1]Institute for Software Engineering and Programming Languages, Universität zu Lübeck
[2]Graduate School for Computing in Medicine and Life Sciences, Universität zu Lübeck

Abstract

This position paper presents a concept on how the dynamic interconnection of medical devices in a service oriented architecture (SOA) can be secured based on formal interface descriptions. In the framework OpenSDC used in the research project OR.NET formal interface descriptions are already used to model medical devices. The new approach is to use this formal descriptions to ensure the correct usage of the interface at runtime and to support developers in implementing medical devices with OpenSDC at the same time.

1 Introduction

As medical devices are safety-critical systems, a risk analysis has to be carried out that comprehensively investigates the hazards to which a machine can expose the patient during surgery. To this end, the interfaces of a machine both to humans but also to other medical devices have to be precisely defined and measures for any (intentional and unintentional) misuse of the interface have to be taken. In this position paper we suggest the following approach to ensure the correct employment of a machine's interface. Starting from a formal definition of the interface together with constraints about its correct usage, we suggest to (i) generate a Java interface skeleton from the formal specification to simplify the implementation and thus the programmer's work, (ii) generate monitors to ensure the correct usage of the interface and (iii) generate possible mediators whose goal is to translate between different interface types for which a well-known translation scheme is available. We are convinced that the presented approach allows the misuse of a given interface to be detected while at the same time offering great flexibility whenever a meaningful connection of devices is suitable. Using this approach the approval process for the overall medical device will be simplified significantly, as a prove that identified risk measures are respected in the implementation is more evident if formal specifications are used to ensure the correct usage of the interface at runtime.

We outline our ideas within the communication framework OpenSDC [GBF12] used in the project OR.NET[1]as reference implementation of the Open Surgical Communication Protocol (OSCP). OR.NET addresses the legal and technical challenges of using interconnected medical devices in an operating room (OR) [BB12]. OpenSDC is a Java library going to be standardized for the communication of interconnected medical devices. To achieve a flexible, dynamic interconnection of devices a service oriented architecture (SOA) is used. OpenSDC can be used to implement providers (medical devices) and consumers (medical clients) for this SOA.

An earlier and more abstract version of this concept was given in [KL13] together with an overview of the legal situation. In this paper we make those abstract ideas more precise.

2 Medical Device/Client Description

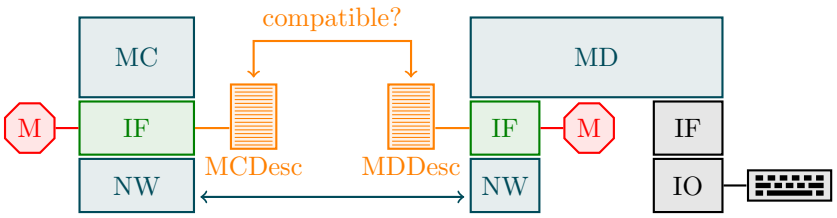

Figure 1: Simplified layered model of the interconnection. A medical device (MD) is connected to its peripheral IO, e. g. a keyboard, through an interface (IF). The same way the MD and the medical client (MC) are connected through an IF with the network stack (NW), but these interfaces can be generated from the medical device description (MDDesc) or the medical client description (MCDesc) respectively. The monitors (M) can be generated from the same source.

The medical device information base (MDIB) is an important component of the architecture used in OR.NET. The MDIB is a hierarchical tree-like structure that stores medical objects. On the lowest layer it describes metrics of a medical device that can be read and set. The concept of an MDIB is described in ISO/IEEE 11073 [x7304] as part of the domain information model for medical devices.

In OR.NET the MDIB consist of two parts: The medical device description (MDDesc) and the medical device state (MDState). The MDState contains current values of metrics and the MDDesc provides meta data on these values. These meta data contain information about the physical quantity and the unit of a metric. Only using both one can interpret the received values correctly. The units and physical quantities in the MDDesc are described with code identifiers that are defined in Part 10101: Nomenclature of [x7304]. In the OpenSDC implementation the MDDesc can be imported from an XML serialization whose format is going to be standardized in the OR.NET context.

In the current implementation the client can retrieve the MDDesc from a device and use this meta data to read and set values in the MDState of the device. In other words the client can retrieve a formal interface description from the server. To extend this idea we propose a medical client description (MCDesc) that contains such formal meta data on the interface of a client. This includes a complete list of metrics set and read by this client and the physical quantities and units of these metrics.

Given an MDDesc and an MCDesc both serialized as XML one can check the compatibility of the interfaces of two medical devices. This way an interconnection can only be allowed if the interfaces of device and client are compatible. At the moment the MDDesc and MCDesc interface description is stateless, which makes such a check quite trivial, but limits the expressiveness of these descriptions, too. Consider an instrument that can be in an ON and an OFF state. While in the ON state the surgeon can apply too much force and hence the device transitions to the OVERLOAD state. Once in the OVERLOAD state the surgeon has to turn the device OFF before it is allowed to turn it back ON. To be able to consider such conditions in the compatibility check, we want to go one step further and add stateful interface descriptions to the MDDesc and the MCDesc. This could be achieved using interface automata as proposed in [dAH01]. The authors already present a ready-to-use compatibility check algorithm.

3 Interface and Monitor Generation

As part of the risk control measures each medical device has to perform a plausibility check on every incoming parameter [JWHK11]. This holds for input coming from physical input devices connected directly to the medical device as well as for input coming over the network. In figure 1 the first case is depicted in the rightmost stack with a keyboard connected to the IO layer. The plausibility check is implemented manually in the interface layer and many tests were written to assure the correct implementation of the check.

In the case of interconnected medical devices such a plausibility check is even more important, but in the current implementation satisfying its own interface description and checking the correctness of incoming messages is completely left up to the user code. We therefore propose to automatically generate the interfaces from the already existing formal interface description in the MDDesc or the new MCDesc. We want to generate some

[1] `www.ornet.org`

Java code doing the hard work and some Java interfaces that the user software has to implement[2]. This way technical details like handles of the metrics and most of the complexity of the highly parallel communication model can be hidden from the user software. This idea is depicted in the other stacks in figure 1 with the link between the document depicting the MDDesc and the interface layer.

In addition to the generated Java interface we want to generate monitors from the MDDesc that assert at runtime that conditions of the interface are satisfied. Of course conditions that are already enforced by the Java interface are not monitored any more, but a Java interface is not able to express more complex conditions, such as allowed ranges of integer values. In the case of stateful interface descriptions the generated monitors could check these conditions, too. Using such monitors provides us with two benefits: We can be sure that incoming values are always in the range of allowed values and one can trust the compatibility check described above. The result of the compatibility check is useless without the guarantee that every communication partner follows its formal interface description at runtime. Using the generated monitors on all medical devices in the interconnection, we can use the compatibility check as a strong argument in risk control.

4 Mediator Synthesis

Last but not least the MDDesc/MCDesc pair can be used to make an interconnection possible in some cases even if the compatibility check fails. Assume we stored some relations on physical quantities and units in an ontology. This knowledge can be used to synthesize a mediator if an MCDesc expects other conditions than the MDDesc provides. For a technique for automated synthesis of mediators compare [BCIJ13]. Such a mediator would be a pure software device consisting of a virtual client and a virtual server. The virtual server provides the device that the MCDesc requests and translates every request to fulfill the requirements of the MDDesc. A very simple example would be a mediator translating Celsius to Fahrenheit. More complex examples could involve stateful transitions like summing up values over time or keeping track of the last values to compute deltas.

5 Summary

With the XML serialization of the MDDesc a formal interface description of medical devices exists, is used in the research project OR.NET and is going to be standardized. Using and extending this formal interface description we can generate Java interface skeletons to hide the complexity of OpenSDC from developers of medical devices and generate monitors ensuring the correct usage of the interface at the same time. Using such a technology the development of and risk analysis for an interconnected medical device will be much easier.

The next steps include implementing a tool to simplify the modeling of the MDDesc as well as implementing the generation of the Java skeleton and the monitor from the MDDesc. To implement mediator synthesis and the proposed compatibility check we need to define the proposed MCDesc in more detail.

Acknowledgements

This work is supported in part by the BMBF project OR.NET under number 16KT1231.

References

[BB12] M. Birkle and B. Bergh. OR.NET: Ein Projekt auf dem Weg zur sicheren dynamischen Vernetzung in OP und Klinik. In *Informatik 2012*, volume 208, pages 1235–1236. GI, 2012.

[BCIJ13] A. Bennaceur, C. Chilton, A. Isberner, and B. Jonsson. Automated Mediator Synthesis: Combining Behavioural and Ontological Reasoning. In *SEFM*, volume 8137 of *LNCS*, pages 274–288. Springer, 2013.

[dAH01] L. de Alfaro and T. Henzinger. Interface automata. In *SIGSOFT*, pages 109–120. ACM, 2001.

[GBF12] D. Gregorczyk, T. Bußhaus, and S. Fischer. A Proof of Concept for Medical Device Integration using Web Services. In *SDD*, pages 1–6. IEEE, 2012.

[JWHK11] C. Johner, S. Wittorf, and M. Hölzer-Klüpfel. *Basiswissen Medizinische Software*. dpunkt.verlag, 2011.

[KL13] F. Kühn and M. Leucker. OR.NET: Safe Interconnection of Medical Devices (Position Paper). In *FHIES*, volume 8315, pages 188–198. Springer, 2013.

[x7304] ISO/IEEE 11073: Health informatics – Point-of-care medical device communication, 2004.

[2] As all parts of OpenSDC are implemented in it, Java is the natural choice when implementing communication based on OpenSDC.