Towards Self-Organization of Networked Medical Devices

Andreas Kliem[†], Joachim Hänsel[*], Matthias Hovestadt[†], Michael John[*], Odej Kao[†]

[†]Technische Universität Berlin
Email: {firstname}.{lastname}@tu-berlin.de

[*]Fraunhofer Institute for Computer Architecture and Software Technology
Email: {firstname}.{lastname}@first.fraunhofer.de

***Abstract*—Soaring costs and an aging society originate the need for novel technologies in health care sector. Especially telemedicine and AAL systems are often promoted as such. Enabling patients to stay in their domestic environment while reducing costs due to hospital visits seems to be an evaluable opportunity on the one hand but introduces a lot of challenges, like interoperability and security, on the other. Interoperability is the main challenge to put emphasis on, when thinking of networks of medical devices recording patients vital data while acting in a plug-and-play manner. Thus, standards are needed that enable manufacturers to build interoperable medical devices and provide solutions for cost effective and easy to use telemedicine systems. Therefore this paper analyzes the ISO/IEEE11073 PHD standard towards its applicability in such environments, where the focus lies on self-organization and plug-and-play functionality. Furthermore we want to present research done on telemedicine/AAL systems during the SmartSenior project as well as introduce our own ISO/IEEE11073 implementation, which served as a basis for our analysis.**

***Keywords*-11073, ubiquitous-Health, telehealth, telemedicine, DPWS, self-organization**

I. Introduction

Nowadays, an increasingly aging society is one of the most important challenges to deal with in the health care sector. Two of the major arising problems are significantly increasing costs as well as patients suffering from reduced quality of life and independence [1]. A solution or at least mitigation may be found in using systems that offer ambient assisted living (AAL) or telemonitoring/telemedicine capabilities, helping to transform the current hospital-centric health care system to a more patient-centered one. Therefore researchers from both academia and industry focused to mobile/personal healthcare applications, also known as ubiquitous-Health (u-Health), enabling patients to stay in their domestic environment without being cut from medical attendance. Project examples are SmartSenior, EDiMed, E-Health@Home, JUTTA, MeDiNa, MATRIX, ALARM, AMICA, WohnSelbst, and MAS.

Building a system not only accepted by the provider but particularly also the patient heavily depends on the availability and simplicity of medical devices capturing patient's vital data (glucose meters, blood pressure and heart rate meters, weighing scales, etc.). Furthermore, basic challenges of u-Health systems are:

- *Heterogeneity:* monitoring/assistance of patients in home/ubiquitous environments enforces the need for many kinds of medical devices that produce different kinds of data. Medical devices should be easily replaceable.
- *Interoperability:* data from medical devices often needs to be analyzed/aggregated, which demands common communication standards. Medical devices should be reusable at different locations as well as expandable in their functionality.
- *Simplicity:* deployment of a Home Area Network (HAN) or Body Area Network (BAN) has to be as easy as possible in order to reduce costs generated by service technicians and to avoid disclaim of patients. Medical devices should be able to enter/leave a HAN or BAN dynamically.
- *Security:* medical data collected in u-Health environments have to pass several networks and systems. A common security policy is required to guarantee at least the same security standards as found in clinical/hospital environments.
- *Safety:* u-Health systems need to operate properly since the well-being of patients may be compromised. Thus testing of all involved components becomes a crucial part of ensuring adequate behaviour of the overall system.

Medical device manufacturers often use proprietary software and communication protocols, which leads to proprietary solutions only working together with devices from the same vendor. However, limiting a u-Health system to products from one vendor can lead to a dependent relationship and less functionality because the demand for various medical devices needed to monitor different kinds of patients often can not be satisfied by one manufacturer. Moreover using one proprietary solution raises difficulties in exchanging devices and potentially generates higher costs.

This lack of interoperability, which can be seen as a serious obstacle for building suitable u-Health solutions, highlights the need for open systems and standards that offer common communication formats in heterogeneous health care systems. Although a numerous of standards are available,

their dissemination particularly corresponding to small and mobile medical devices is marginal. Among standards like HL7 [2] or DICOM [3] rather targeting interoperability for large scale health care systems, especially the ISO/IEEE11073 family of standards gained attention in industry and research. A subset of these standards [4] addresses the demand for simplified and optimized medical device communication in u-Health as well as in personal health environments and therefore seems to be an promising solution. As mentioned earlier, the availability of many kinds of interoperable medical devices is crucial for establishing an u-Health system but manufacturers tend to use their own proprietary solutions. However, in case of the ISO/IEEE11073 standard this trend seems to be just changing, promoted by the release of the Bluetooth Health Device Profile (HDP) [5] and the foundation of the Continua Health Alliance [6]. HDP enables manufacturers to build interoperable wireless devices based on ISO/IEEE11073 whereas the Continua Health Alliance, an open consortium of healthcare and technology companies, works towards propagating and establishing interoperable u-Health systems based on ISO/IEEE11073.

In this paper we want to evaluate ISO/IEEE11073 towards its applicability for self-organizing u-Health systems, and present the problems we found as well as motivate possible solutions. Section II will briefly introduce the ISO/IEEE11073 standard. Motivated by our findings during the implementation, which is described in the appendix, Section III discusses the standard's capabilities towards a self-organizing u-Health environment. In Section IV we introduce the SmartSenior project whose main goal is to develop AAL and telemonitoring/telemedicine systems in order to advance the quality of life of aging people. Section V describes our model based ISO/IEEE11073 compliance test environment, which eases to gain Continua certification for their own ISO/IEEE11073 devices and software. Finally, Section VI summarizes future research tasks based on our analysis and Section VII presents research related to ISO/IEEE11073.

II. Introducing ISO/IEEE11073

The development of the ISO/IEEE11073 family of standards started in 1982 due to demand and benefits that plug-and-play would provide to medical devices. In the context of ISO/IEEE11073 medical devices are called agents and devices communicating with them (such as smartphones) are called managers. With home and wearable healthcare systems gaining more attention, work was done on investigating [7] and improving ISO/IEEE11073 towards the usage in such environments, resulting in a set of sub-standards, where the most important are:

- *11073-20601:* This part defines a Domain Information Model (DIM), which describes several classes and attributes to model a medical device in an object oriented way, a service model, which defines basic mechanisms to exchange data, and a communication model. Moreover ASN.1 based data types and Medical Device Encoding Rules (MDER) are introduced to provide a canonical data representation. Additionally, a finite state machine is defined for both agent and manager, which constitutes the association process between agent and manager as well as the states that allow for medical data exchange.
- *11073-104xx:* These parts define medical device specializations like blood pressure monitors, weighing scales or glucose meters. Generally, each of them uses a subset of objects, attributes and services defined in ISO/IEEE11073-20601.

Agents in context of ISO/IEEE11073-20601 are defined by an object-oriented model. The basic implementation of the ISO/IEEE11073-10415 weighing scale device specialization for instance, consists of a *MDS* and a *Numeric* object, as shown in Figure 1. The top level *MDS* object defines status (e.g. power-status or time information) and identification (e.g. system-model or system type) properties of an agent and has to be realized by every agent. The *Numeric* object is a subtype of the *Metric* object, which is the base class for all objects representing measurements. In case of the weighing scale the *Numeric* objects attributes include unit codes, the actual measured weight and some more context information. Measured data is accessed by managers through the service model. One possibility is to use the *Data Request Service*, which triggers the agent to observe the *Numeric* object for changes (new measurements) and send them to the manager by using ASN.1 encoded messages, which are defined by the standard too. In general, the communication between agents and managers can be divided into three main phases:

- *Association phase:* Agent and manager negotiate a data protocol, which specifies the encoding to be used as well as a set of operating parameters (e.g. which kinds of the Data Request service are supported). This phase is always triggered by the agent.
- *Configuration phase:* After the association phase has completed successfully and the default configuration announced by the agent is not already known by the manager, the configuration phase is started. A configuration is the set of objects and corresponding static attributes used by the agent to describe itself. By exchanging this configuration, the manager is able to create a local copy of the agents object-oriented model.
- *Operating phase:* If both agent and manager have agreed on a common data protocol and configuration, they enter the operating state and are allowed to exchange data by using services described by the service model. Examples are the *Get Service*, that allows to request attributes stored in the MDS object, or the already mentioned *Data Request Service*. In general, all services are used to access dynamic attributes (attributes that can change during an association between agent and manager and are not included in the configuration) and to keep the managers copy of the agent object-oriented model up-to-date.

In order to have an open platform for our ongoing research in the area of u-Health, we decided to develop our own im-

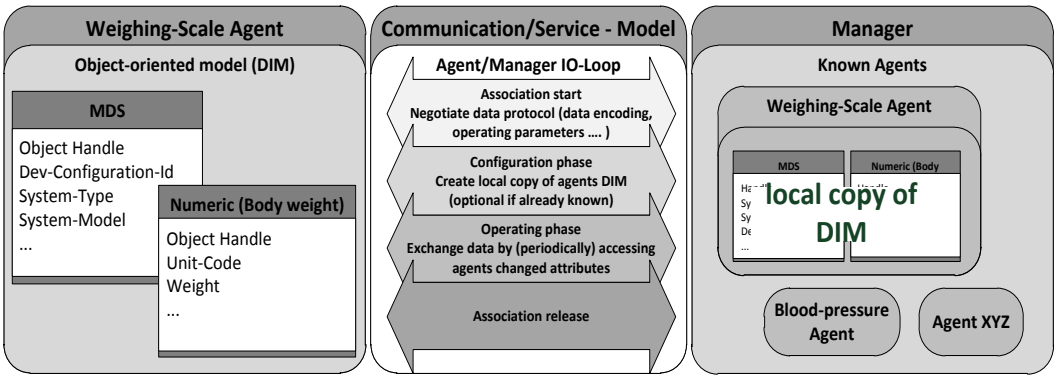

Fig. 1: ISO/IEEE11073 agents are represented by an object-oriented model, also called Domain Information Model(DIM). Measurements are exchanged by accessing objects and attributes of an agent through services defined in the service model. The communication model specifies states (associated, configuring, operating), that allow different services to be called (e.g. measurement data transfer is only allowed during the operating state).

plementation of ISO/IEEE11073-20601 and its device specializations, which is introduced in the appendix. The following section discusses the applicability of ISO/IEEE11073 towards self-organizing and dynamic u-Health systems, based on our findings during the implementation.

III. Analyzing ISO/IEEE11073 towards self-organizing u-Health systems

Thinking of self-organization in u-Health systems, Plug-and-Play (PnP) functionality comes into focus. As per definition, one could also imagine networks of medical devices reacting to events (i.e. critical blood pressure recorded) by dynamically initiating appropriate actions, which could be described as a system of independent devices spawning new functionality through self-organization. However, as ISO/IEEE11073-20601 is meant to be an "optimized exchange protocol" for personal health device communication and interoperability, covering PnP seems to be the first step. Furthermore, realizing PnP is crucial for u-Health to minimize costs originated due to technicians and frustrated patients.

In our analysis, we found some serious obstacles for self-organizing systems, currently not covered by the standard, which we want to discuss in detail.

A. Agent initiated associations

The finite state machine, defined by ISO/IEEE11073, manages the process of starting an association to the agent. When we think of Bluetooth/HDP or other Near Field Communication (NFC) enabled medical devices automatically connecting to a patient's manager when it comes into range, this seems to be a good practice. But examined from a more service orientated point of view, some problems arise. Looking at a network of medical devices connected over IP (i.e. Wireless LAN), it is unclear how the managers are discovered. Moreover, embedded devices normally must operate with low energy, which is in contrast to polling periodically for connected managers. Therefore we suggest that agents, as the service providers, wait for managers discovering and connecting to them. Our implementation solved this problem, by remaining in the *Disconnected* state, until a connection between manager and agent was established, regardless of whether the agent is or is not connected to a W-LAN for instance. Another problem directly associated to this one, are scenarios where we have several managers and agents connected to a network.

B. Multiple Agent/Manager scenario

Having multiple managers and one agent connected to a network, the standard does not define how the manager is chosen by the agent, which is starting the association. Furthermore, choosing the wrong manager could result in security issues. However, ISO/IEEE11073 knows the concept of unique agent and manager IDs, but does not define any realization towards a secure pairing of both, which is in conflict with the major goal of interoperability. In general, this problem can be formulated as an agent to manager pairing problem, which also takes place in case of HDP(Bluetooth) communication. Imagine two patients living together, both having a manager (e.g. their smartphone) and both sharing one blood pressure monitor. First, the blood pressure monitor does not know which patient wants to use it and second, if the monitor successfully established a connection to a patient's manager, a request for measurement transfer was started and the connection suddenly disrupts, it cannot be guaranteed that the blood pressure monitor will reconnect to the same manager. This could cause measurements transfered to the wrong patient.

C. Mobile agents changing their location

Allowing agents to change their location might cause situations where two or more agents of the same type become available to a manager (e.g. several patients with active blood pressure monitors waiting together in a medical practice). Similar to the above obstacle, wrong measurement data could be transferred. Moreover managers could connect to agents not belonging to them and could override or influence parameters

of ongoing measurements or even the whole device configuration.

D. Agent-Patient mapping

ISO/IEEE11073 brings the possibility to record medical data related to patient IDs but does not define how this mapping has to be realized by medical device manufacturers. Moreover, the standard allows agents to record data while they are not connected to a manager as well as it allows managers to request all stored data. This can result in situations, where managers are not able to confirm which measurement belongs to them. Furthermore security issues appear, if managers are able to access all stored data on an agent.

E. Security specification

Building self-organizing u-Health systems, medical devices and managers need to enter and leave HANs or BANs dynamically. Therefore a specification for defining which device can enter which network and which agent is accessible by which manager is needed. Furthermore such a specification should enable restriction of managers trying to access medical data not belonging to them. It is also necessary to consider, whether relying on transport layer security (e.g. Wireless LAN Encryption) is sufficient or whether application layer security has to be introduced in addition. As the ISO/IEEE11073-20601 standard primarily focuses on data exchange to enable interoperability, security issues are currently not covered, even though they play an important role related to the already mentioned problems too.

F. Results

In order to overcome these problems, current u-Health architectures based on ISO/IEEE11073-20601 (as used in the SmartSenior project for instance) need to realize networks, where the agents are tightly coupled to one manager (i.e. one patient). This results in higher costs, because every new/replaced agent has to be set up by technicians in order to operate in his network and is bound to one patient, which seems to be inefficient when thinking of several patients living in the same home.

To realize cost-effective and honorated u-Health systems, PnP functionality is needed. Medical devices (agents) as well as managers must be able to dynamically enter or leave u-Health networks. Moreover medical devices should be able to operate independently from their location and offer their services to multiple managers. Managers may have the possibility to dynamically discover agents and must be restricted towards their agent/medical data access. As presented, the standard does not fulfil these requirements, because it is intended to be used as a data exchange protocol only. Therefore an overlay or extension is required, which keeps interoperability while addressing the mentioned issues.

IV. Ubiquitous Health in SmartSenior

SmartSenior is a publicly BMBF (Federal Ministry of Education and Research, Germany) founded project developing AAL and telemonitoring/telemedicine systems to advance quality of life of aging people by moving parts of their medical assistance to their domestic environment. While the project focuses on many parts of telemedical solutions (e.g. emergency management, patient monitoring), which have a high degree of complexity due to cross system interaction, we want to demonstrate the complexity of telemedical applications by introducing a SmartSenior Use Case directly related to ISO/IEEE11073 and the discovered problems related to self-organization.

As shown in Figure 2 the Use Case covers accommodation and mobility aspects of a patient. The general system layout consists of several medical devices, which are either locally bound (W-LAN sensors for indoor usage) or mobile (Bluetooth sensors for in - and outdoor usage), and a smartphone which acts as the manager in the context of ISO/IEEE11073. Furthermore a backend server is used, which stores patient's profiles (e.g. devices and patient ID, vital parameters to be monitored, measuring times) and communicates with the smartphone in order to push profile updates or retrieve vital data. The starting point for interaction with medical devices is a clinic or medical practice, where a doctor initially creates or updates a patient's profile. By entering a global unique patient ID into the patient's smartphone, the profile is downloaded or automatically updated respectively and a search process for fitting medical devices is triggered. Here, two challenges for self-organization can be found. The first one targets the issues presented in Section III. When devices are dynamically discovered independent from their location, rules are needed that define which device is accessible, allow simple pairing and avoid transmission of invalid data (e.g. from a different patient). The second challenge covers the system's behaviour due to different locations. While entering the home the localization information initiates a handover process of networks (UMTS to WLAN). The smartphone smoothly connects itself to the home area network and establishes a continuity of services (e.g. streaming of vital data to the clinic).

The presented use case relies on self-organization capabilities that are currently under development (especially the ones regarding ISO/IEEE11073). That is why the current field test architecture, as described in Section III-F, differs from the presented one in order to provide a stable system with the limited capabilities of ISO/IEEE11073. However, it is planned to realize the Use Case under laboratory conditions.

The first challenge to solve is to implement a reliable discovery mechanism, which is briefly discussed in the description of our own ISO/IEEE11073 implementation in the appendix. The next step is to define how agents can enter and leave networks. Because agents normally have very restricted resources, we promote to realize this functionality within a protocol between managers and the backend, which is extended by an agent directory service. As ISO/IEEE11073 defines a globally unique ID for each agent, it is easy to identify them in the directory. Beside localization and other context information the directory stores a state for each agent that can be one of:

- *Unknown:* the agent is unknown and has to be first

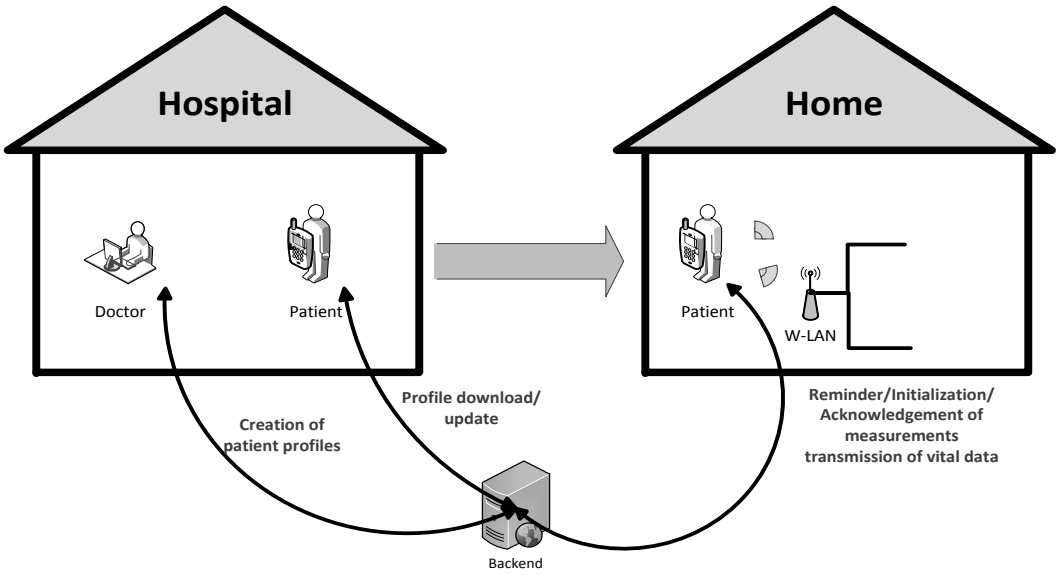

Fig. 2: A SmartSenior use case presenting the accommodation of a patient as well as mobility aspects. Recording of vital data is controlled by a profile that configures which medical devices are needed and how often measurements are taken.

discovered by a manager in order to get registered at the directory

- *Known:* the agent is registered at the directory, its location is known and it can be used by managers
- *Connected:* in addition to the state *Known*, the agent is in use by a manager
- *Lost:* the agent is registered at the directory but its location is unknown (i.e. it was not discovered by managers at its expected location for an amount of time)

The state of an agent defines how managers have to interact with them (e.g. *Unknown* agents must be registered before usage, *Known* agents have to be marked as *Connected* before usage, etc.). This approach requires managers to keep the information stored in the dictionary up to date and relies heavily on the discovery feature and frequent communication with the backend service. Moreover, security issues are not covered at the moment. However, we believe that this model can act as a basis for our ongoing research.

Because of the complexity of such systems and the need for reliable behaviour, we also focused on test environments in SmartSenior, which we will discuss in the next section.

V. 11073 Compliance Test Environment

As already pointed out in the introductory section, medical systems in general and u-health systems in particular need to behave properly and reliably. This is due to the high quality expectations resulting from the systems potential in affecting patients well-being. In the same section the importance of the establishment of standards for inter-device communication has been introduced and the ISO/IEEE11073 family of standards was motivated as a suitable protocol for vital data monitoring.

While a standard forms the basis for reliable communication, its implementation is rarely free of errors from the beginning. One of the reasons for this is the usually complex behaviour defined by communication standards like ISO/IEEE11073. Obviously thorough testing is necessary to gain enough confidence in proper behaviour of the software implementing the standard. To achieve this, large test suites need to be written manually. However, although manually written tests help raising the confidence in the system, it is often a tedious, time consuming and therefore costly task to create them.

Automating the test design process by the application of model based testing is a common way to approach this issue. Consequently we adopted model based testing to create a test suite which was able to cover the behaviour specified by the ISO/IEEE11073 standard to a high degree.

Initially we realized an UML state machine model as described in the ISO/IEEE11073-20601 communication model (Part 8 and Annex E). Furthermore we analyzed all outgoing and incoming messages to the state machines in order to model an appropriate test interface. Based on the interface definition we implemented a test adapter. We decided to run the first tests on ISO/IEEE11073 devices that are communicating over tcp/ip to ease the realization of the test adapter. With the state machines and the test adapter we finally had a sufficient setup to deploy a test automation tool which could generate test cases from the model which would run on the test adapter.

We used the Conformiq Test Designer tool for automating the test design. We chose to activate all available coverage criteria for the generation process to get a sound test suite. The tool generates abstract test cases which need to be transformed into concrete test cases which can then be run against the test adapter. Therefore we implemented this transformation by a so called scripting backend which we integrated into the generation process provided by the tool. Eventually we

realized the whole model based testing process and created in an automated way a JUnit test suite that covered a large amount of the ISO/IEEE11073-20601 communication model behaviour.

With the models, scripting backend and test adapter we created a test environment as a basis for all the further test activities such as generating more tests with a combination of the ISO/IEEE11073 test models and models that describe additional behaviour of the system.

VI. Future Work

Our main focus tends towards realizing ISO/IEEE11073 extensions to address the problems described in Section III. Therefore we are currently working on an overlay discovery protocol, which enables managers to discover ISO/IEEE11073 compliant devices and provides authorization/security features. The primary requirements we concentrate on are features described in Section III-F, that enable self-organization in a PnP manner. In a further step, we plan to integrate this functionality in ISO/IEEE11073 by exploiting the possibilities introduced by manufacturer specified data protocols in ISO/IEEE11073. From our point of view this might enhance interoperability and could be easily adoptable by industry manufacturers of ISO/IEEE11073 compliant devices. Having realized a PnP u-Health systems, further research will address self-organization on a higher level, where several independent medical devices spawn new functionality by interaction and self-organization. An example would be a BAN, where medical devices react on critical sensor data, by influencing the measurement profile of the whole BAN, dynamically integrating new devices or contacting higher instances.

As described in the appendix, our own implementation is generally intended to be used as a manager development platform. We successfully tested our framework on an Android 2.3 device together with the socket based communication module. Future work in this area addresses problems we observed in combination with our web services module. Furthermore we plan to realize a HDP implementation based on Androids Bluetooth stack BlueZ [8], which already supports HDP functionality.

VII. Related Work

Promoted by ISO/IEEE11073 personal health extensions and its possibilities to realize interoperable u-Health systems, a a large body of research addressed this topic.

In [9] Park et al. proposed an universal Personal Health Device (PHD) adapter, enabling non ISO/IEEE11073 compliant devices to be integrated in a ISO/IEEE11073 based home healthcare environments. The authors conclude that using such a adapter hides complexity and the time consuming process of standardization from legacy device manufacturers and therefore may decrease time to market of interoperable PHDs. However, each legacy communication protocol has to be supported by the adapter in order to operate correctly, which may result in deferring work from standardization back to legacy protocols. In [10] Park et al. developed a ISO/IEEE11073 enabled settop-box for chronic care in home environments. They covered the missing of ISO/IEEE11073 compliant devices by using their adapter approach presented in [9]. Furthermore their settop-box is able to communicate with higher level services like Google Health. Unfortunately they did not discuss the problems shown in Section III, which may reflect the fact that they did not use multiple manager environments (only one settop-box) as well as did not cover the mobility of sensors/patients.

Fioravanti et al. presented a different approach [11] using the ISO/IEEE11073 standard as a middleware platform to store and access data from biomedical sensors in a standardized manner. Therefore they designed an ontology and a sql-scheme adopting the ISO/IEEE11073 data layout allowing higher level services to easily access the medical data as well as easy integration of future ISO/IEEE11073 compliant devices.

In [12] and [13] Serrano et al. proposed guidelines covering the complex process of implementing ISO/IEEE11073 agents and managers as well as introduced their own implementation realizing a ISO/IEEE11073 compliant middleware which permits PnP functionality. Moreover they realized a set of GUI applications enabling the frameworks usage as a testbed for ISO/IEEE11073 compliant devices. During their discussion they also mentioned multiple agent to manager communication as a future task but did not cover the obstacles we discussed above in Section III.

VIII. Conclusion

Beside interoperability, a key feature of u-Health systems is self-organization. Enabling medical devices to enter/leave BANs/HANs dynamically and operate location independently, making them easily replaceable as well as discoverable are main challenges to deal with. Due to its capabilities regarding interoperable device communication and the growing research and industry attention, ISO/IEEE11073 can act as a basis. However, further research and enhancements are needed in order to target self-organization requirements. As described in Section III, security and simple device pairing issues have to be solved. Therefore an overlay protocol or an extension based on ISO/IEEE11073 capabilities for manufacturer defined data protocols are suggested as promising solutions.

Appendix

Because our analysis in Section III was mainly motivated by the findings we made during implementing ISO/IEEE11073, we want to shortly discuss our own implementation, which is completely written in Java and aims at providing a fully featured realization of the ISO/IEEE11073 standards related to home healthcare. However, the process of testing the compliance towards the standard is still in progress at the moment of writing this paper. Due to the typical usage of resource constrained embedded devices as agents (medical devices), our framework generally is intended to be used as a platform for developing manager devices, which are normally less constrained in resource usage than agents, as well as quickly building simulated agents for test purposes. Therefore our

implementation consists of some basic libraries, that provide all necessary standard related features and enables developers to build agents in a plug-and-play manner. For instance, the process of creating the ISO/IEEE11073-10415 weighing scale device specialization only involves writing about 150 lines of code, which simply is about plugging together the needed DIM objects, their attributes and the required communication services. As shown in Figure 3, the provided basic libraries are:

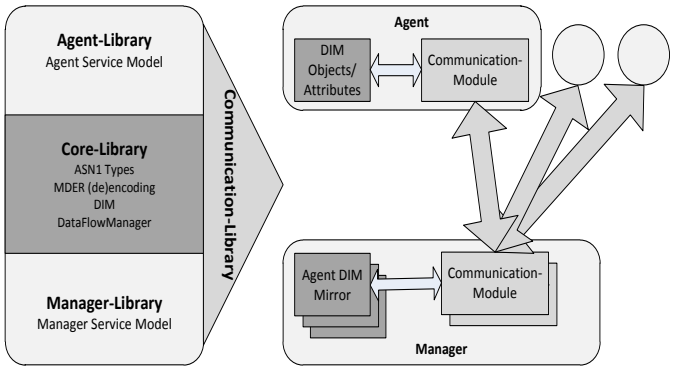

Fig. 3: Overview of implemented ISO/IEEE11073-20601 stack. Agents are build using the base and agent libraries, plugging all necessary DIM objects/attributes together, and choosing a communication-module from the communication library. Managers are build using the base and manager library and adding desired communication modules. While communicating with an agent, they create a mirror of the agents DIM through their service model.

- *core-library:* implements all ASN.1 datatypes and the MDER encoding, the entire DIM and interfaces and modules to execute services on the data contained in the DIM
- *agent-library:* implements the agent side functionality defined in the ISO/IEEE11073-20601 service model, the agents state machine and some utility methods to quickly create agents and manage their communication
- *manager-library:* implements the manager side functionality defined in the service model, the managers state machine, and utility methods to store configurations of already known agents, manage agent discovery and the communication
- *communication-library:* this library implements communication modules, that can easily be bound to either agent or manager, allowing them to communicate over Sockets or Web Services.

Because the intended purpose of ISO/IEEE11073-20601 is to enable platform independent communication and interoperability, we paid special attention towards implementing the service model. To preserve simplicity and modularity, we separated each described service in an agent and manager implementation. Each service can be invoked and executed by creating an *Execution Environment*, which encapsulates environmental data (e.g. the DIM object the service belongs to/should be executed on) as well as needed invocation arguments. Allowed invocation arguments are predefined by each service. The next step towards service execution is to create a *Data Flow*, which simply is a queue of *Execution Environments* to be executed successively. A started *Data Flow* will extract the first *Execution Environment*, put it on a stack and start executing it. To allow execution of a service from within another one, either a new *Data Flow* can be started, which will be executed independently, or the service is able to put a new *Execution Environment* on his own stack. This allows modeling of complex relationships, which is needed for special kinds of the *Data Request* service for instance. Newly created *Data Flows* must be passed to the *Data Flow Manager*, which handles them by either creating a new thread for execution or enqueuing them if resource limitations are reached. An example of the *Data Flow* concept is shown in Figure 4 that illustrates the handling of a *Data Request* by manager and agent side *Data Flows*.

Beside the application layer communication, our framework currently supports two transport layer communication modules. Each module consists of a manager and agent implementation, where the manager side implementation simply extends the agent one with functionality to handle multiple connections and to search for agents. As already mentioned the current implemented modules are based on sockets and Web Services. The socket communication module uses stream sockets and provides an agent discover mechanism via UDP broadcasts. Therefore each agent has to start a separate datagram socket listening on a predefined port for special ping packets. If a ping packet is identified as a valid manager request, the agent answers with an appropriate package, allowing the manager to establish a connection.

In addition to the basic socket based communication, we started developing a communication module based on Web Services. Web Services provide standardized features for discovery and security, which are issues regarding to our analysis of ISO/IEEE11073. The Web Service implementation is based on the Device Profile for Web Services (DPWS) [14] stack provided by the JMEDS [15] project. The DPWS standard combines a selection of WS-specifications together with some enhancements to provide Web Service and PnP [16] functionality for resource constrained networked devices. The communication module enables agent discovery through the DPWS build in WS-Discovery [17] capabilities. A manager trying to discover agents has to search for a predefined service, which is an eventing service started by each agent using the DPWS module. To support agent and manager initiated communication without using Web Services on both sides, we decided to use an eventing service, each manager intending to communicate with an agent has to subscribe to. Additionally, agents offer a simple Web Service method for message reception, which means agent originated messages are sent over the eventing service and manager originated ones are sent over the normal service. As agents allow only one simultaneous connection, manager subscriptions are rejected if the agent is

already connected. Moreover, each ASN1 encoded message, which is mapped to *base64 binary* attachments, is extended with an agent and manager ID identifying both communication partners and allowing them to reject unexpected messages.

After a manager has established a connection to an agent, the agent receives a connect indication, resulting in its state machines switching from the *Disconnected* to the *Unassociated* state. The agent then can start the association procedure defined in ISO/IEEE11073. This involves the service model and *Data Flow Execution Engine* described above.

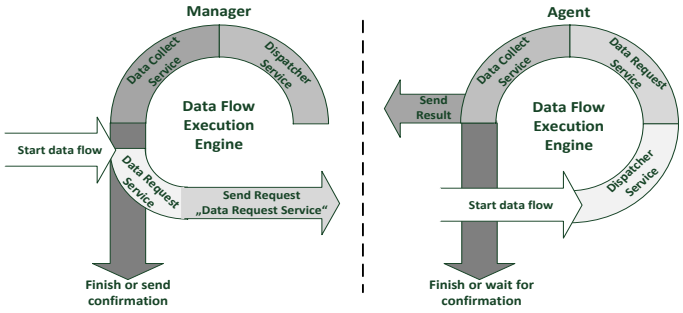

Fig. 4: Basic example of the Data Flow Execution Engine executing a manager data request. Two *Data Flows* are started. One by invoking the Data Request service at the manager and one by receiving the request at the agent.

ACKNOWLEDGMENT

This work was carried out within the scope of the SmartSenior Project sponsored by the Federal Ministry of Education and Research(BMBF, Germany).

REFERENCES

[1] C.C. White, D. Fang, E.-H. Kim, W.B. Lober, and Y. Kim. Improving healthcare quality through distributed diagnosis and home healthcare (d/sub 2/h/sub 2/). In *Distributed Diagnosis and Home Healthcare, 2006. D2H2. 1st Transdisciplinary Conference on*, pages 168 –172, april 2006.

[2] Devices Special Interest Group. Hl7, health level seven. http://www.hl7.org/Special/committees/healthcaredevices/index.cfm.

[3] DICOM. Digital imaging and communications in medicine. http://medical.nema.org.

[4] Iso/iec/ieee health informatics–personal health device communication–part 20601: Application profile–optimized exchange protocol. *ISO/IEEE 11073-20601:2010(E)*, pages 1 –208, 1 2010.

[5] Bluetooth Health Device Profile HDP. https://www.bluetooth.org/technical/specifications/adopted.htm.

[6] Continua Health Alliance. http://www.continuaalliance.org/index.html.

[7] Jianchu Yao and Steve Warren. Applying the iso/ieee 11073 standards to wearable home health monitoring systems. *Journal of Clinical Monitoring and Computing*, 19:427–436, 2005. 10.1007/s10877-005-2033-7.

[8] Linux Bluetooth protocol stack. http://www.bluez.org.

[9] Chan-Yong Park, Joon-Ho Lim, and Soojun Park. Iso/ieee 11073 phd standardization of legacy healthcare devices for home healthcare services. In *Consumer Electronics (ICCE), 2011 IEEE International Conference on*, pages 547 –548, jan. 2011.

[10] Joon-Ho Lim, Chanyong Park, and Soo-Jun Park. Home healthcare settop-box for senior chronic care using iso/ieee 11073 phd standard. In *Engineering in Medicine and Biology Society (EMBC), 2010 Annual International Conference of the IEEE*, pages 216 –219, 31 2010-sept. 4 2010.

[11] A. Fioravanti, G. Fico, M.T. Arredondo, D. Salvi, and J.L. Villalar. Integration of heterogeneous biomedical sensors into an iso/ieee 11073 compliant application. In *Engineering in Medicine and Biology Society (EMBC), 2010 Annual International Conference of the IEEE*, pages 1049 –1052, 31 2010-sept. 4 2010.

[12] M. Martinez-Espronceda, L. Serrano, I. Martinez, J. Escayola, S. Led, J. Trigo, and J. Garcia. Implementing iso/ieee 11073: Proposal of two different strategic approaches. In *Engineering in Medicine and Biology Society, 2008. EMBS 2008. 30th Annual International Conference of the IEEE*, pages 1805 –1808, aug. 2008.

[13] I. Martinez, J. Escayola, M. Martinez-Espronceda, L. Serrano, J. Trigo, S. Led, and J. Garcia. Standard-based middleware platform for medical sensor networks and u-health. In *Computer Communications and Networks, 2008. ICCCN '08. Proceedings of 17th International Conference on*, pages 1 –6, aug. 2008.

[14] Device Profile for Web Services. http://dows.oasis-open.org/ws-dd/dpws/wsdd-dpws-1.1-spec.html.

[15] Java Multi Edition DPWS Stack. http://ws4d.e-technik.uni-rosotck.de/jmeds/.

[16] Microsoft Corporation. Introducing device profile for web services. http://download.microsoft.com/download/b/5/3/b53ea430-dbe5-440c-a308-df97b10280b7/introducing_dpws.pdf.

[17] Web Services Dynamic Discovery. http://dows.oasis-open.org/ws-dd/discovery/1.1/wsdd-discovery-1.1-spec.html.