Protocol Adapter: A Reusable Solution to Interoperability and Integration Issues in mHealth Data-collection Systems

Alexandru Serbanati, Marcello Morena and Laura Lancia
Consorzio Roma Ricerche, Via Giacomo Peroni 130, Rome, Italy

Keywords: Medical Device, Data Collection, Interoperability, mHealth, Machine-to-Machine, Bluetooth.

Abstract: Healthcare has been changing in the last years due to several inputs, the main ones being moving from assistive to preventive care and the introduction of patient-centric care models. In support to this tendency, the number of consumer mobile applications for remote healthcare delivery is rapidly increasing and the use of mobile medical sensor devices is also following. Standardization in the domain of data collection for mHealth is still moving its first steps and, as a consequence, those who aim at developing remote healthcare solutions must face significant problems related to the heterogeneity of sensor devices. In general, issues related to low interoperability and low code-reusability of data collection software in mHeatlh severely limit further developments in this sector. These issues have been addressed thanks to the Protocol Adapter which provides a single, uniform interface for both the collection of rich data and the management of medical devices. This article gives an overview of this component and its development process in order to provide a better understanding of its value when integrated in mHealth applications. After an introduction to the state of the art, the requirements for the data collection in mHealth systems are discussed. The design phase is then described along with the final architectural solution and the features of this free, open source implementation for Android are discussed. Finally, future works on the Protocol Adapter are discussed in the hope to attract the interest of device producers and of mHealth developers.

SCIENCE AND TECHNOLOGY PUBLICATIONS

1 INTRODUCTION

In a context where the patient base is growing along with the rising of the population average age, prevention and management of (multiple, coexisting) chronical diseases has an increased importance in the proposed healthcare models. As a result, in many countries worldwide, cost-effectiveness has been one of the main drivers in the changes that are still undergoing in this sector (Health and Human Services, 2011).

Another trend in this field is the focus that has been placed on patient-centric approaches aiming at moving the treatment context from the hospital recovery to locations more comfortable for the patients (Fass, 2007, and Eurobarometer, 2007). Even WHO in its 2016-2026 roadmap envisions a shift toward “outpatient and ambulatory care” in an effort of reorienting the model of care (WHO, 2015).

Like in many other domains, ICT supported these changes in many ways and, as a result, new models for care delivery have been developed. It has been demonstrated that “ICT-based services for domiciliary care improve quality of life for older people and carers, access to qualified long-term care, and the integration of health and social care services” (Carretero et al, 2012).

1.1 Problem Statement and Objectives

The solutions envisaged by mHealth aim to leverage the connectivity and processing capabilities of mobile devices to provide the relevant carer figures with fresh and complete clinical data collected from patients without imposing their presence in the clinic for a reduced stress and increased comfort of the same. To this purpose, a great variety of medical devices that sense different clinical parameters and communicate with different communication protocols are available on the market. Medical devices in the context of this paper are intended sensor devices that can sense clinical parameters and which are small-sized and ergonomic in order to enable patients to wear or transport them easily.

To integrate such heterogeneous devices in mHealth applications, often the only solution is to

Serbanati, A., Morena, M. and Lancia, L.
Protocol Adapter: A Reusable Solution to Interoperability and Integration Issues in mHealth Data-collection Systems.
DOI: 10.5220/0005845605500560
In *Proceedings of the 9th International Joint Conference on Biomedical Engineering Systems and Technologies (BIOSTEC 2016) - Volume 5: HEALTHINF*, pages 550-560
ISBN: 978-989-758-170-0
Copyright © 2016 by SCITEPRESS – Science and Technology Publications, Lda. All rights reserved

develop vertical, device-specific modules for the management of the connection and of the data collection, increasing the architecture complexity and reducing the part of the application that could be re-used.

This paper describes the design of a framework for data-collection to be used in mHealth mobile platforms that aims at avoiding these issues by providing a public, well-defined, well-documented and reusable interface for interacting with medical devices.

The result is the Protocol Adapter (PA), a modular and extendable software architecture that provides a single management and data collection interface for several, heterogeneous medical devices. In this way, connectivity aspects are separated from business logic and GUI design so that implementation efforts can concentrate on these aspects rather than on integrating the communication with medical devices.

1.2 Structure of the Paper

In the following chapters the design process is described and the resulting architecture of the PA is discussed. In Chapter 2 an overview of the state of the art that was taken as starting point is described: a brief overview of the reference architectures used in mHealth is provided in order to understand the role and importance of sensor medical devices in the "big picture". In Chapter 3 the requirements are discussed and in Chapter 4 the analysis and design phases are addressed. In Chapter 5 the resulting architecture and its implementation in Android are described. In Chapter 6 possible future work topics will be introduced and a brief highlight of the most important achievements completes the article.

The suggested audience is mainly ICT professionals, i.e. developers, software analysts and designers, system architects, etc.

2 BACKGROUND AND STATE OF THE ART

Under the drivers of scalability, domain-wide coherence, interoperability and the appeal of increasing development efficiency, the need for an open and widely accepted architecture for mHealth was identified long ago (Estrin and Sim, 2010).

Despite the fact that standardization efforts are still to bear fruit, when analysing mHealth applications, common architectural solutions can be identified and scientific and industrial effort is invested in this field.

2.1 Architectural Reference for mHealth

In (GSMA, 2012) a good overview of the architectural characteristics of mHealth applications is provided. It is also interesting to note that similarities exist with the reference architecture for mobile cloud computing (Dinh et al., 2013). Both foresee the presence of a mobile device that acts as sink and processor for the raw data coming from the sensor and, at the same time, as an Internet gateway for the sensors device.

The Communication Model of mHealth applications can be viewed as a specialization of the Internet of Things (IoT) one. In Figure 1 the components of the model, based on and extended from the Communication Model from the IoT-A project (Walewski, 2011) are shown for reference and divided by the base platform they run on.

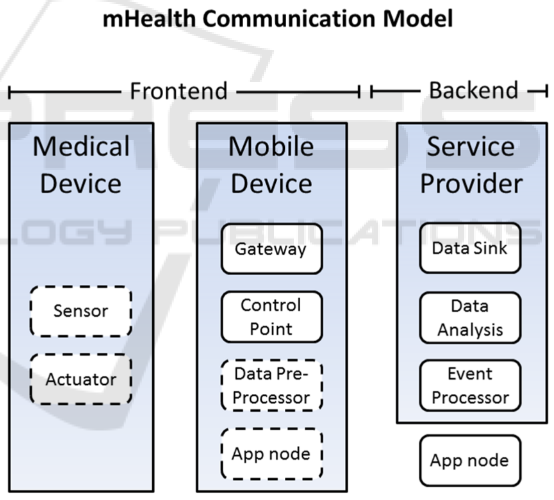

Figure 1: mHealth communication model components. Dashed lines represent optional components.

It is important to note that other conventions exist. One relevant example is the term gateway: from the initial meaning related to protocol adaptation at network (NWK) layer level, the concept evolved to include security, management, data aggregation and even application-specific features (ITU-T, 2014). In the mHealth context however, all these functionalities are provided by the mobile device, which indeed is often called gateway.

With the evolution of the capabilities of such devices into smartphones, the mobile devices also came to host application-level software in order to enable the interaction between the distributed mHealth application and the user.

The data collection flow usually has the following steps in mHealth systems:

1. raw data is collected by the means of sensors embedded in medical devices (also called sensor devices sometimes);
2. the raw data is forwarded (generally) wirelessly to the gateway device, e.g. a mobile or a smartphone;
3. the data *can be* pre-processed locally and *can be* displayed for the user's benefit on the display of the mobile (if the mobile is also an application node),
4. the collected data is then sent by the gateway over the Internet to a data sink in the backend;
5. the data is filtered, aggregated and/or stored;
6. the resulting information is made available to users over the web/private network as services;
7. finally, an event processor notifies users that subscribed to specific events or clinical alerts.

The users of mHealth systems range from patients and care personnel (i.e. medical staff, relatives of the patient or informal carers) to other ICT systems of the healthcare ICT environment in a machine-to-machine (M2M) point of view.

The information flow for control and management functions will not be investigated since it is very specific to the examined application.

2.2 Peripheral Connections

While many technologies and standards exist for connecting medical devices to the mobile devices, generally data exchange happens either through a Wireless Personal Area Network (WPAN), or through a Wireless Local Area Network (WLAN). At the moment of writing this paper, almost all mobile devices are equipped with different versions of Bluetooth and Wi-Fi interfaces for what concerns WPAN and WLAN respectively.

An overview of the available communication technologies and relative standardization activities can be found in (mHISS, 2013). The majority of the medical devices used in mHealth solutions uses Bluetooth technology though for communication. When designing medical sensor devices, the choice between using Bluetooth or Wi-Fi is based on the bandwidth and range requirements of the medical sensor device. If the Bluetooth technology can satisfy them, then it is preferable to use it instead of the Wi-Fi.

The reasons behind this choice are that 1.) Bluetooth power consumption is much lower (Lee et al., 2007) than Wi-Fi for peer-to-peer connections, 2) Wi-Fi either needs a network infrastructure which Bluetooth doesn't or 3) in ad-hoc mode, Wi-Fi interaction is less user-friendly than the Bluetooth one, which was designed for this specific scenario with usability in mind.

Moreover, one of the main drawbacks of Bluetooth, its low speed compared to Wi-Fi, is no more an issue since v3.0 and later implementations can also include the High Speed (HS) optional feature which enables handover to the alternate Wi-Fi MAC/PHY in order to achieve high data rates (Bluetooth SIG, 2009).

3 SYSTEM REQUIREMENTS

The PA development was based on requirements that resulted from a refinement process that consisted in three steps.

The starting point was obviously the business goal described in the project charter: to design a software component for mobile platforms that would enable mobile application developers to easily integrate the communication and the management of the vast majority of mobile medical devices available today off-the-shelf. A first discussion on these requirements led to implications on the architectural constraints related to scalability. Licensing policy was also taken into account and open source release was decided in accord with company policies and in order to achieve the favour initial distribution and take-up.

In a second step, stakeholder requirements were gathered. For the above reasons, in the PA project, users are of two types: developers and medical personnel (as well as patients). While one could argue that the *end* users eventually are the patients and the medical personnel, but we thought that it was right to include the developers too since they will be the first users of the results of the project both when integrating the PA implementation for collecting medical data and when they would use the PA architecture to extend existing implementations. In fact in the requirement elicitation process, use case scenarios were developed for both categories.

The PA was developed in the frame of the FI-STAR project, as one of the components of the platform frontend. As such, the PA development team could leverage the use case models and generally the requirements documentation of the pilots of the FI-STAR project in order to derive functional requirements. In particular, six pilots used medical devices and their documentation helped a lot in getting a good understanding of the usage context.

In the following we provide a brief, narrative outline of the system requirements, highlighting the requirements that were identified during the later stages.

3.1 Functional Requirements

Starting from the previously described business goal statement, functional requirements for the PA were initially derived from existing FI-STAR pilot documentation. Since this project is centred on this topic, a reference model for communication interoperability based on the work from Tolk (Tolk et al., 2007) was adopted. Figure 2 shows where the PA impacted for the achievement of communication interoperability.

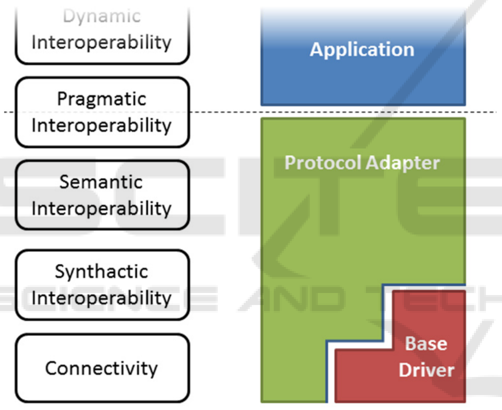
Dynamic Interoperability
Pragmatic Interoperability
Semantic Interoperability
Synthactic Interoperability
Connectivity
Application
Protocol Adapter
Base Driver

Figure 2: Interoperability reference model and impact of the Protocol Adapter.

As the aforementioned use case models and medical requirements were too loosely defined, further work towards the definition of the more technical functional requirements was necessary.

An initial set of 32 portable medical devices that measured 11 clinically relevant variables (SpO2, respiratory rate, spirometry, heart rate, blood pressure, pulse, body temperature, ECG, weight, acceleration, blood glucose level) was collected thanks to the support of about 15 partner organizations.

While the ultimate aim is to support all the (types of) devices from the aforementioned set, only a subset was chosen for actual employment in the FI-STAR pilots, based on the requirements of the clinical partners. The PA architecture had to mandatorily support these devices as a high priority requirement.

The identified functional requirements were:

- to support (also with implementations) all device types used by FI-STAR pilots;
- to support all interaction patterns between the mobile and the medical device;
- to collect clinical data from all devices used in the use cases;
- to support the largest number of device models on the market;
- to provide the clinical data in a single format;
- to provide information about the status of medical device;
- to manage the connection with the medical device.

As we believe that this technical details might be interesting for the reading audience, the relevant ones will be detailed in the following.

3.1.1 Interaction Patterns

Medical sensor devices are very heterogeneous for what concerns the way they operate. The difference regard:

- the operations that need to be performed in order to establish a communication channel (such as Bluetooth pairing or physical attachment);
- the sequence of operations that establish the communication channel itself: whether the mobile or the medical device is the initiator, i.e. which one initiates the communication;
- the role of the devices: related to which is the server and which the client; please note that this is not necessary related to the initiator role;
- the duration of the connection: some devices maintain it until the application decides to terminate it, others automatically cut it off as soon as they have sent the data;
- the necessity to perform a setup before operation: more complex devices need to be provided with operational parameters upon connection in order to start operating;
- the need to send a command in order to start data acquisition; this command can also include information about the measurements to be performed;
- the way devices send the data: some devices send data automatically and immediately after performing the measurement, while others send the data only after the proper command is received.

All of these differences have to be taken into account. Moreover, information about the status of the device must be provided to the application.

3.1.2 Data Collection

Data collection, i.e. the process by which the values measured by the medical sensor devices are collected by the mobile devices, can use several different communication technologies as well as different protocols. The main concern, derived directly from the business goal, is that the Protocol Adapter must be able to integrate all these options to collect data.

This requirement set is also related to the available implementation options: while the design drives the implementation, when designing it is important to know what are the implementations constraints. In our case, we had to deal with the fact that many medical devices used (sometimes proprietary) protocols for which closed libraries existed. These protocols – and the relative libraries – reach different levels of the communication stack.

The initial set of devices used different communication solutions at PHY/MAC layer level: Bluetooth, audio jack, and Wi-Fi. However, even in the Bluetooth device set, different Bluetooth Profiles were used: Health Device Profile (HDP) (Bluetooth SIG, 2012), Smart Bluetooth (Bluetooth SIG, 2010), or open and closed protocols over SPP. Moreover, it was decided to keep the most generic approach possible in order to be able to support all kind of existing devices and to make it possible to easily provide support even to future ones.

3.1.3 Data Provision

All the previously illustrated differences have implications on the syntactic and semantic level. The following requirements were gathered with the help of the developers of the pilots and are related to the medical needs of the pilot use cases:

- the data has to be provided with a uniform syntax and semantics, despite differences in the single device protocol,
- the data has to be provided as soon as it arrives,
- the data has to be as rich as possible: no information has to be omitted and, moreover, contextual information that regards the device should be sent along with the measurement.

3.2 Non-functional Requirements

Some constraints derived mainly from the high level goals, e.g. from the project charter:

- mHealth architecture compliant: the PA had obviously to comply with the aforementioned mHealth architecture;
- FI-WARE Protocol Adapter architecture compliant: as part of the Future Internet programme, the PA was initially supposed to implement the interfaces of the Protocol Adapter component of the Internet of Things Services Enablement (FI-WARE, 2015) FI-WARE chapter;
- the component had to be easily extensible in the future in order to support new technologies, maintaining at the same time backward compatibility;
- M2M and IoT readiness: while the current mHealth architecture is slightly influenced by the IoT one, current architectures appear to be centralised and the M2M approach is only considered in the backend part. The PA team expects that this situation will evolve and that data will be eventually provided directly “at source”, i.e. from the mobile device, allowing the user to really be in control of his data.

3.2.1 External Interface

For the PA, the external interface was the interface for communicating with the application that needed to collect the data. Such applications can be either local or remote or, in some cases, a local application will use the data but will also forward it to the backend. So, no assumption could be made on whether the consumer of the information was local, as in the case of the gateway pre-processing the data, or remote, i.e. in the backend environment.

Another non-functional requirement related to the external interface was the need to reduce complexity and increase the possibility to reuse both knowledge and code in order to increase development efficiency. This was both a business goal and a developers’ need.

3.2.2 Deployment Requirements

Different operating systems have different architectures, resulting in different resources that can be used, different security restrictions, different way of integrating components and communicating with them, different best practices, and so on.

Also, both the PA implementations and the applications that are going to use them must use the APIs and resources provided by such platforms (i.e. operating systems) or on APIs provided by vendors which, in turn, run on the lower layers of the OS.

3.2.3 Performance and Other Requirements

The PA is meant to mediate and simplify interactions with external devices connected to the mobile device where the application and the PA itself run. When integrating the PA to collect data from a medical device, it will not degrade significantly the performance of the system when compared to a system directly integrating the medical device control into the application itself. This means for example that the PA will have to be able to manage fast data flows as in the case of ECG sensors. This kind of performance can also be impacted by the number of functionalities that can be accessed through the PA compared to the number of functionalities provided by the device.

It was also flagged as important that product vendors also needed to be able to extend the PA with their own software that could be closed source or generally subject to different licensing policies compared to the PA.

4 DESIGN AND ARCHITECTURE

In first stance, the initial (i.e. the broadest) set of device models and the sequences needed for their use were considered. To this purpose, since the interaction had to be carried out programmatically, their development kits and protocol documentation was analysed. Interaction models were abstracted from this information. Several different use cases (and models) requiring different sequences of interaction were identified. All were taken into account except that of a device that was designed to communicate only with a proprietary app in order to send the collected data directly to a third party, predefined Internet server. It was found that several models would be needed for representing these device types and yet different models had subsets of common features.

Indeed we found out that it was better to abstract characteristics than to use monolithic models. These characteristics are related to connection roles and modes as well as to operational requirements. In the following a list of such characteristics is provided:

- the device acts as a connection initiator;
- the device supports external configuration: four behaviours were identified (configuration supported only at startup, supported at runtime, supported both at startup and runtime, not supported);
- the device supports external commands;
- the device can be detected programmatically prior to connection;
- the device needs pairing (or other previous setup) prior to connection;
- the device can be disconnected programmatically;
- the development kit provides a reliable way to know if the device is connected and operating.

For what concerns data collection and interoperability, it was clear that several protocols were used by the devices on the market. The individual syntax and semantics had thus to be abstracted by the PA in order to provide a single external interface for managing the data collection from medical sensor devices.

The use of such protocols had to be supported at different levels of the communication stack and they guaranteed different levels of interoperability. Some protocols were only documented, while some devices provided data collection features through APIs. Thus, the resulting data representation varied from one device type to another.

The HDP device type was considered as single type on par with device models for the purpose of this analysis because the reference Personal Health Device standard (i.e. the IEEE11073 standard family) directly provides pragmatic interoperability for all the compatible device models. Indeed implementations of the PA that cover HDP already have a large pool of compatible device that are supported.

Finally, the analysis also resulted in the rejection of some requirements. For instance, the requirement to provide a component that implemented the FI-WARE Protocol Adapter interface had to be dropped because it was conflicting with the best practices for development on mobile devices. In particular, using NGSI-9/10 Context Management specifications (Open Mobile Alliance, 2012) over REST connections could not be fulfilled because workarounds for the fact that mobile devices had dynamic network addresses would severely impact on the device batteries.

4.1 Notes about Architecture Design

When designing an abstract model that could represent all types of devices, the right level of abstraction had to be found. Emphasis was thus put on providing a high level model with relatively few and generic functionalities that could be easily understood and mastered.

In order to understand at what level of the interoperability stack would the PA be placed, the available APIs and the documentation of communication protocols was analysed. Contextual information and descriptions of the measurements was available and could be provided along with the raw values communicated by the medical devices. Moreover, the sequence of operations needed for the proper operation of devices changed from one type of devices to another and trying to automate this part would increase the complexity of the PA interface. For these reasons it was decided against trying to provide fully-fledged pragmatic interoperability.

It was thus decided that the PA would only provide descriptive information to the upper layers (i.e. the application) about the device types and their characteristics. While this information is not sufficient to operate all devices and it is not meant to replace the knowledge about the operation of more complex devices, it helps in avoiding the misuse of the PA. We take as granted that, if a developer has to use a device that requires the sending of a command to start the acquisition of data, this is known to the developer.

The devices were characterized by: the ID of the device, their serial number, the model name, manufacturer name, the physical address of the device and a collection of the attached sensors to the device represented using the Sensor Model.

The sensors, in turn, were characterized by: the name of the sensor, the name of the property measured by the sensor and the measurement unit of said property.

The Device and Sensor Models are meant to be used together to describe the devices and their sensors.

After the device model was defined, the internal architecture of the PA had to be designed. In order to have a single interface towards the application and to allow third parties to expand the PA support independently, device specific functionalities were separated: a specific component called Device Adapter (DA) would manage low-level, device specific functionalities while a higher level component, the Protocol Adapter Manager (PAM), would provide the single point of interface with the application and DA-management functionalities.

The implementations of the DA are required to be able to recognize the devices that they could manage and provide protocol adaptation for a given type of sensor medical devices.

Moreover, in order not to limit the development possibilities, it was also chosen to allow in principle the existence of more DAs able to manage the same types of devices and even to allow their implementations to coexist on the same mobile device: the user (human user or application) will then have to decide which implementation should manage which device.

The communication requirements of the devices were analysed and a set of message types was defined for what concerns the interaction of the PAM with both the application and the DAs. The majority of interactions that could be started by the application resulted in an asynchronous feedback from the device because the device had to perform some physical measurement (in some cases even with the user's implication) and a synchronous response could not be guaranteed. However, some management interactions between the PAM and the DA which don't depend on the medical devices could be carried out synchronously.

For what concerns the collection of data, this has to be translated from the original format to a single common format. In this way, applications will receive the data structured in a uniform way despite the different device source, the different measurement types, the different formats that are provided in input and so on. In turn, this will allow developers both to use only one interpretation routine for all the devices they used and also to reuse existing code over different projects.

For this reason, an Observation Model based on the information types returned by the considered devices was defined in order to provide an uniform information model. In the design, M2M requirements were kept into account along with the previous experience that the team had in the field of data collection and IoT. This impacted for example on the data collected from devices that provided measurements continuously, such as for example electrocardiographs. It was decided that in the event in which the medical device sent streams of data, this had to be grouped in packets as an internal mechanism of the DA, while the PAM should not support data streaming. One of the main reasons for this choice was that such type of data is not generally supported in M2M frameworks where data is represented as a static, albeit possibly complex, resource.

The model was meant to represent in a well-known format every possible measurement carried out by every possible device. It includes the name of the property which the measurement refers to, the measurement unit, the time of the measurement, the duration of the measurement and a collection of raw samples of the measured value.

4.2 Resulting Architecture

Based upon the previously described analysis and design, the architecture of the Protocol Adapter is shown in Figure 3. The PAM provides an interface to the application above and manages from an operational and communication point of view any number of DAs below it.

In more detail, the DA must establish (or allow the medical sensor device to establish) a communication channel. If needed, it then must use this channel to send the configuration or specific commands needed for bringing the device to an operational status. Finally, it will provide protocol adaptation between the device specific formats and the single PAM format, for which it actually acts as an abstraction layer. If vendor drivers are available for a given device, DA designers can use them to ease the development process.

The PAM on the other hand must discover all the DA instances available on the system upon start and afterwards it must manage their life cycle. It must provide all the collected data to the application, handling all the measurements and the events generated by the DA. In the other way, it also has to route the application device commands and management inquiries to the right DA.

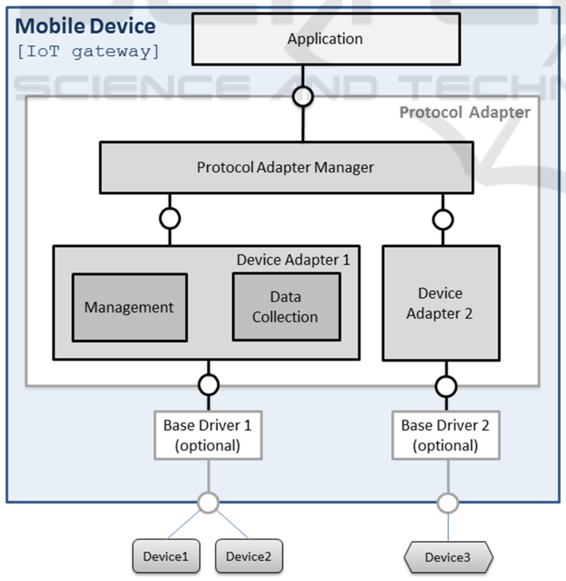

Figure 3: Protocol Adapter Architecture.

Also, Device, Sensor and Observation logical models were used to represent characteristics and functionalities of the managed devices, integrated sensors and collected data. These models are used together to coherently pass knowledge above these concepts from one component to another.

5 ANDROID IMPLEMENTATION

To respect the requisites of flexibility and expandability, it has been chosen to implement the PA as a collection of separate Android applications communicating via Inter Process Communication (IPC) mechanism (Google, 2015a).

5.1 Components and Interfaces

In particular, the PA Android implementation consists of a certain number of DA applications and one PAM application, each of them deployed through a specific .apk. In the following part of this chapter DA and PAM will be used with the meaning of implementations, i.e. Android applications, unless explicitly stated that they are meant as models.

The two crucial design decisions about the Android implementation were: what IPC mechanism to choose amongst the ones offered by Android and how to implement the discovery phase.

The choice of the right IPC mechanisms was based on two development requirements. In first stance, reliable communication channels between the PAM and DAs and between the PAM and third party applications was needed. In second stance, it was necessary to package all the software needed for the establishment of said channels inside a library, in order to make the integration of the PA as simple as possible for third party developers. In the end we chose Android Interface Definition Languages (AIDL). In this way, the Java implementations of the Device, Sensor and Observation models, called respectively DeviceDescription, SensorDescription and Observation, could be packaged along with the AIDL interfaces and the Capabilities class used to represent DA properties and distributed as a library. Also this allowed to have a reliable return value for the methods that needed one.

This method is used so that, after a successful binding to services, Android applications can invoke methods of remote objects (belonging to other Android applications) as if they were local.

At this point, two kinds of channels existed: one that enabled the communication of the DAs with the PAM, and the channel between the third party application and PAM. Since we wanted bidirectional communication for each channel, using AIDL we created a total of four Java interfaces, two for each channel. Every time a service is bound to, depending on the kind of the channel, two objects implementing the related interfaces are exchanged between the application that implements the service and the application that is bounding to it. This allows a bidirectional communication that is carried on in the most natural way for Java software: simply invoking methods on objects.

Finally, the DAs and the PAM were implemented as Android Services because, being them software modules designed to run in the background without a direct user interaction and to offer APIs to third party applications, the Service paradigm was the most appropriated.

5.2 Discovery Implementation

The discovery feature was implemented with another Android IPC mechanism that is more common and more lightweight than AIDL, yet less powerful and less reliable: the Intent system (Google, 2015b). When using Intents to send messages between applications, there has to be a software component inside the receiving application called Broadcast Receiver. This component must declare what types of Intents it is interested into and is invoked every time that a suitable Intent is dispatched in the system by a sender application. However, using this facility, the sender application can never know if the Intent has been successfully delivered and it is impossible for the receiving application to acknowledge or directly reply to the received message.

In the PA, every DA implements a Broadcast Receiver that obtains all the discovery Intents generated by the PAM and sends back a reply Intent (which the PAM has a Broadcast Receiver for) to notify its presence on the system. Moreover, the DA sends, together with the reply Intent, an object called Capabilities; this object contains all the relevant information about the DA itself that the PAM needs to know in order to properly handle it.

5.3 Operation

The application binds to the Manager using facilities included in the library and establish the bidirectional communication channel. The PAM, upon the start, performs the discovery process to retrieve information about all the DAs available on the system. Once this phase is done, the PAM activates all the DAs that it needs by binding to their related services and establishing a bidirectional communication channel with every one of them. At this point, the third party application gets notified that the initialization phase is over. From now on, devices that are initiators can spontaneously connect and send data, while devices that are not initiators can be connected to (upon request of the third party application) and triggered to send data. At a certain point, all types of devices will eventually be connected. When this happens, the third party application gets notified and receives all the details of the newly connected device via a DeviceDescription object. Every time a device sends new data, this is forwarded to the third party application encapsulated in an Observation object. These operations continue until the third party application is bound to the PAM. When the third party application decides to terminate the bound with the PAM, it will be shut down gracefully together with all the active DAs, releasing all the connected devices in the process.

The descriptive models of devices, sensors and observations are implemented as java classes called respectively DeviceDescription, SensorDescription and Observation. They provide a well-known structure that could be used to encapsulate information forwarded to the PAM or to the application.

6 CONCLUSIONS AND WAY FORWARD

The PA is a free, open source component that succeeds in bridging the interoperability gap that exists at the low level in the mHealth communication domain. The development process and the resulting components with their features have been presented in this paper, along with their Android implementation, in the hope to raise the interest of two stakeholders that the authors believe to be essential to the success of the PA: the mHealth application developers and device manufacturers.

The Android version of the PA source is available at https://github.com/theIoTLab/ along with DAs for HDP devices and for the Zephyr BioHarness 3 device. Other DAs have been developed but could not be published due to licensing issues.

6.1 Further Developments

EHealth is a domain that is rapidly growing, continuously providing new solutions and integration with traditional healthcare. To keep the pace, the PA will need to adequate to major reference architectures, standards and best practices.

For this reason, while the PA is already a FI-STAR component, we would also like to integrate the PA in the FI-WARE architecture as a native Generic Enabler.

Another development thread is related to supporting new device models and device types. For example, the development of an Android DA implementation for Smart Bluetooth is under evaluation.

Last but not least, we think that the security aspect of the interaction with the devices should be improved. Currently, for example, there is no way for the gateway (or for the application running on the gateway) to authenticate the medical device. Unfortunately, this cannot be implemented on the PA side only, because it also requires support from the medical devices. Yet, we believe these features to be critical from a long term perspective since they are required for securely implementing the support of mHealth actuators.

ACKNOWLEDGEMENTS

The research on the Protocol Adapter was funded in the frame of the FP7 FI-PPP Phase II project Future Internet Social and Technological Alignment in Healthcare (FI-STAR) under grant no. 604691.

REFERENCES

Bangash, J. I., Abdullah, A. H., Anisi, M. H., and Khan, A. W., 2014. A survey of routing protocols in wireless body sensor networks. In *Sensors* vol.14, n. 1, pp. 1322-1357. MDPI.

Bluetooth SIG, 2012. Health Device Profile Specification. Available online from https://developer.bluetooth.org/TechnologyOverview/Pages/HDP.aspx.

Bluetooth SIG, 2009. Bluetooth High Speed Technology, available online at https://www.bluetooth.org/en-us/specification/adopted-specifications.

Bluetooth SIG, 2010. Bluetooth Smart (Low Energy) Technology. Available online at https://developer.bluetooth.org/TechnologyOverview/Pages/BLE.aspx.

Carretero, S., Stewart, J., Centeno, C., Barbabella, F., Schmidt, A., Lamontagne-Godwin, F., Lamura, G., 2012. Can Technology-based Services support Long-term Care Challenges in Home Care? *Analysis of evidence from social innovation good practices across the EU CARICT Project Summary Report*. Publications Office of the European Union.

Dinh, H. T., Lee, C., Niyato, D., and Wang, P., 2013. A survey of mobile cloud computing: architecture, applications, and approaches. In *Wireless communications and mobile computing*, vol. 13, n. 18, pp. 1587-1611. Wiley Online Library.

Estrin D, and Sim, I., 2010. Open mHealth architecture: an engine for health care innovation. In *Science*, vol. 330, n. 6005, pp.759–760.

Eurobarometer, Special, 2007. Health and long-term care in the European Union." *Special Eurobarometer* 283.

Fass, L., 2007. *Patient-centric healthcare*, IET.

FI-WARE, 2015. Internet of Things Services_Enablement. Available online at http://forge.fiware.org/plugins/mediawiki/wiki/fiware/index.php/Internet_of_Things_(IoT)_Services_Enablement.

Fricker, S. A., Thuemmler, C., Mival, O, 2013. Technical Requirements and Architecture Report including Open Call Requirements. EC FP7 FI-STAR (604691) project deliverable D1.1.

Google, 2015a. Bound Services, Developer's documentation, available online at http://developer.android.com/guide/components/bound-services.html#Creating.

Google, 2015b. Intents and Intent Filters, Developer's documentation, available online at http://developer.android.com/guide/components/intents-filters.html.

GSMA, 2012. Connected Mobile Health Devices: A Reference Architecture.

ITU-T, 2013. Common requirements and capabilities of a gateway for Internet of things applications, Recommendation ITU-T Y.2067. In *Global Information Infrastructure, Internet Protocol Aspects And Next-Generation Networks*, ITU.

Lee, J. S., Su, Y. W., and Shen, C. C., 2007. A comparative study of wireless protocols: Bluetooth, UWB, ZigBee, and Wi-Fi. In *Industrial Electronics Society, 2007. IECON 2007. 33rd Annual Conference of the IEEE*, pp. 46-51. IEEE.

mHIMSS, 2013. mHIMSS Roadmap: Standards and Interoperability. Available online at http://www.himss.org/ResourceLibrary/genResourceDetailPDF.aspx?ItemNumber=21293.

Open Mobile Alliance (OMA), 2012. NGSI Context Management, Version 1.0.

Schweitzer, J., and Synowiec, C., 2012. The economics of eHealth and mHealth. In *Journal of health communication*, vol. 17, n. sup1, pp. 73-81. Taylor & Francis.

Walewski, J.W., (ed.), 2011. Initial Architectural Reference Model for IoT. EC FP7 IoT-A (257521), project Deliverable Document D1.2.

World Health Organization (WHO), 2015. Reorienting the model of care, In *WHO global strategy on people-centred and integrated health services*, WHO Press.

Tolk, A., Diallo, S., Turnitsa C., 2007. Applying the levels of conceptual interoperability model in support of integratability, interoperability, and composability for system-of-systems engineering. In *Journal of Systemics, Cybernetics and Informatics*, vol 17, n. 5, pp. 65-74.

SCIENCE AND TECHNOLOGY PUBLICATIONS

U.S. Department of Health and Human Services, March 2011, *2011 Report to Congress: National Strategy for Quality Improvement in Health Care*. Available online at http://www.ahrq.gov/workingforquality/nqs/nqs2011annlrpt.htm.

SCITEPRESS
SCIENCE AND TECHNOLOGY PUBLICATIONS