Implementation Guidelines for an End-to-End Standard-based Platform for Personal Health

I.Martínez, J.Escayola, J.Trigo,P.Muñoz, J.García
Aragon Institute for Engineering Research (I3A/GTC)
University of Zaragoza (UZ)
c/ María de Luna, 3, 50018 Zaragoza, Spain
{imr, jescayola, jtrigo, pmugnoz, jogarmo}@unizar.es

M.Martínez-Espronceda, L.Serrano
Electrical Electronics Engineering Dept.
Public Univ. Navarra (UPNA)
Campus de Arrosadía s/n. 31006 Pamplona, Spain
{miguel.martinezdeespronceda, lserrano}@unavarra.es

***Abstract*— There is a need to develop open sensors and middleware components that allow transparent integration and plug-and-play interoperability of Medical Devices (MDs) and Computer Engines (CEs). The use of standards seems to be the internationally adopted way to solve these problems and allow implementing ubiquitous solutions, including wearable devices, focused on the new paradigm of Personal Health (p-Health). However, to the best of our knowledge there is no experience where the full chain from personal healthcare environment to Health Centre had been implemented using interoperability standards. Even though there have been some initiatives to combine different standards, the vision of an entire end-to-end standard-based system is not yet a fact end-to-end solution. This paper presents the implementation guidelines of a ubiquitous platform for p-Health. It is end-to-end standards-based, using ISO/IEEE11073 in the patient environment and EN13606 to communicate the information to an Electronic Healthcare Record (EHR) server. The platform has been designed to comply with the last ISO/IEEE11073 and EN13606 available versions and tested in a laboratory environment to demonstrate the feasibility of an end-to-end standard-based solution.**

Keywords—EN13606 standard, end-to-end design, implementation guidelines, ISO/IEEE11073 standard, p-Health.

I. INTRODUCTION

The remarkable growth of Information and Communication Technologies (ICTs) during the last decades has fostered the development of innovative healthcare applications. The traditional fixed hospital locations for these telemedicine applications (usually called e-Health) have gradually evolved to new mobile environments (m-Health), user-centered personal applications (p-Health) and finally to ubiquitous healthcare systems (u-Health).

All these systems, applications and environments make use of Medical Devices (MDs) to acquire user biomedical signals and measurements that can be sent to HealthCare Information Systems (HCIS) in order to be subsequently analyzed by physicians. To fulfill the needs of the new health environments, Personal Health Devices (PHD) provides users ubiquitous and remote supervision by including facilities such as wireless technologies and high portability.

Manufacturers of MDs all around the world have been fighting for a place in the market by creating their own protocols to transmit and manage biomedical signals. This fact causes an obvious lack of interoperability, specially suffered by end-users and healthcare system managers. Several organizations have been promoting standardization to fill this interoperability gap. The main European organization in this field is the Committee European of Normalization (CEN), specifically its Technical Committee CEN/TC251, and its countries mirrors as Spanish Normalization Association (AENOR), with its Technical Committee AEN/CTN139 (to which our research group belongs).

There are also several norms and standards for medical information interoperability that are being developed: DICOM for medical images, SCP-ECG for ECG signals intercommunication, HL7 for medical messages exchange; and ISO/IEEE 11073 [1] for MD interoperability, and CEN/ISO EN13606 [2] for Electronic Healthcare Record (EHR) exchange that are the two protocols called to solve the interoperability leak in the European context. The integration of all this plethora of standards into end-to-end solutions is still an intricate work. In this context, some private initiatives such as Integrating the Healthcare Enterprise (IHE) [3] and Continua Health Alliance [4] have emerged and are willing to collaborate with the aforementioned organizations to encourage standardization.

This paper presents the implementation guidelines for an end-to-end standard-based ubiquitous platform oriented to p-Health solutions, developed through the most recent evolutions of ISO/IEEE11073 and EN13606 and proposes a new protocol for End-to-End Standard Harmonization (E2E-SHP). In Section II the whole platform architecture is described by detailing its technical features as evolution of ISO/IEEE11073, integration with EN13606, and inclusion of E2ESHP. In Section III the proposed standard-based design and implementation guidelines are analyzed by distinguishing the specific requirements for both standards. In Section V the results of this new platform oriented to PHD are assessed and discussed for their implementation into MD by using microcontrollers. The strong and the weak points of p-Health solution are discussed as global conclusions in Section VI for its real solution transferable to the health system.

II. Platform Architecture.

One of the main challenges in the research lines for standard development and integration is its further real implementation in a telemedicine solution, transferable to the healthcare system. In last years several previous contributions have been developed for studying the viability of applying ISO/IEEE11073 (X73) in sanitary environments by implementing solutions to monitor patients in the Point-of-Care (X73PoC) [5]-[7], new approaches for Personal Health Devices (X73PHD) as proof-of-concept to test the standard evolution [8]-[9], or isolated implementations of EN13606 in healthcare systems [10]. Nevertheless, there are not European antecedents or proposals about global end-to-end solutions that integrates X73PHD and EN13606 oriented to p-Health, as it is presented in this paper.

The platform architecture (see Figure 1) is based in a Compute Engine (CE) that collects all the information acquired by different patient's MDs that define the ubiquitous and personal healthcare environment. This CE communicates, through the communication networks, with a Monitoring Server (MS) that manages the different CEs and gathers all the information arriving from each patient monitoring scenario to update the EHR. The characteristics of these different elements that form the system architecture are:

- *MDs*. The original medical data acquisition follows the vendor format (X73-compliant MDs are yet to be released into the market even though they include universal interfaces, USB or Bluetooth, the protocols are proprietary). Thus, this adapter creates the specific MD specification that generates the associate Domain Information Model (DIM) and establishes the Finite State Machine (FSM) to allow MDs acting as agents of X73 communication.
- *CEs*. Compute engine is designed as an X73-manager that recollects medical data from MDs through FSM. The information is stored in an X73-data file that, with the specific configuration profile, is the data input to the frame creation process for end-to-end protocol (E2E-SHP). E2E-SHP is a proposed protocol that allows CE (as client) and MS (as server) to guarantee the harmonized communication: supervision and remote control, platform updating and management, database access and user info monitoring, network status, intelligence of the system, etc.
- *MS*. Monitoring server is composed by two entities. The first one acts as E2ESHP server because it is in charge of receiving data from X73PHD, decoding E2ESHP frames and extracting the appropriate X73PHD data (distinguishing of the associated user' information) in order to store them in the database. The second one acts as EN13606 client/server because it implements a double function: acceptation of EN13606 queries for its further translation to mandatory fields to search them in database, and generation of EN13606 extracts following the archetype.

From this proposed platform architecture, the design guidelines and the implementation process have to guarantee several technical specifications regarding to the specific requirements for both X73PHD and EN13606 standards, as it is detailed in the next section of this paper.

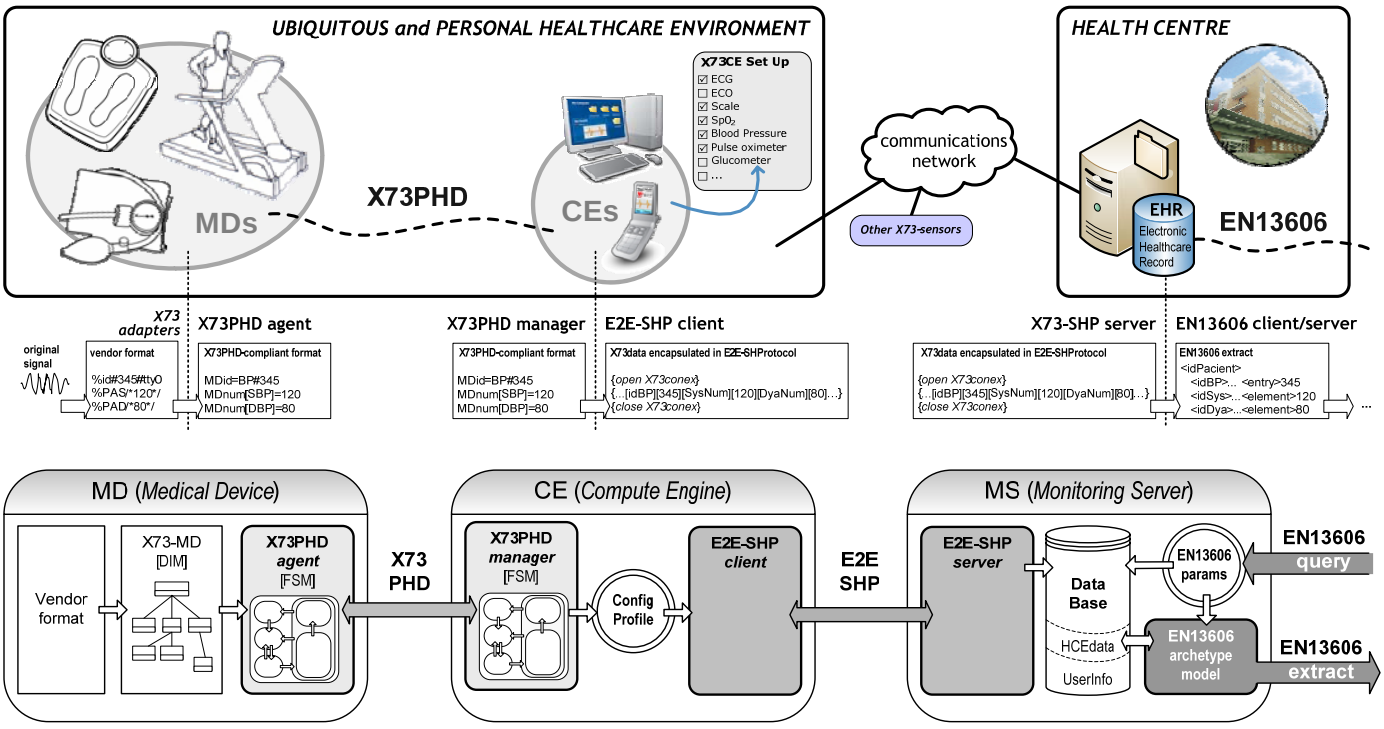

Figure 1. Implementation proposal for end-to-end standard integration in a p-Health platform

III. STANDARD-BASED DESIGN AND IMPLEMENTATION GUIDELINES

A. Standard ISO/IEEE 11073 requirements

X73PHD-based p-Health solutions rely on a communication star network topology defined by several MDs (agents) and a CE (manager). The manager will receive agent's association requests and decide either to accept them and enter into operation stage or reject them. As different transmission technologies can be used in X73PHD as well as other traffic and hardware limitation issues, the following key points should be followed in order to obtain a successful overall application:

- *MD adaptation*. Due to the lack of availability of X73PHD-compliant MDs, even though they include universal interfaces, proprietary MDs without X73PHD output are used including an X73 adapter.
- *MD integration*. The system should be able to incorporate new devices even if they make use of several transport technologies, both wired and wireless. At this point, X73PHD also defines a set of quality parameters and protocol payload that must be met, regardless of the type of transport technology used.
- *CE mobility*. The manager might be running on a mobile platform (Smartphone, PDA) in those cases where patient requires a monitoring while not being at home, like travelling, doing sports, etc. All kind of hardware and software solutions should be evaluated prior to any development. Battery autonomy, complexity of usage, stability, user's interface design, as well as programming issues, etc. are some of the facts to consider.
- *CE optimization*. It is necessary to analyze the protocol traffic efficiency so that new optimizations can be added to the protocol. Although a specific MD Encoding Rule (MDER) was designed for X73PoC, X73PHD and the additional features could require a revised version of MDER and the data type defined in ASN.1.
- *MD-CE communication.* In order to evaluate the protocol layers independently, the entire protocol stack should be implemented in both agent and manager. As X73PHD standard is relatively new and drafts have been available for a short time, new devices could still have protocol frame errors that would lead the system to a critical failure. This test-bed environment can be used as a verifying tool for new implementations with X73PHD to check the stability of the protocol, as well as timeouts, etc.

PHD devices are designed to be provided with limited hardware/software specifications and resources. This applies to both agents and managers when used in mobile solutions. Optimizing as much as possible the code is one of the development restrictions, and for that the platform reference is based on a microcontroller. This requirement being satisfied, upgrading to a higher platform with more resources available (interfaces, multimedia, and other services) should be feasible with minor changes.

Data transmission technology has to be selected based on different aspects. The signal's features should make some technologies more favorable against others, as data size, frequency, robustness against transmission errors, delivery priority, to name a few, may differ. On the other hand, wireless transmission might not be used, for instance in hospital environments without a proper EM-isolation, and wired technologies will significantly reduce the mobility of the equipment and patient/users. Nevertheless, some other real implementation issues should be taken into account as well, as some technologies may not have modules that meet the device design.

Same programming language (C++) has been used to develop the new platform like the two previous solutions. The main reasons have been:

- The use of pointers in order to easily manage object tree, efficient memory use and data frames.
- Already acquired programming skills with C/C++ and Integrated Development Environments (IDE) like Visual C++.
- Low level hardware access to control buffers and communication drivers.
- It can be used to develop solutions for embedded systems (embedded C++).
- Easily integrated into Windows application either for PC or Windows Mobile by including the Microsoft Foundation Class (MFC).

Microsoft Visual Studio C++ and Microsoft eMbedded Visual C++ have been used as Development Environments allowing to target two different platforms while keeping the code (especially the X73PHD communication library) as close as possible between them. While X73PHD library has been entirely programmed by using C++ and standard libraries, the application layer, Graphic User Interface (GUI) and other wrappers needed to access other services have been developed with the MFC libraries. These libraries can encapsulate both the protocol code and the required service functionalities in simpler classes. It also provides useful tools to create easily window-based applications adding a wide range of controls like buttons, information fields, graphics, etc. Despite of obvious graphical differences, interfaces can be created to Desktop environments as well as Mobile with these libraries.

For X73PHD protocol development, the *ISO/IEEE P11073-20601/D20 Draft Standard for Health informatics - Personal health device communication - Application profile - Optimized exchange protocol* has been followed, released on May, 2008. Although some minor changes have been included since then, none of them has been critical enough to consider some parts of the overall system to be short-time modified. In other words, it was stable enough to base a solution development on it.

Due to the amount of characteristics that the protocol features and some of them may found no reason to be applied depending on the type of device, it is desirable to list those aspects that will be necessary to implement. Nevertheless, our project attempts to develop, at the same time, a protocol implementation as close as possible to the one described in the draft. The reason of this is because the previous solution adopted a fragment of X73PoC enough for a proof of concept, but not for a multi-purpose application. Thus, in the design proposed in this paper, X73PHD FSM has been completely implemented from the beginning, as well as ASN.1 and MDER encoded data types. Other X73PHD features like *Permanent Metric* and *Enumeration* have been included later as new use cases have been brought into the application scenario.

In order to reduce the system complexity, the protocol stack and data structure models from X73PHD have been modified. While previous platforms incorporated all the layers from OSI model as well as both functions and packets needed to be processed by all of them, this time just the header is processed. Once the header is analyzed a field content chart will be followed to simplify the FSM operation. As the frame structure is known previously, it is possible to access certain information byte locations without need to read the whole packet. A comparison chart between the two approaches is shown in Figure 2.

All the data types are defined in X73PHD using ASN.1. The result is an abstract declaration of the data to be used later in the protocol, that is, name and type. Then, this complex data has to be mapped into a byte-frame in order to be manipulated by a computer and lately sent over a transmission medium. This byte-representation is achieved using MDER, a more efficient version of the classic Basic Encoding Rules (BER). Although BER, along with Packet and XML Encoding Rules (PER and XER), has available C++ compilers for conversion, MDER required some additional modifications and code arrangements to obtain C++ encoding functions. This approach raised the complexity of the previous solution, as well as memory requirements. That is the reason why in this solution direct byte-encoding technique has been used in order to provide transparency to the frame creation process as well as reduce the memory requirements (aiming microcontroller-based hardware). At the same time, special precautions have to be taken while defining the frame creation within the code to avoid critical protocol errors.

Frame processing is done in a similar way like the previous solutions, although in this case it is more atomized due to MDER features. As information fields within a frame are constant in location, like protocol-definition headers, lengths or segment indicators, measure-containing fields are updated with patient signals from the MDs.

From this, a solution design based on frame patterns is developed storing previously designed byte-patterns on memory, and filled up with the necessary parameters prior to be sent. This method will simplify the processing required on, for instance, a microcontroller as there is no need to process a new frame. In case of agents, from the received frame and based on the state-tables some of the reply-frame fields can be automatically completed from a pattern, and add finally the required information.

From previous considerations, it is desirable to perform a study of the real environment in order to develop applications and go into improving its specifications in depth. The main purpose is to discover all the standard potential applications and suggest modifications to those aspects believed to bring a better overall system's performance. A methodology followed in our work is described as follows:

- Find and analyze health services based on telemedicine.
- Enumerate, for each one of them, the MDs from the application and find out if the devices implement automatic signal acquirement.
- Study the medical services in which these devices are being used and, if use cases are defined, classify them.
- Propose, at this point, an X73 based solution. When selecting the kind of MD to be incorporated to the system, the compatibility/interoperability factor should be considered the last while other features like quality, price, versatility and efficiency should be taken into consideration first. Design objectives should be to provide the system with more reliability (reducing human error), allow the user/patient to move or make use of portable devices (ubiquitous) and simplify the set-up/replacement process in case it is necessary.

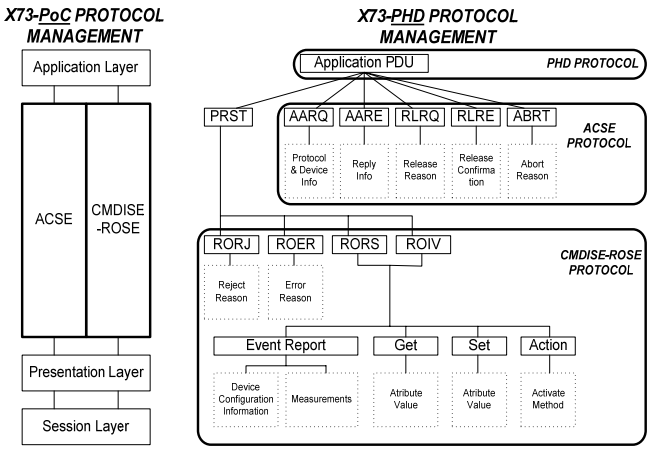

Figure 2. X73PoC and X73PHD protocol stack comparison

Following this methodology, the design of the X73PHD FSM (see Figure 3) is the key issue of any solution based on X73PHD since it defines the behavior procedure. In the design, the following states defined in X73PHD must be taken into account: DISCONNECTED, CONNECTED, UNASSOCIATED, ASSOCIATED, CONFIGURING and OPERATING. The operating procedure would be as follows:

- When both devices are turned on, the local initialization procedure is executed (MDIB and other state parameters).
- From that initialization, a connection is established through the transport layer; if it has been successfully driven, then both devices enter into CONNECTED state (but UNASSOCIATED). In order to get associated, the agent sends an association request [AARQ] to the manager.
- If the manager already knows the agent configuration, either because it is standard or because the manager has the configuration stored from previous operations, they enter into the ASSOCIATED state and they will be ready to operate. If not, then the manager will ask previously for the agent configuration (CONFIGURING) and store it for future connections (this facilitates the plug-and-play functionalities).
- In the OPERATING state, the sending of measurements begins. They can be directly sent by the agent or on demand by the manager. Under this mode, the agent can perform either a single sending or successive, during a period of time which can be limited or not.

At any time both agent and manager can get disassociated because of error situations, end of measurements, or because of other circumstances. To do so, it exists a disassociation request [RLRQ] followed by a confirmation on the other side [RLQE], or a direct disassociation [ABRT].

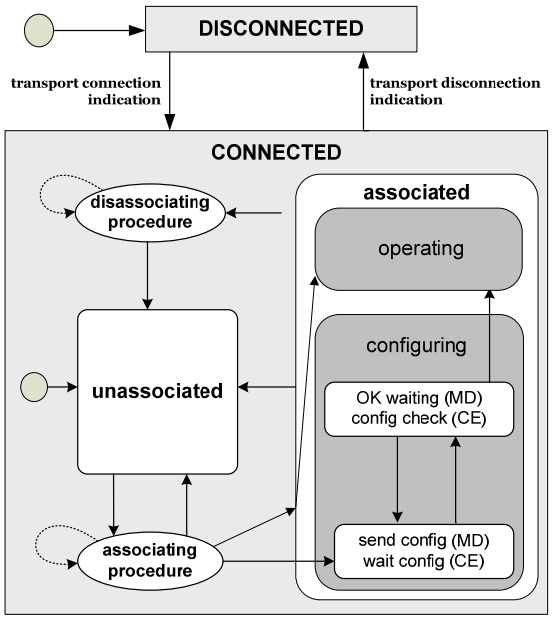

Figure 3. X73PHD Finite State Machine (FSM)

B. *Standard CEN/ISO 13606 requirements*

EN13606 standard [2] has been developed in order to represent any information included in the EHR, as well as its communication between EHR systems, managing semantic interoperability of the transmitted data. The main objective of that standard is to normalize the way EHR (the whole EHR or part of it) is interchanged to make them interoperable. Thus, EN13606 is not intended to specify the internal architecture of EHR system or the way data are stored or consulted, but the way the clinical information must be transmitted. To do that, EN13606 is based on a dual model: Reference Model, which supports information, and Archetype Model, which define "knowledge" (an archetype is a pattern that represents the specific characteristic of the clinical data). In this way, if the knowledge changes (or we need to represent additional characteristics) only the archetype under data are represented will change: for example, one patient's blood pressure would be expressed as the Reference Model but the fact of blood pressure is composed by systolic pressure and diastolic pressure is represented by the Archetype Model.

The standard is divided into 5 parts: 1-Reference Model, 2-Archetype Specification, 3-Reference Archetypes and Term lists, 4-Security, y 5-Interface Specification. Though EN13606 does not specify how data has to be stored, to transmit an interoperable EHR, we have to be able to represent several kinds of structures (see Figure 4). Briefly, those are the logic blocks the transmitted clinical data would consist of:

- *Extract*: The top-level container of part or all of the EHR of a single patient.
- *Folder*: The high level organization within an EHR (i.e.: episode of care, compartments of care, etc.)
- *Composition*: a single clinical encounter or record documentation session (Reports, test results, etc.)
- *Section*: clinical headings reflecting flow information (i.e.: Subjective symptoms, Findings, Treatment, etc.).
- *Entry*: Clinical Statements.
- *Cluster*: The means to organize nested multi-part data structures (tables, time series, etc.)
- *Element*: A container of a single data value. It is the leaf node of the hierarchy.

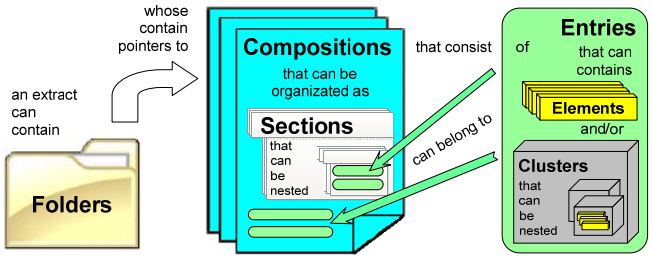

Figure 4. EN13606-compliant EHR extract structure (*published in* [2])

EN13606 standard is not completed yet. In the beginning of 2009, Part 5 had not been ratified and Reference Model has several differences with the previous version (ENV13606) of 2004. These main differences, in order to adapt the concepts that are needed to transmit in an EHR query/extract are:

- Sensitivity is not a mandatory parameter, and now it is represented by an integer. If sensitivity is not transmitted, a default value is supposed to determinate who is allowed to access that information.
- Attribute used to group a set of COMPOSITIONs (*contribution_id*) goes from AUDIT_INFO class to COMPOSITION class, so its meaning is more related to clinical information rather than context information.
- CLINICAL_SESSION class and its attributes are now included by RECORD_COMPONENT and FUNTIONAL_ROLE classes; in this way, COMPOSITION class contains now optional attributes as *session_time* (time interval of the session) and *territory* (country where the extract is created), and FUNTIONAL_ROLE class contains *healthcare_facility* (organization at which the role was performed) and *service_setting* (type of service location at which the role was performed).
- Attribute representing who is legally responsible for the care of the patient at the time COMPOSITION is committed (*hca_legally_responsible_for_care*) disappears.
- Optional attribute *composer* is replaced by a mandatory association with *committal* belonging to AUDIT INFO class. In this way, context information is higher because of it must be said who sends, when was send and from which EHR system. In addition, it can be made difference between who send it and who created it, which can in fact be different people.
- Attribute *version_specific*, which makes reference to the target of a link was a RECORD_ COMPONENT or a version of it, is deleted. Because of every version of a record component have a unique identifier, is logic to make reference to that identifier giving no importance if it is a version or not.

After studying the evolution of these fields in the standard and the inheritance between classes, the main fields to be transmitted with EHR are shown in Table I, as well as the *data types* (in brackets) and their meaning in the standard. Those fields are mandatory in transmission, but if we include additional (optional) ones that are not mandatory but takes singular relevance in telemedicine applications, other relevant information could be expressed like if the extract has been automatically generated by a machine and the doctor who authorized that transmission, if it is attached a view of the screen when the test was made, even the date and the time interval at which the item occurred.

TABLE I
EN13606 REFERENCE MODEL. RELEVANT FIELDS

EHR_ EXTRACT	Header of the container. After that, all compositions are transmitted
ehr_id [*Instance Identifier*]	The unique EHR identity (from which the EHR_EXTRACT was created for that EHR provider system for this subject of care)
ehr_system [*Instance Identifier*]	The identity of the EHR provider system from which the EHR_EXTRACT was created
rm_id [*String*]	The identity and version of the Reference Model standard under which the EHR_EXTRACT was created
subject_of_care [*Instance Identifier*]	Unique identifier of the subject of care
time_created [*Time Point*]	Date/time at which data from this subject of care's EHR was queried/exported to create the EHR_EXTRACT
EXTRACT_ CRITERIA	Optional parameters specified in the EHR Request. They are not mandatory to repeat
all_versions [*Boolean*]	Indicator if it includes all historic versions
archetype_ids [*Set Instance Id*]	Set of Archetypes if they were used as a basis of selecting data to include in the EHR_EXTRACT
max_sensitivity [*Integer*]	Used sensitivity for EHR_EXTRACT generation
multimedia_included [*Boolean*]	If multimedia data have deliberately been excluded from the EHR_EXTRACT
other_constraints [*String*]	additional criteria that were used to generate the EHR_EXTRACT
time_period [*Interval TimePoint*]	Date or time interval to which the EHR_EXTRACT is limited
RECORD_ COMPONENT	Abstract class that introduces those mandatory fields
name [*Text*]	Name of the record
rc_id [*Instance Identifier*]	Unique identifier of the record
synthesised [*Boolean*]	TRUE if this RECORD_COMPONENT was created in order to comply with this standard
COMPOSITION	Inherited attributes of RECORD_COMPONENT and a mandatory association with *commital* (from AUDIT_INFO), which contains those attributes:
committer [*Instance Identifier*]	The party responsible for committing the RECORD_COMPONENT
ehr_system [*Instance Identifier*]	EHR system in which the RECORD_COMPONENT was committed
ehr_id [*Instance Identifier*]	The unique EHR identity (from which the EHR_EXTRACT was created) in which the RECORD_COMPONENT was committed
time_committed [*Time Point*]	Date/time at which the RECORD_COMPONENT was committed within the identified EHR system
ENTRY	Inherited attributes of RECORD_COMPONENT
uncertain_expressed [*Boolean*]	If it contains data that indicates some degree of uncertainty
ITEM	Abstract class (CLUSTER and ELEMENT inherits from it). It introduces *optional* fields:
<*obs_time*>	Date and time (interval) at which the ITEM occurred
<*emphasis*>	A way of denoting that the composer wished to mark this ITEM (*optional*)
CLUSTER	Inherited attributes of RECORD_ COMPONENT, and:
structure_type [*Code SimpValue*]	Time/spatial organization of data within this CLUSTER
ELEMENT	Inherited attributes of RECORD_COMPONENT and ITEM
value [*Data Value*]	DATA_VALUE containing the value, unless this is indicated as absent by a *null_flavour* attribute

Furthermore, an EN13606 implementation would require a MS from which we would be able to generate a valid EHR extract, as shown in Figure 5. As we can see, this example contains additional fields like *meaning* to bind the measurement meaning to a clinical terminology and *archetype_id* to identify the pattern data which is transmitted after. In this example, CEN Data types (TS14796) has been used meanwhile the definition of a new data type, common to ISO, HL7 and CEN, would be carried out.

EHR_EXTRACT
ehr_system.extension = **HospitalServet**
ehr_system.assigningAuthorityName = **Salud**
ehr_system.valid_time = **1/1/1900 – 1/1/3000**
ehr_id.extension = **ExtractoHCE.120025022008**
ehr_id.assigningAuthorityName = **Salud**
ehr_id.valid_time = **1/1/1900 – 1/1/3000**
subject_of_care.extension = **441003686941**
subject_of_care.assigningAuthorityName = **Salud**
subject_of_care.valid_time = **1/1/1900 – 1/1/3000**
time_created.time = **15/02/2009 17:32**
rm_id = **EN13606-1.0**
COMPOSITION
rc_id.extension = **0003**
rc_id.assigningAuthorityName = **MiguelServet-Salud**
rc_id.valid_time = **1/1/1900 – 1/1/3000**
name = **Listado de datos de telemedicina**
sensitivity = **3**
committal.ehr_system.extension = **HospitalServet**
committal.ehr_system.assigningAuthorityName = **Salud**
committal.ehr_system.valid_time = **1/1/1900 – 1/1/3000**
*committa*l.committer.extension = **Dr. Perez**
committal.committer.assigningAuthorityName = **Salud**
committal.committer.valid_time = **1/1/1900 – 1/1/3000**
committal.time_commited = **10/01/2009 17:32**
ENTRY
rc_id.extension = **0004**
rc_id.assigningAuthorityName = **MiguelServet-Salud**
rc_id.valid_time = **1/1/1900 – 1/1/3000**
archetype_id.extension = **CENArch.Entry.TMWeightMeasure.v1**
archetype_id.assigningAuthorityName = **MiguelServet**
archetype_id.valid_time = **1/1/1900 – 1/1/3000**
name = **Medida del peso**
meaning.codingScheme = **2.16.840.1.113883.6.96**
meaning.codingSchemeName = **SNOMED**
meaning.codingSchemeVersion = **7**
meaning.codeValue = **301333006**
meaning.displayName = **Medida del peso corporal**
synthesissed = **FALSE**
sensitivity = **3**
ELEMENT
rc_id.extension = **0005**
rc_id.assigningAuthorityName = **MiguelServet-Salud**
rc_id.valid_time = **1/1/1900 – 1/1/3000**
name = **Medida del peso**
meaning.codingScheme = **2.16.840.1.113883.6.96**
meaning.codingSchemeName = **SNOMED**
meaning.codingSchemeVersion = **7**
meaning.codeValue = **301333006**
meaning.displayName = **Medida del peso corporal**
sensitivity = **Clinical**
synthesized = **FALSE**
value.PQ.value = **77**
value.PQ.units = **kg**
value.PQ.property = **Weight**

Figure 5. Example scheme of an EN13606-compliant EHR extract

IV. IMPLEMENTATION INTO MEDICAL DEVICES

Following the standard-based design and proposed development guidelines in previous section, the results of the implementation into MD are presented and discussed. As patient's data are usually collected via wearable PHDs, a key point in these MDs is ergonomics since it improves patient's quality of life. In order to improve the ergonomics, MD developers shall increase device's autonomy and decrease its weight which usually means that the board must use both batteries as small and light as possible and low capacity embedded processing systems in order to minimize the power consumption. Typical components in a board are a microcontroller, a communication module and a sensor, but sometimes the unique component in the board is a System-on-Chip (SoC) module that integrates all these components in a single chip. Once the hardware has been selected, the required software solution has to be developed. The software framework for such a task is commonly determined by the hardware involved. For example in the Bluetooth or ZigBee case, the framework is determined by the communications stack that usually provides a proprietary Real Time Operating System (RTOS) and a proprietary Application Programming Interface (API) to access the RTOS functions. Typical features of medical device hardware used in medical applications are a few Kilobytes of Random Access Memory (RAM), a few tens of Kilobytes of non-volatile solid-state memory (typically flash or Read-Only Memory, ROM), and a few Megaflops processor. All these requirements are finally translated into software implementations that have in mind to reduce processor and memory as much as possible. As far as software is concerned, this type of devices do not require a high grade of intelligence and sources code are usually written in assembler, C, or embedded C++. Operating System (OS), if used, is platform dependent and its API differs strongly from some devices to others.

This section proposes general software architecture to implement MDs. It must be noticed that X73PHD defines completely a point-to-point communication between agent and manager. That means that both syntaxes (the contents of each message, byte to byte) and semantics (the meaning of the message contents and its dynamics: what action to execute, how to respond, etc.) are defined for each transaction that usually entails any of this processes: a state change in the FSM, modifications in some of the DIM objects, and/or execution of some actions.

Based on our implementation experiences a guideline to incorporate X73PHD into MDs and how to proceed with a MD implementation based on the concept of pattern is given below. A pattern is defined as a model that can be used to produce a copy exactly equal to a reference. The concept can be applied here to interchanged messages between a specific MD and its CE. Taking an X73PHD specific type of message it can be seen that there are some sections or blocks that keep constant and unchanged during all MD life.

These similarities appear because in each transaction one or more ASN.1 structures are transmitted and its attributes are nearly always the same as they are needed only to enable interoperability and plug-and-play and do not transmit information. Blocks of bytes that do not differ can be extracted for a specific type of message, following X73PHD. We have coined each of them as pattern since they allow building X73PHD messages. All of the patterns needed to construct a message in a point-to-point communication between a MD and its CE are stored into what we call patterns library. Starting from the patterns library, each of the messages interchanged between MDs and its CE can be reproduced with a minimum of code lines. An example of the process is shown in Figure 6 in which the message of interest is filled with patterns from patterns library, and a few program variables, such as *Invoke-Id* or an *ObservedValue* (obtained, for example, from the last blood pressure measurement). Messages generated in this way can be compared with incoming ones, or transmitted.

The architecture of components for software development is proposed in Figure 7 and it is composed by the patterns library [11], the X73-kernel, the drivers, the adaptation layer, and the transports. The X73-kernel is the task that copes with pattern assembling, processing, comparison and transmission. It also manages the state of the FSM, the state of objects in the DIM, and some system signals. The signals managed include data sent or data received signals, connection established, connection lost, timer signals for scanners (such as *PeriCfgScanner*), etc. Drivers provide basic functions that depend on the MD specialization that allow the X73-kernel manipulate the hardware. The adaptation layer provides services that allow the X73-kernel managing peer-to-peer communications through the transport.

The guidelines above can be used to produce MD implementations. Each of these implementations can be shared between more than one MDs but only when both devices share the same communication profile and specialization. For example, it is possible to use the same implementation in a Bluetooth bathroom scale and a RS-232 laboratory weigh scale when both of them use polling profile (since both of them use the weigh scale specialization and the polling mode communication profile), but is not possible to use that implementation in a Bluetooth bathroom scale that use the baseline or PHD profile. These platforms usually use a common multiplatform programming language, such as C, to program the X73-kernel so it can be easily ported.

Untying these MD specialization-MD communication pairs gives great optimization in memory space. Most of the memory required by this solution is non-volatile type. Consequently, the low memory consumption makes possible to use the same X73-kernel and library of patterns in a big number of MDs that, at the same time, allows sharing it between different manufacturers improving interoperability.

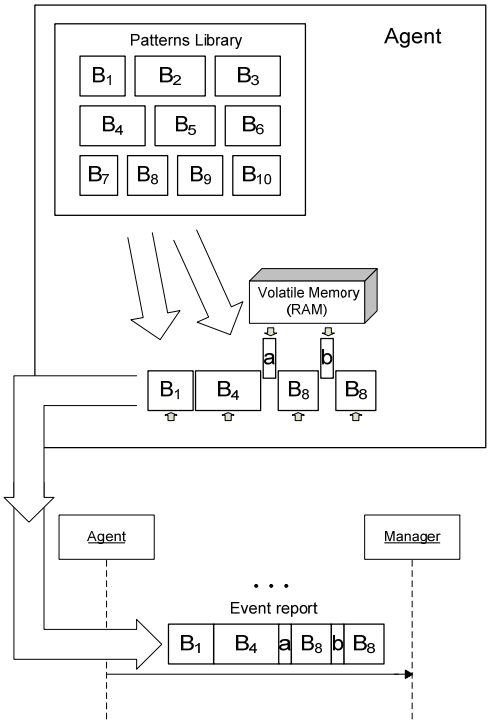

Figure 6. X73 messages synthesis from patterns

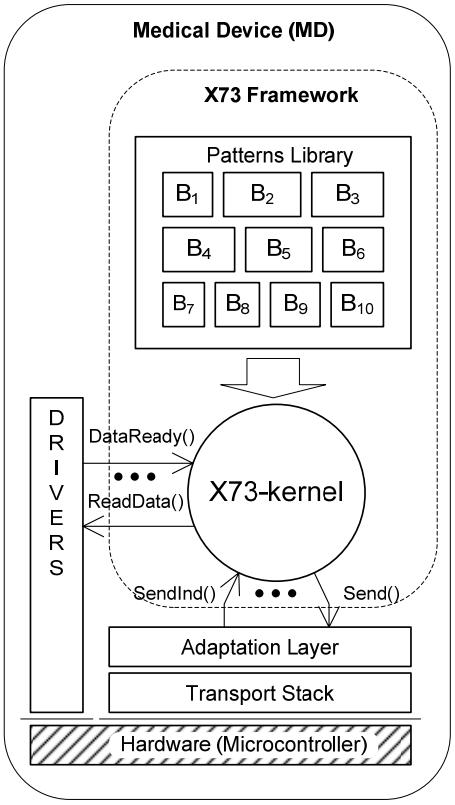

Figure 7. Architecture proposed for X73 MDs implementation

An exhaustive analysis of the X73PHD standard is required to generate the patterns library and the X73-kernel. Our proposal is depicted in Figure 8. Two-steps analysis is need: Firstly, to determine the set of messages that the device needs and the sequence they should follow, and, secondly, the analysis of this set of messages (typically a dozen) to obtain patterns. Once obtained, they are stored in the patterns library. Duplicated ones are discharged in order to reduce memory consumption. Once the patterns library has been generated, the X73-kernel that provides all the functionalities is developed using Object Oriented Analysis and Design (OOA/D) principles following a stimulus-response schema.

Once developed, an implementation can be shared and it offers a Rapid Application Development (RAD) that will allow a novel X73 implementer to develop a MD with only a basic knowledge of X73 since the unique needed modules to be developed are the adaptation layer and the device drivers. For example, in the case of a heart rate monitor, device drivers must provide with a signal that informs the X73-kernel when a new *sample-array* is ready, and a method to access the data. In order to maximize interoperability, it is preferable to create a framework that shall be used by all manufactures. To create this framework, the help of X73 world experts, experiences of other implementations, and the expertise of Special Interest Groups (SIG) is a key point.

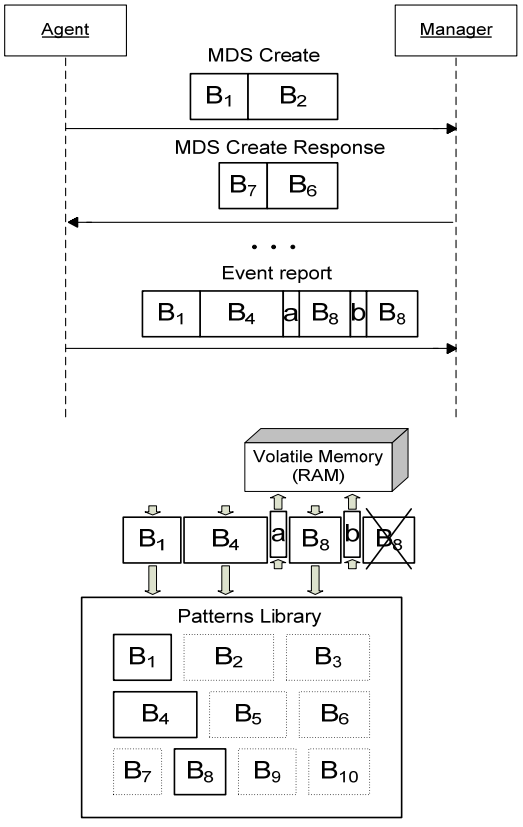

Figure 8. X73 library synthesis for microcontroller-based design

V. Conclusion

The need of interoperability and standard integration for the proposal of harmonized healthcare applications have derived to the implementation of an end-to-end standard-based platform that allows achieving a ubiquitous and personal health solution. The followed design guarantees the specific requirements for the two main standards in this context, ISO/IEEE11073 and EN13606 (both adopted as European way of medical interoperability), and opens new challenges currently under research as the implantation on micro-controllers or the development of a new protocol for end-to-end standard harmonization (E2E-SHP).

Acknowledgment

The authors wish to thank PHDWG and Melvin Reynolds, *convenor* of the CEN/TC251 WGIV, for the contributions to this research. We also appreciate the contribution of Miguel Galarraga (UPNA associate professor and researcher) and Adolfo Muñoz (researcher of the *Instituto de Salud Carlos III*, AENOR/CTN139 general manager, and CEN/ TC251 member) to the excellent results carried out during the last years in this work. This research work has been partially supported by projects TIN2008-00933/TSI and TSI2005-07068-C02-01 from *Comisión Interministerial de Ciencia y Tecnología* (CICYT) and European Regional Development Fund (ERDF), TSI-020302-2008-35/*Plan Avanza I+D* from *Ministerio de Industria, Turismo y Comercio*, and FPI grant to M. Martínez-Espronceda (Res. 1342/2006 from Public University of Navarre).

References

[1] ISO/IEEE11073.CEN/TC251-Health Informatics.www.cenorm.be - www.centc251.org. Point-of-Care MD Communication standard (X73-PoC). Personal Health Devices standard (X73-PHD). www.standards.ieee.org/ - www.ieee1073.org. Accessed April 09.

[2] EN13606-CEN/TC251. Electronic Healthcare Record (EHR) Communication. Parts 1, 2, 3, 4 and 5. http://www.medicaltech.org. Accessed April 09.

[3] Integrating the Healthcare Enterprise (IHE). http://www.ihe.net/. Accessed April 09.

[4] Continua Health Alliance. http://www.continuaalliance.org/home/. Accessed April 09.

[5] Yao J. and Warren S. Applying ISO/IEEE 11073 standards to wearable home health monitoring systems. *J Clin Monit Comput* 19(6):427-436, 2005.

[6] Warren S, Lebak J, and Yao J. Lessons learned from applying interoperability and information exchange standards to a wearable point-of-care system. *Conf Proc Transdisciplinary Conf Distr Diagn Home Healthcare*. doi: 10.1109/DDHH.2006.1624807, 2006.

[7] Clarke M, Bogia D, Hassing K, Steubesand L, Chan T, and Ayyagari D. Developing a standard for personal health devices based on 11073. *Conf Proc IEEE Eng Med Biol Soc*, pp.6175-6177, 2007.

[8] Martinez I, Fernández J, Galarraga M, Serrano L, Toledo P de, Jiménez-Fernández S, Led S, Martínez-Espronceda M, and García J. Implementation of an end-to-end standard-based patient monitoring solution. *IET Commun* 2(2):181-191, 2008.

[9] Galarraga M. *et al*. Standards for medical device communication: X73PoC-MDC. *Stud Health Technol Inform* 121, pp. 242-256, 2006.

[10] Muñoz A, Somolinos R, Pascual M, Fragua J A, Gonzalez M A, Monteagudo J L, and Salvador C. H. Proof-of-concept Design and Development of an EN13606-based Electronic Health Care Record Service. *J Am Med Inform Assoc* 14:118-129, 2007.

[11] Martinez-Espronceda M, Serrano L, Martínez I, Escayola J, Led S, Trigo J D, and García J. Implementing ISO/IEEE 11073: Proposal of two different strategic approaches. *Conf Proc IEEE Eng Med Biol Soc*, pp.1805-1808, 2008.