Optimización de una Plataforma Telemática para Monitorización de Pacientes orientada a u-Salud, y basada en Estándares y *Plug-and-Play*

I. Martínez, J. Escayola, I. Fernández de Bobadilla,
M. Martínez-Espronceda, L. Serrano, J. Trigo, S. Led, y J. García

***Resumen*— Este artículo aborda la optimización de una plataforma telemática de monitorización de pacientes basada en el estándar ISO/IEEE11073 (X73) para interoperabilidad de dispositivos médicos. Para ello, se analizan las virtudes y mejoras pendientes de actualizar en la evolución de X73, orientada a entornos ubicuos y dispositivos llevables (*Personal Health Devices*, X73-PHD), y abierta a nuevas funcionalidades *Plug-and-Play* y de gestión remota. Tras un análisis de alternativas, se detalla la migración de la plataforma para adaptarla a la nueva arquitectura requerida para dichas funcionalidades y que posibilite el desarrollo de sistemas *end-to-end* basados en estándares. Se analiza el diseño e implementación del modelo *agente-manager*, particularizado en X73-PHD al protocolo de comunicación entre un dispositivo médico (*Medical Device*, MD) y un *gateway* concentrador (*Compute Engine*, CE). Por último, se valoran los resultados obtenidos orientados a los nuevos casos de uso para entornos ubicuos y a la implantación sobre dispositivos *wireless*, objetivo clave dentro del Comité Europeo de Normalización.**

***Palabras clave*— plataforma telemática extremo-a-extremo (*end-to-end telematics platform*), protocolo de comunicación (*communication protocol*), modelo agente-manager (*agent-manager model*) estándar X73 (*X73 standards*), casos de uso (*Use Cases*), asistencia sanitaria ubicua u-Salud (*u-Health*).**

I. Nomenclatura

ACSE	*Association Control Service Element*
BER/DER	*Basic/Distinguished Encoding Rules*
PER/MDER	*Packet/Medical Devices Encoding Rules*
CE	*Computer Engine*
CMDISE	*Common MD Information Service Element*
CMIP	*Common Management Information Protocol*
DIM	*Domain Information Model*
FSM	*Finite State Machine*
MD	*Medical Device*
MDIB	*Medical Data Information Base*
MDAP/MDDL	*Medical Device Application Profile/Data Language*
ROSE	*Remote Operation Service Element*
X73-PoC/PHD	*X73-Point of Care/Personal Health Device*

Este trabajo ha recibido el apoyo de Comisión Interministerial de Ciencia y Tecnología (CICYT) y de los Fondos Europeos de Desarrollo Regional (FEDER) TSI2007-65219-C02-01 y TSI2005-07068-C02-01, del Programa de Estímulo de Transferencia de Resultados de Investigación (PETRI) PET2006-0579, y una beca FPI a M.Martínez-Espronceda (Res. 1342/2006-UPNA).

I. Martínez, J. Escayola, I. Fernández de Bobadilla, J. Trigo, S. Led, y J. García pertenecen al Grupo de Tecnologías de las Comunicaciones (GTC), Instituto de Investigación en Ingeniería de Aragón (I3A), Universidad Zaragoza (UZ), c/ María de Luna, 3, 50018 Zaragoza, Spain (correo e.: imr@unizar.es).

M. Martínez-Espronceda, L. Serrano, y S. Led pertenecen al Departamento de Ingeniería Eléctrica y Electrónica, Universidad Pública de Navarra, Campus de Arrosadía s/n, 31006 Pamplona, Spain (correo e.: miguel.martinezdeespronceda@unavarra.es).

II. Introducción

A lo largo de los años noventa los servicios de telemedicina estaban enfocados en el control de los pacientes de los hospitales, sobre todo de aquellos a los que había que controlar sus constantes vitales y se encontraban en la Unidad de Cuidados Intensivos (UCI) [1]. Los electrocardiogramas (ECG) y electroencefalogramas (EEG) requerían una atención continua del especialista y el papel del ingeniero telemático se basaba en una labor técnica que quedaba reducida tanto al soporte y mantenimiento de los equipos ante eventuales fallos, como a la renovación y actualización de los mismos. Cada fabricante realizaba un equipo propio y el hospital que utilizara dicho dispositivo quedaba sujeto a sus condiciones, lo que implicaba grandes barreras y limitaciones para avanzar en soluciones globales y homogéneas de e-Salud.

Con el fuerte impacto en los últimos años que ha supuesto el despegue de las nuevas tecnologías, los servicios de e-Salud vieron la importancia de no quedarse atrás y supieron aprovechar la coyuntura tecnológica. Con la evolución hacia la e-Salud es el propio especialista, los equipos, la tecnología, etc. la que se desplaza alrededor del paciente que se coloca en el centro del sistema sanitario. Así surgen las tecnologías de computación ubicua, a través de las redes de área personal y corporal (*Personal/Body Area Network*, PAN/BAN) [2].

Ante esta nueva situación, la interoperabilidad de equipos de distintos fabricantes a través de la estandarización de protocolos, se convierte en un requisito básico para avanzar hacia la telemedicina ubicua: u-Salud [3]-[4]. Este es un largo proceso que ya viene impulsado en las últimas décadas desde varias organizaciones dedicadas a la estandarización: Health Level 7 (HL7) [5], OpenEHR [6] y el Comité Europeo de Estandarización (CEN) [7] a través de su Comité Técnico 251 (TC251) [8] que se encarga de la informática médica y desde el que se colabora en el desarrollo de los nuevos estándares que son objeto de estudio en este artículo: la norma EN13606 [9] para gestión del Historial Clínico Electrónico (HCE), así como la familia de normas ISO/IEEE11073 (X73) que en su primera versión X73-PoC [10], para interoperabilidad de dispositivos médicos en el punto de cuidado (*Point-Of-Care*, PoC); y en su reciente evolución X73-PHD [11], reorientada para cubrir también comunicaciones de dispositivos médicos llevables (*wearables*) y con funciones *Plug-and-Play* (P&P) en entornos personales (*Personal Health Devices*, PHD).

Existen contribuciones previas [12]-[14], desarrolladas en EE.UU. por el grupo de investigación del Dr. Warren, que estudian la viabilidad de implantar estándares en entornos sanitarios e implementan plataformas similares de monitorización de pacientes en el PoC. Sin embargo, no existían antecedentes europeos en este campo ni tampoco propuestas de soluciones telemáticas globales extremo a extremo que introducen nuevos casos de uso de monitorización de pacientes en entornos ubicuos y que estén orientadas a ser compatibles con la nueva versión de la norma X73-PHD, como se plantea en este artículo.

Así, y a partir de desarrollos anteriormente publicados [15]-[16] y del trabajo presentado en la anterior edición de Jitel [17] en los que se presentaba una primera implementación basada en estándares (X73 y EN13606) extremo a extremo, en este artículo se avanza en la optimización de dicha plataforma. La nueva arquitectura ha de dar solución, en primer lugar, a los contextos ubicuos establecidos por la norma X73-PHD. Pero, en segundo lugar, su diseño ha de permitir la evolución de X73-PoC a X73-PHD, incorporando los cambios en el modelo de comunicación *agente-manager*, la definición de la máquina de estados finita, y las nuevas capas de transporte y de nivel físico. Además, se debe adaptar a las futuras actualizaciones de la norma, por lo que el nuevo diseño propuesto ha de estar adaptado a las futuras necesidades y permitir a la par la investigación, la experimentación de los más recientes estándares de tecnología médica, y su demostración en un entorno de pruebas integrado.

Por todo ello, es necesaria una arquitectura que no sólo posibilite la interoperabilidad entre dispositivos médicos en el punto de cuidado, sino que garantice su portabilidad a otros entornos, situaciones, y casos de uso (servicios geriátricos y de rehabilitación, seguimiento de atletas o autocontrol personal de la salud, escenarios móviles, etc.), facilite la gestión remota (información médica, alarmas, patrones de comportamiento, etc.), e incorpore nuevas funcionalidades (*P&P*, *HotSwap*, etc.), tecnologías (Bluetooth, ZigBee, RFID), y dispositivos (PDAs, *SmartPhones*, microcontroladores, etc.).

En la Sección III se analiza la evolución que ha seguido la norma X73 en los últimos años, así como su estado actual y previsiones futuras. A partir de este análisis y justificándose en el mismo, se realiza el planteamiento de la transición del sistema existente *plataforma1.0-αlfa* (basada en SO Linux y centrado en la compatibilidad con X73-PoC) hacia una nueva *plataforma1.5-βeta* (basada en SO Windows y orientada a la nueva versión del estándar X73-PHD). Se debaten los puntos fuertes y débiles de ambos sistemas, y las nuevas posibilidades que introduce la migración. En la Sección IV se expone el diseño planteado, se describe la arquitectura completa de la nueva plataforma detallando sus características técnicas, y se desarrolla la implementación específica seguida. La Sección V valora y discute los resultados de esta nueva implementación orientada a X73-PHD y analiza la incorporación de las nuevas funcionalidades en lo que constituirá la versión final de este trabajo de cara a convertirse en una solución transferible al sistema de salud: la *plataforma2.0-release*. Las conclusiones globales del trabajo se discuten en la Sección VI.

III. Evolución de X73 y de la plataforma

Las evoluciones sufridas por X73 en los últimos años y sus implicaciones en las funcionalidades a incluir en la plataforma se analizan a continuación para poder plantear el nuevo diseño.

A. *Evolución de X73: de X73-PoC a X73-PHD*

Este trabajo se basa en el modelo que define la norma X73. Como toda norma, desde el comienzo de su desarrollo (en el que multitud de ingenieros han trabajado en paralelo con universidades, instituciones y entidades internacionales) hasta la actualidad, ha sufrido un proceso evolutivo. La norma nace desde la necesidad de profundizar en la interconexión entre el punto de cuidado (PoC) y los proveedores de servicio con tecnologías *middleware* que proporcionen interoperabilidad para la comunicación entre los diversos dispositivos médicos (*Medical Devices*, MDs) de fabricantes distintos. Este es el origen de X73-PoC [10], considerada como la vía europea de estandarización desde el Comité Técnico TC251/CEN.

El posterior desarrollo de nuevos MDs *wearables*, con sensores de alta calidad y sobre tecnologías *wireless* (como Bluetooth o ZigBee), y el incremento de accesos de banda ancha a redes multimedia, aceleró la evolución del estándar X73 hacia una versión optimizada y adecuada a estas nuevas tecnologías: X73-PHD. Así se llega a la situación actual, en la que la norma está evolucionando más rápido, como se recoge en los continuos avances del documento "*Draft Standard for Health informatics* IEEEP11073 - 20601 TM/D16" [11].

Con esta evolución cambia la arquitectura del protocolo y, por consiguiente, cualquier sistema o solución basada en X73. No sólo esto, si no que la adopción de X73-PHD involucra la toma de decisiones de evolución de soluciones X73-PoC migrando hacia X73-PHD, justificación central de este trabajo.

En términos generales, X73-PHD posibilita la conexión entre MDs y sistemas externos (*Compute Engines*, CE) como extensión del denominado *gateway* en X73-PoC. Estos CEs en el contexto de X73-PHD abren el abanico de nuevos casos de uso a los que X73 debe dar respuesta: entrenadores personales, medicina deportiva, atención personalizada de pacientes crónicos, escenarios móviles con dispositivos *wireless*, etc.

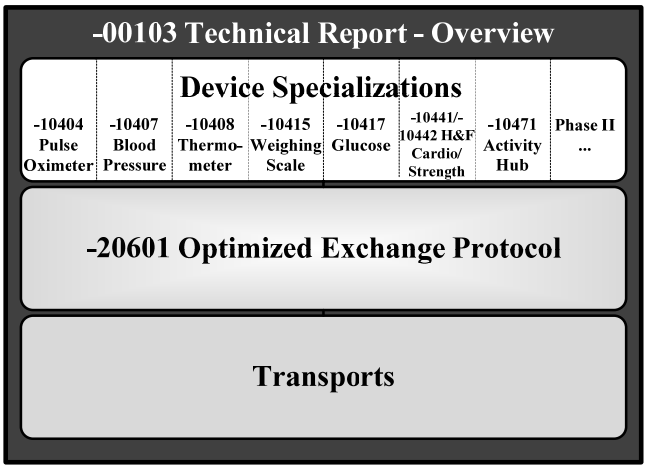

Fig. 1. Mapa de la pila del protocolo X73-PHD

Estos escenarios son posibles gracias a la nueva arquitectura de la norma X73-PHD dividida en tres niveles (ver Fig. 1):

- ***Device Specialization***. Un conjunto de modelos descriptivos que aglutina el total de objetos y atributos relacionados con los componentes de los dispositivos, como la configuración global del sistema (*Medical Device System*, MDS), la métrica persistente (*PM-Store*) o las especificaciones de la métrica.
- ***Optimized Exchange Protocol***. La parte principal del estándar. Consiste en un marco de terminología médica y técnica (*Domain Information Model*, DIM) que se encapsulará dentro de una trama (*Protocol Data Unit*, PDU). La versión previa de X73 definía esta parte como *Medical Device Data Language* (MDDL). Después un Modelo de Servicio define un conjunto de mensajes e instrucciones para obtener datos del agente basado en el DIM. Además, proporciona una conversión de datos de Sintaxis de Notación Abstracta (*Abstract Syntax Notation*, ASN.1) a una sintaxis de transferencia usando reglas de codificación (*Encoding Rules*, ER) optimizadas denominadas *Medical Device Encoding Rules* (MDER) además de reglas estándar de codificación básicas (*Basic* ER, BER) e incluso más soporte efectivo con las reglas de codificación de paquetes (*Packet* ER, PER). Los elementos de servicio (*Service Element, SE*) de la plataforma anterior que siguen siendo válidos son: *Remote Operation* (ROSE, optimizado para MDER), *Association Control* (ACSE) y *Common Management Information* (CMISE). Finalmente, el Modelo de Comunicación describe una conexión punto a punto basada en una arquitectura *agente-manager* a través de una máquina finita de estados (*Finite State Machine*, FSM).
- ***Transport Layer***. La transmisión de datos es independiente de la tecnología de transporte debido a que X73-PHD establece suposiciones que requieren un soporte directo por esta capa, para permitir que varios tipos de transporte puedan ser implementados. Por tanto, las especificaciones del tipo de capa de transporte quedan fuera del alcance de X73-PHD.

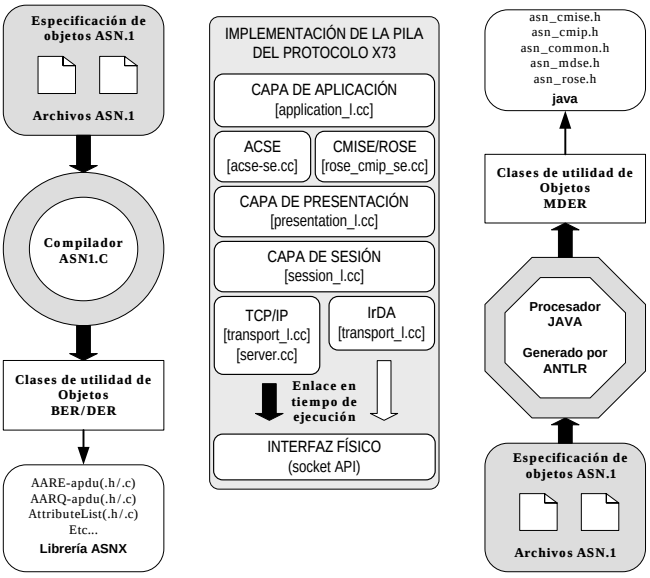

Fig. 2. Esquema de diseño y herramientas empleadas en las plataformas X73

B. *Evolución de la plataforma: de 1.0-αlfa a 1.5-βeta*

Los detalles técnicos de *plataforma1.0-αlfa* se recogen en [17]. Fundamentalmente consiste en una solución *end-to-end* dividida en dos sistemas: uno que permite la conexión vía X73-PoC entre MDs y un elemento central (*gateway*), y otro que soporta el almacenamiento de la información médica en un servidor de HCE según el estándar EN13606. Dado que las evoluciones las ha sufrido la norma X73, la migración a *plataforma1.5-βeta* se ha centrado en el primer sistema.

Este sistema se implementó en un conjunto de archivos, agrupados en librerías, escritos en lenguajes C/C++ y Java, o generados a través de compiladores ASN1.C. Opera bajo SO Linux y/o la herramienta CYGWIN (estándar de programación POSIX/GNU GCC 3.4.4), que permite simular la consola de Linux en computadoras con SO Windows. Para la arquitectura de la pila de protocolos X73-PoC (ACSE, ROSE, CMISE) se usaba, por un lado, la sintaxis abstracta fijada por X73-PoC en MDDL (es decir, qué conjunto de mensajes se van a intercambiar) y, por otro lado, la sintaxis de transferencia fijada en MDER y simplificada respecto a BER/DER (es decir, cómo van codificados los mensajes). Un esquema de este diseño se muestra en Fig. 2.

Este primer diseño fue de alta utilidad ya que constituye el primer desarrollo completo *end-to-end* totalmente compatible con X73 y EN13606. Las estructuras y métodos de clases C/C++ encajaban perfectamente con las especificaciones X73-PoC y, dada la complejidad de la norma, C/C++ nativo de las máquinas con SO Linux (*open source* y de disponibilidad masiva en red), aporta sencillez y utilidad óptimas para un primer diseño, además de ser el único SO con código fuente abierto (*opensource*) para RS-232 e IrDA (que a su vez son los únicos interfaces de capa física reconocidos en X73-PoC).

Sin embargo, la vertiginosa evolución de X73 y sus constantes actualizaciones dentro del CEN, obligaron a una rápida migración (ver Fig. 3) debido a los siguientes motivos:

- La evolución de los casos de uso de sistemas fijos a móviles, y la evolución hacia conexiones P&P o *HotSwap*, hace que el nivel físico IrDA/RS-232 deba evolucionar hacia otros sistemas como USB, Bluetooth, y RFID, entre otros.

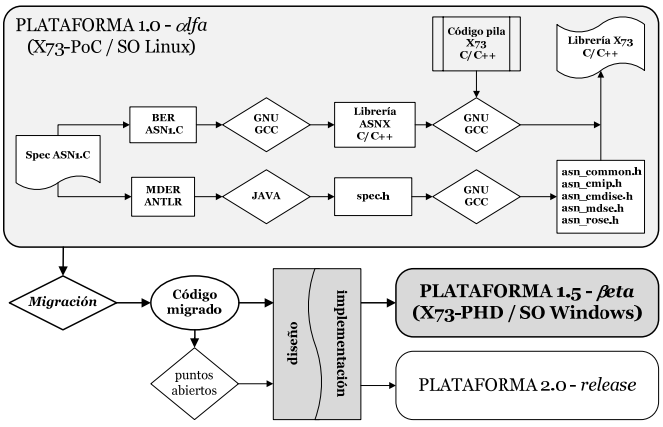

Fig. 3. Esquema de evolución de las plataformas X73

- Los nuevos MDs, como consecuencia de la evolución de X73-PoC a X73-PHD, involucran un cambio sustancial en la forma de comunicación con CE, lo que implica un nuevo diseño de la FSM. Esta nueva FSM es demasiado compleja para implementarse en *plataforma1.0* ya que se debe buscar un nuevo sentido a los métodos que implementa FSM.
- La nueva norma X73-PHD no especifica un diseño concreto a nivel de transporte. Sin embargo, *plataforma1.0* presentaba un diseño específico complicado para aportar modularidad.
- La evolución del adaptador X73 hacia un microcontrolador, la inminente implementación de CEs a dispositivos *wireless* (PDAs, *SmartPhones*, etc.), la transición de escenarios (fijos a ubicuos) y de usuarios (hospital a pacientes en su domicilio o móviles), y la necesidad de aportar usabilidad a través de interfaces gráficos (*Graphic User Interfaces*, GUIs), exigen un encapsulado de alto nivel más simple del que ofrece *plataforma1.0* (demasiado compleja y opaca a bajo nivel).

A partir de estas premisas, se antoja evidente una evolución de *plataforma1.0* que debe pasar por un proceso de migración que conduzca a un nuevo código (sobre SO Windows) y el consecuente diseño e implementación de *plataforma2.0*. Por migración se entiende tanto la traducción de código C/C++, (ambos entornos no comparten las mismas especificaciones C/C++) como la adaptación del código a las librerías propias del SO Windows y sus interfaces (*Application Programming Interface*, API). Esta migración conlleva modificaciones menores (una función cambia de nombre pero realiza la misma tarea, o una cabecera de inclusión no existe en SO Windows o no implementa clases que se implementaron en su equivalente sobre SO Linux) y modificaciones mayores (aquellas partes del código no compatibles con el compilador de Visual Studio C++ que hacen imposible su portabilidad y que implican supresión de cabeceras e inclusión de otras, inclusión de nuevas definiciones de constantes manifiestas, modificación de procesos, *threads* y *sockets*, gestión de memoria, o creación de zonas adicionales de compilación condicional). Un esquema global de este proceso se muestra en Fig. 3.

Con estas consideraciones, el código resultante se engloba en tres carpetas principales:

- *X73lib1.5*, implementa la pila X73, y contiene los archivos de cada una de las capas: transporte, sesión, presentación, ACSE, ROSE, CMISE, implementación del objeto DIM, carpetas BER, MDER, y especificaciones ASN.1 (*libasnx*).
- *X73adaptador1.5*, que implementa el MD (en este caso, un tensiómetro) y el adaptador de su código propietario a X73.
- *X73gateway1.5*, que implementa dos comunicaciones, la de CE con MD y la de CE con el servidor de telemonitorización.

Este código mantiene la filosofía de comunicación entre MD (adaptador) y CE (*gateway*): CE pide asociación a MD, lo que desencadena la creación de un objeto MDIB actualizado con los datos adquiridos del tensiómetro (según perfil periódico-*baseline* o episódico-*polling*). MD envía la estructura del objeto a CE, que copia en una base de datos actualizable y, en su caso, envía al servidor de telemonitorización. Desde esta base se van a analizar los requisitos que establecerán las reglas de diseño para realizar la nueva implementación.

IV. Diseño e Implementación de la Solución Propuesta

Tras el análisis técnico por el que se justifica una migración de la plataforma, se describe a continuación la arquitectura, reglas de diseño, y desarrollo de implementación de la solución.

A. Arquitectura de la migración. Reglas de diseño

La nueva arquitectura que se proponga ha de cubrir una serie de compromisos principales o pautas de diseño.

Primero, buscar un diseño funcional y conforme a X73. Para ello, se ha de eliminar la dependencia con la tecnología de transporte buscando una solución genérica y configurable (denominada gestor de capa de transporte o *handler*). Así, los datos adquiridos, primero se transforman a X73 actualizándose cada vez que hay una nueva medida (pero en modo episódico, no periódico como en la anterior *plataforma1.0*) para, después, iniciar el envío de datos (ya conforme a X73) del adaptador X73 al CE a petición del usuario. Por tanto, se deja al desarrollador la inclusión de los archivos que den soporte a la tecnología de transporte que estime oportuno para cada MD.

Segundo, el diseño de pila genérica X73 debe mantenerse. Sin embargo, se ha de optimizar el código e introducir la librería de definiciones ASN1.C, llamada ASNX en X73, para que enlace correctamente en la vinculación que se realiza desde el entorno de desarrollo para todos los recursos. Las clases que son una abstracción de los previos *adaptador* y *gateway* también han de modificarse para implementar la nueva máquina de estados definida por CEN para X73-PHD.

Por último, al no incorporar una tecnología de transporte concreta, los datos que se encapsulan a través de las distintas capas llegan a una disposición en estructuras de *buffers*, que recogen el conjunto de PDUs de las distintas capas de la pila. Estos *buffers* han de ser correctamente gestionados para que responda al protocolo de comunicaciones que marca la norma.

Toda esta definición de arquitectura del sistema a partir de las reglas de diseño establecidas para la nueva versión, conducen al siguiente apartado que desglosa la implementación técnica y funcional que se ha seguido en la *plataforma 1.5*.

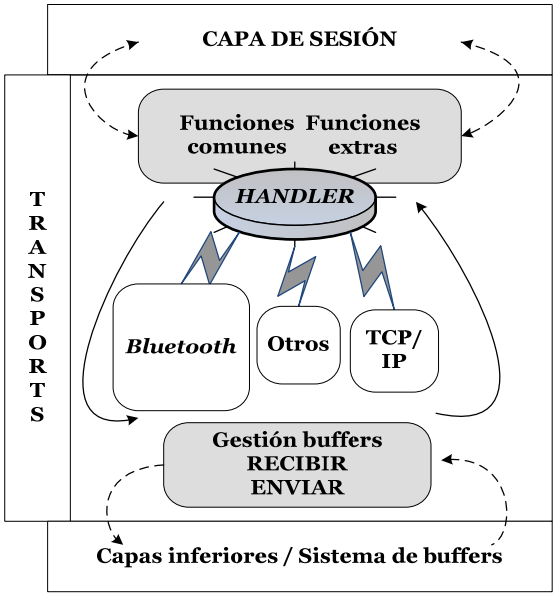

Fig. 4. Mapa de la pila del protocolo X73-PHD

B. *Implementación de la solución*

1) Gestor de capa de transporte (handler)

Ante la necesidad de X73-PHD de una capa genérica de transporte, la comunicación entre ambos extremos a dicho nivel es responsabilidad del desarrollador. Para ello, en la *plataforma1.5* se incluye la nueva capa genérica TRANSPORTS, como muestra Fig. 4. La comunicación con la capa de sesión se realiza a través del interfaz genérico de pila (*stack*) para cada extremo (MD y CE), y con la capa física a través de un sistema de *buffers*, como se detalla en apartados siguientes. La previa *plataforma1.0* establecía un enlace en tiempo de ejecución (externo al programa principal *main*, tanto de MD como de CE) para llamar al archivo de transporte sobre IrDA o TCP/IP. Al eliminar la dependencia, *main* queda esperando peticiones de conexión, independiente-mente del protocolo de transporte asociado. Esta función se implementa mediante un interfaz genérico gestor de capa de transporte (*handler*), transparente al protocolo implementado.

A nivel de implementación, este *handler* es autosuficiente puesto que solo hace de enlace entre la capa de sesión y la capa física a través del sistema de *buffers*. En caso de querer introducir un protocolo de transporte determinado, se debería recompilar el *software*, y enlazarlo con la aplicación a través de *transport_[nameProt]_l.cc* y *transport_[nameProt]_l.h*. Además, habría que enlazarlo con el interfaz de pila, a través de un archivo tipo *transport_if.h*.

Este *handler* es un objeto C++ que implementa los métodos más importantes de una interfaz de transporte (ver Tabla I). Además se le han incorporado otros métodos adicionales (ver Tabla II) que dan solución a ciertos requerimientos:

- Averiguar el perfil de comunicación: episódico (*polling*) o periódico (*baseline*)
- Determinar si el protocolo de transporte que se implemente está soportado o no por el *manager* de la comunicación (CE)
- Manejo de estructuras *st_buffer* recibidas por parte del interfaz del sistema de *buffers* y posterior envío al método *handle_data_ind*.

TABLA I. MÉTODOS BÁSICOS DE UN INTERFAZ GENÉRICO DE TRANSPORTE

MÉTODO	ACCIÓN
handler_t_con_req	Envío de petición de conexión
handler_t_send_req	Envío de datos
handler_t_dis_req	Envío de petición de desconexión
handler_n_con_ind	Indicación de conexión
handler_data_ind	Recepción de datos/mensajes
handler_n_dis_ind	Indicación de desconexión

TABLA II. MÉTODOS ADICIONALES DEL *HANDLER* DE TRANSPORTE

MÉTODO	ACCIÓN
scanner_com_profile	Tipo de perfil de comunicación
scanner_transport_tecnology	Soporte de protocolo de transporte
buffer_received	Estructura *st_buffer* recibida (MD)
buffer_received_gw	Estructura *st_buffer* recibida (MD)

2) Maquina de estados finita FSM

La plataforma en todas sus versiones se caracteriza por ser conforme a X73, siguiendo con meridiano rigor el diseño de las nuevas máquinas de estados (FSM) de comunicación para los extremos del sistema: agente (MD) y manager (CE). La filosofía de diseño de esas FSM (ver Fig. 5) ha sido clave en el diseño junto con la implementación de los estados por los que atraviesa la comunicación MD/CE en las dos pilas del modelo.

Para diseñar estas máquinas FSM se ha creado en el programa principal un bucle continuo en que se implementan los estados indicados en Fig. 5: DESCONECTADO, CONECTADO, DESASOCIADO, ASOCIADO, CONFIGURANDO, y OPERANDO. El proceso de funcionamiento sería, a grandes rasgos, como sigue:

- Los extremos están inicialmente apagados, por lo tanto antes de establecer una conexión, deben inicializarse o encenderse.
- A partir de este momento y en virtud a las respectivas FSM de cada uno de ellos se intentará establecer una conexión a través de la capa de transporte; si tiene éxito, hará que ambos entren al estado CONECTADO, pero estén no asociados. Para asociarse MD deberá pedir una petición de asociación a CE.
- Si CE sabe la configuración de MD, directamente quedan ASOCIADOS y podrán operar. Si no, CE acepta asociarse con MD pero necesitará configurar y guardar varios parámetros en base de datos para evitar hacerlo más veces (esto facilita la función P&P). Si la versión de MD o los protocolos de capa de transporte o física, no son soportados por CE, no es posible asociarse con MD, y la comunicación se interrumpe.
- En el estado OPERANDO, se realiza el proceso normal de toma de datos: codificación a X73, y envío del MD al CE.
- En cualquier momento del estado ASOCIADO se puede querer desasociar, voluntaria o involuntariamente cualquier par. Voluntariamente pasan a desasociarse e involuntariamente se envía un mensaje de abortar comunicación.
- Así mismo durante el período de tiempo que dura la conexión entre ambos a nivel de transporte, pueden desconectarse, también voluntariamente por el final de la comunicación o involuntariamente por algún fallo.

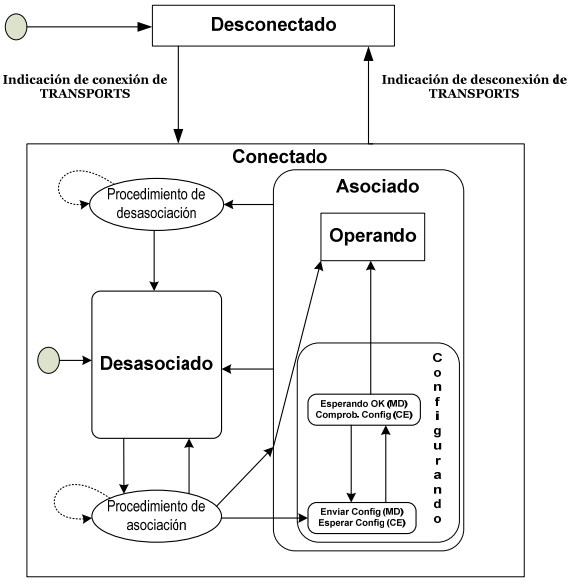

Fig. 5. Máquina de estados FSM genérica de MD y CE en *plataforma1.5*

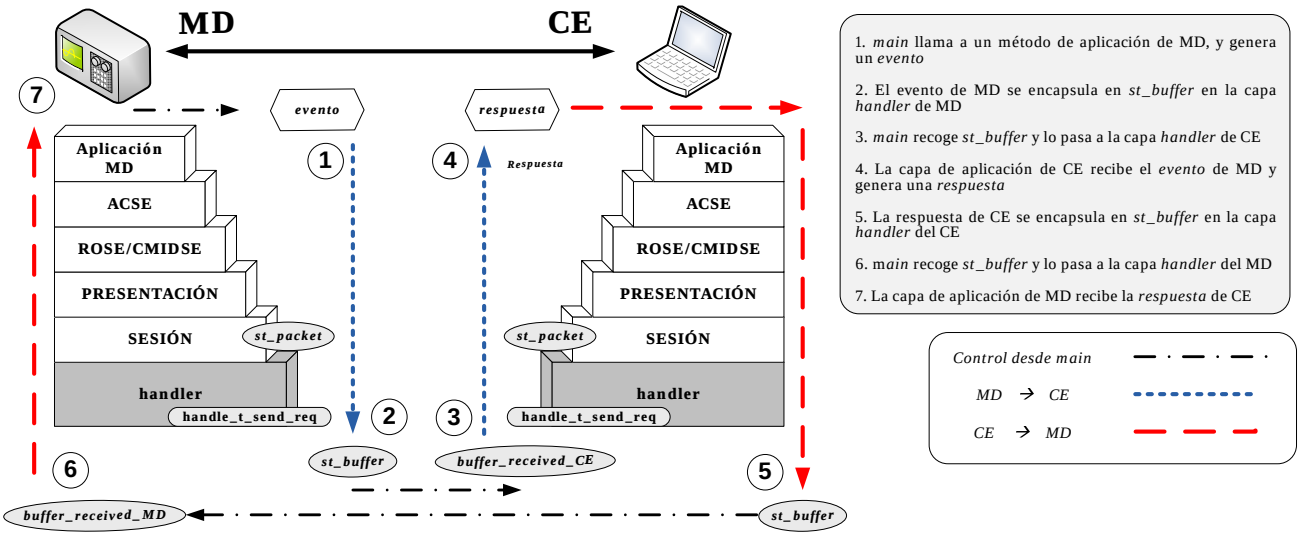

Fig. 6. Modelo de comunicación entre MD y CE a través del sistema de *buffers*

3) Sistema de buffers. Modelo de comunicación MD-CE

De la misma manera que en la capa de transporte, se ha dejado la libertad de implementar cualquier capa física. Esta capa física se encarga de recibir y enviar las estructuras *st_buffer* que se envían MD y CE a través del *handler*. La estructura *st_buffer* es un contenedor de datos en memoria. Los bits que se codifican en cada PDU de cada capa siguen esta estructura de *st_buffer*, así son más fáciles de manejar. A su vez, un conjunto de *st_buffer* se puede agrupar en *st_packet*.

Como se muestra en Fig.6, el modelo de comunicación a través del sistema de *buffers* sigue el siguiente proceso:

- Desde *main* invocamos un método que genere un evento en la capa de aplicación de MD (1). Este evento, es un mensaje para CE que se va encapsulando capa tras capa hasta llegar a sesión donde forma un *st_packet*. Este *st_packet* se encapsula en otra PDU que la capa del *handler* utiliza en forma de *st_buffer* (2). Este *st_buffer* contiene el mensaje que MD ha enviado a CE y que se ha ido encapsulando.
- El control lo recupera *main* que recoge, como parámetro devuelto, ese *st_buffer* por ser un sistema de dos pilas.
- Se encapsula *st_buffer* de MD en un *buffer* para CE (3). Ese *buffer* atraviesa las capas de la arquitectura del CE *down-up*, invocando la capa *handler* de CE. *st_buffer* se desencapsula hasta llegar a la capa de aplicación que lo lee y responde (4).
- Esa respuesta de CE a MD recorre la pila desde la capa de aplicación a *handler* y genera un *st_buffer* de respuesta que se recoge desde la llamada previa de *main* (5).
- El mensaje de respuesta llega a MD (6), pasa por *handler*, sube hacia capa de aplicación y genera otro mensaje (evento o respuesta), comenzando el proceso desde el principio (7).

Una clave de esta versión *plataforma1.5* es que se pueden controlar los bits *on the wire* (es decir, los bits que forman el *st_buffer* y que se pasan ambas pilas a través de su interfaz). De esta manera se permite realizar control de flujo y control de errores desde el programa principal, objetivo perseguido desde el CEN y que posibilitará gestión de datos y alarmas.

Como resumen de todo lo anterior, se desglosa la estructura final del código completo que constituye *plataforma1.5*:

- Primero, los archivos que forman la base de la aplicación:
  - *mainX73pila.cpp*: archivo con el programa principal donde se ha implementado la máquina de estados FSM.
  - *utilidades_globales.h*: archivo de inclusión en el que se ha diseñado un conjunto de funcionalidades (sistemas gráficos de consola, funciones de test, sistema de I/O para menús, utilidades de audio, estructuras y variables, constantes manifiestas, etc.) para ejecución del programa principal.
  - *application_l.cc* y *.h*: archivos que implementan la capa de aplicación del adaptador X73. Incluyen métodos nuevos para: inicializar la estructura MDIB en la nueva FSM [*inicializar*], actualizar medida X73 [*tomar_medida*], gestión de *st_buffer* [*get_first_buffer*, *get_last_buffer*], etc.
  - *application_bcc.cc* y *.h*: archivos que implementan la capa de aplicación del CE X73, con numerosos métodos nuevos especialmente los asociados con el *handler* de transporte.
- Segundo, la librería X73 [*X73lib1.5*], que incluye:
  - *Pila X73*: con los archivos **.cc* y **.h* que implementan la pila X73; en especial, se ha creado el *handler* de transporte [*transport_handler_l.h* y *.cc*, y *transport_handler_if.h*], e interfaces específicos de cada pila [*stackMD*] y [*stackCE*].
  - *Librería ASNX*: con archivos/cabeceras de implementación de las especificaciones ASN usadas para la comunicación BER/DER (migradas de SO Linux a SO Windows).
- Finalmente, la carpeta *adquisidor* que contiene:
  - *Diseño del interfaz de MD*: a partir del diseño de los archivos [*tensiómetro.cpp* y *.h*] se implementa un interfaz que lee los datos del tensiómetro. El *software* sobre C++ estándar y C++ nativo del API de Windows, permite directamente extender esta implementación a nuevos MDs.
  - *Software USB*: para usar los puertos USB asociados al interfaz que provee el tensiómetro para leer la conexión de datos se usan los archivos [*USTlib.h* y *USFlib.h*], con sus librerías dinámicas, *.dll*, enlazadas en tiempo de ejecución. Esto es replicable fácilmente para otros interfaces físicos.

V. Resultados y puntos abiertos

Tras la implementación de *plataforma1.5* no sólo se ha conseguido la migración requerida por la evolución de X73 sino que se ha desarrollado una aplicación que, en sí misma, constituye un demostrador de la comunicación conforme a X73-PHD. Además, la inminente *plataforma2.0-release* incluyendo los puntos abiertos que se plantean a continuación, permitirá convertirla en una solución transferible al sistema sanitario.

A. *Demostrador X73-PHD*

La plataforma implementada constituye un útil y eficaz demostrador X73-PHD. Este demostrador comienza preguntando al usuario qué dispositivo MD desea usar de una lista disponible (ver Fig. 7). Tras la elección del MD (aunque sólo está incluido el tensiómetro, en breve se prevé implementar pulsioxímetro, báscula, y adquisidor de ECG), se muestra información del tensiómetro, y se informa al usuario de las mediciones que se le tomarán: las lecturas de presión y las pulsaciones por minuto.

A continuación se muestra el menú de control de FSM que atraviesa los extremos de comunicación (MD-CE), y que está basada en X73-PHD. El usuario puede desplazarse a cualquier estado pero, según la FSM, para llegar a uno se ha tenido que pasar por todos los anteriores, como muestra Fig. 8.

A partir de aquí, MD se inicializa, se crean las capas de las pilas y los interfaces de funcionamiento (*stacks*). Por otro lado en MD, se crea la estructura MDIB: objeto MDS, VMD y las subramas del árbol. Posteriormente, se pregunta por el sistema de transporte que soporta la comunicación, preparando el *handler* para soportar los correspondientes protocolos. Además, en pantalla se muestran líneas de ejecución del programa que ayudan al ingeniero a conocer los métodos de las capas que se atraviesan. También se enseña cómo los *buffers* mandan la información X73-PHD y demás parámetros de configuración, correspondientes a los eventos y respuestas intercambiados entre MD y CE.

Fig. 7. Demostrador X73-PHD: elección de dispositivo médico

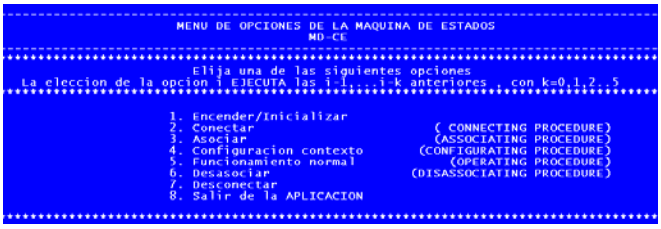

Fig. 8. Demostrador X73-PHD: modelo de comunicación según FSM

Tras asociarse, MD entra al punto de configuración en que CE le envía el objeto MDS, todavía sin medidas de lecturas tomadas del tensiómetro. En CE se crea un contexto de recepción de datos (episódico, dadas las características del tensiómetro). Así, MD queda listo para la toma de medidas (siempre a petición del usuario) entrando en el estado OPERANDO de la FSM. Al usuario se le pregunta si desea tomarse alguna medida. Si responde que sí, se informa de los valores que deberían resultar de la medición. Así el sistema está listo para adquirir. Pulsando el correspondiente botón en el tensiómetro, el dispositivo tomará automáticamente la tensión, sonando al acabar un pitido informativo de que hay datos disponibles porque las medidas han sido leídas. Tras asegurarse de que los datos leídos son los correctos a enviar a CE, se muestra un menú (ver Fig. 9) con la posibilidad de enviar o no datos.

Al enviar datos, MD actualiza con las medidas tomadas el objeto MDS, y lo envía a CE, para que también lo actualice. Se muestran las medidas recibidas detallando las identificaciones conforme a X73 (19230, 19229 y 18442) correspondientes al tensiómetro (presión diastólica, presión sistólica y pulso, respect.), como muestra Fig.10. A partir de aquí se informa si van a tomarse más medidas o, por el contrario, se pasa a un menú para desasociar MD y CE o desconectarlos como indica FSM de X73-PHD.

Los proceso de desasociación y desconexión son equivalentes. Una entidad (MD o CE) elige desasociarse. La aplicación pregunta por cuál quiere hacerlo, como corresponde a un sistema real y comercial. Tras seleccionar la entidad, se establece un tiempo máximo (*timeout*) configurable. Pasado ese tiempo, si ninguna entidad ha procedido, implica que ambas se desasocian pero sin entendimiento, lo que desemboca en una situación de error de sistema. Para evitarlo, transcurrido el *timeout* quien había iniciado el proceso de desasociación, lanza un mensaje de *abort*, que será confirmado por su par, concluyendo el proceso.

Finalmente se pregunta por la opción de volver a asociar dispositivos, desconectarlos o salir definitivamente del demostrador X73-PHD, lo que será notificado mediante un tono de cierre que confirme la finalización de la ejecución.

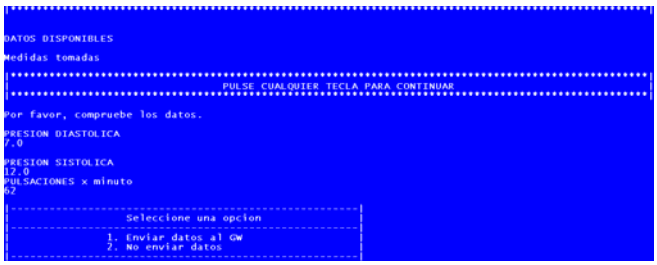

Fig. 9. Demostrador X73-PHD: toma de medidas desde un tensiómetro

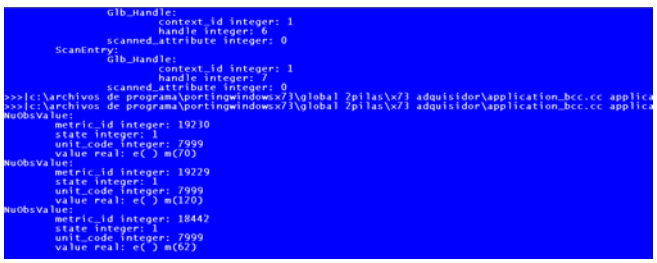

Fig. 10. Demostrador X73-PHD: recepción de medidas X73 de CE desde MD

B. Puntos abiertos: plataforma2.0-release

Como se ha comentado, quedan algunos avances técnicos para conformar la versión definitiva de la plataforma que han de implementarse de forma inmediata. Dichos puntos abiertos son:

- Implementar cada pila por separado en un microcontrolador (adaptador dentro de MD) y en un dispositivo *wireless* (CE), considerando sus correspondientes tecnologías de transporte y nivel físico en el *handler* o gestor de conexiones. Por ejemplo, al conectar MDs con interfaces físicos diferentes (uno USB y el otro Bluetooth) a un mismo CE, el *handler* ha de comunicarse con ambos MDs independientemente de los protocolos de transporte. A la arquitectura diseñada para el *handler* con los parámetros y métodos implementados, habrá que añadir el código completo de los diferentes perfiles de comunicación para poder tramitar en tiempo real los correspondientes interfaces de acceso a cada servicio. Para ello, quedan por diseñar más métodos exclusivos (privados) mediante *sockets* y *threads* del *handler* que den robustez y mejoren su gestión. Este diseño podría completarse con la implementación de modelos de prioridades para atender ciertos MDs (en fase de consideración por el CEN).
- Soportar la conexión de múltiples MDs a uno o varios CEs, optimizando la creación y gestión de los diversos MDIBs, e implementando un gestor de estados de la FSM tal que lea los parámetros de configuración de MD y los incorpore a una base de datos para garantizar las funcionalidades P&P. Esto enlaza con el punto anterior desde la perspectiva de X73-PHD. Esta problemática, planteada en el CEN, analizaba que, ante la lentitud de estandarización de la norma, debía existir un adaptador/concentrador de MDs que aglutinará la gestión de cada MD. De esta forma, por ejemplo, si un MD se actualiza, este concentrador podría conectarse a la web del fabricante, descargarse las actualizaciones oportunas, y enviar los parámetros correctos al MD creando una base de datos para gestionar el árbol MDIB de cada MD. Para equipos comerciales X73-compatibles, esta idea se sustituye por implementar el adaptador en microcontroladores dentro de cada MD, mientras que se incorpora como función del CE para gestión de los MDs actuales no conformes X73.
- Migrar el interfaz de consola a un GUI interactivo y adaptable al tipo de dispositivo (miniPC, teléfono móvil, *SmartPhone*, PDA, etc.), de forma transparente, sencilla y usable. Una solución comercial requiere sustituir la aplicación de consola (que si bien puede resultar completa y útil para un demostrador, no es atractiva para su uso extendido) por un sistema multimedia basado en ventanas y configurable. Así, el uso en *plataforma1.5-βeta* de IDE *Visual Studio* permitirá fácilmente en *plataforma2.0-release* diseñar aplicaciones Windows, basadas en sus propias GUIs. Para ello, se dispone de *Microsoft Foundation Classes* (MFC, que permiten diseñar desde botones, formularios, etc., hasta una verdadera infraestructura gráfica). A largo plazo, se prevén interfaces basados en Java, .Net, o Web 2.0, adaptados a las necesidades de cada caso de uso y SO específico (*Windows Mobile*, *Android*, *Symbian*, etc.) según el tipo de dispositivo.

VI. Conclusiones

La evolución de X73-PoC a X73-PHD ha conducido a una optimización de la plataforma *end-to-end* robusta, flexible, y cuyo diseño permitirá, en su inmediata migración, completar una solución *Plug-and-Play* basada en estándares, transferible al sistema de salud para la monitorización personal de pacientes en entornos ubicuos. Además, se ha diseñado completamente conforme a la norma, y soporte de cualquier tecnología a nivel de transporte y físico. Por último, el sistema constituye un eficaz demostrador X73-PHD como entorno de pruebas para los retos que se debaten dentro del CEN: control de flujo y errores, gestión de errores y alarmas, conexión de múltiples MDs a uno o varios CEs, o implantación en micro-controladores, dispositivos *wearables* o equipos *wireless*.

Agradecimientos

Los autores quieren agradecer las contribuciones a este trabajo desde el X73-PHD Working Group del Comité Europeo de Normalización y especialmente a Mr. Melvin Reynolds, *convenor* del CEN TC251 WGIV; así como a Miguel Galarraga, por sus excelentes contribuciones a esta investigación durante los últimos años, y también a Adolfo Muñoz (Instituto de Salud Carlos III), Paula de Toledo (Univ. Carlos III), y Silvia Jiménez (Univ. Politécnica de Madrid).

Referencias

[1] T. P. Clemmer, "Computers in the ICU: Where we started and where we are now," *Journal of Critical Care*, vol. 19, pp. 201-207, 2004.

[2] R. Kling, "Learning about IT and social change: The contribution of social informatics," Information Society, vol. 16, pp. 217-232, 2000.

[3] W. W. Stead, R. A. Miller, M. A. Musen and W. R. Hersh, "Integration and beyond: Linking information from disparate sources and into workflow," *J Ame Med Inf Assoc*, vol. 7, pp. 135-146, 2000.

[4] S. Pedersen and W. Hasselbring, "Interoperability for information systems among the health service providers based on medical standards," *Informatik - Forschung Und Entwicklung*, vol. 18, pp. 174-188, 2004.

[5] HL7. Health Level 7 - IEEE interoperability JWG. http://www.ieee1073.org/jwg/hl7ieeeinterop.html. Última visita: 04/08.

[6] Open EHR. Disponible en: http://www.openehr.org/. Última visita: 04/08.

[7] CEN. Comité de Normalización. http://www.cenorm.be/.Última visita: 04/08.

[8] CEN/Comité TécnicoTC251. http://www.centc251.org/.Última visita: 04/08.

[9] ENV13606. "Electronic Healthcare Record Communication. Parts 1, 2, 3 and 4, Pre-standard, 2000," http://www.medicaltech.org. Última visita: 04/08.

[10] ISO/IEEE11073 Point-of-Care Medical Device Communication standard (X73-PoC). Health informatics. [Part 1. Medical Device Data Language (MDDL)] [Part 2. Medical Device Application Profiles (MDAP)] [Part 3. Transport and Physical Layers]. http://www.ieee1073.org. See also the previous standards: IEEE13734-VITAL and ENV13735-INTERMED of CEN/TC251, http://www.medicaltech.orgh. Última visita: 04/08.

[11] ISO/IEEE11073 - Personal Health Devices standard (X73-PHD). Health informatics. [P11073-00103. Technical report - Overview] [P11073-104xx. Device specializations] [P11073-20601. Application profile - Optimized exchange protocol]. IEEE Standards Association webpage: http://standards.ieee.org/. Última visita: 04/08.

[12] S. Warren, R.L. Craft, R.C. Parks, L. K. Gallagher, R. J. Garcia and D. R. Funkrouser, "Proposed information architecture for telehealth system interoperability," *Annual International Conference of IEEE Engineering in Medicine and Biology*, vol. 2, pp. 702, 1999.

[13] J. Yao and S. Warren, "Applying ISO/IEEE 11073 standards to wearable home health monitoring systems," *Journal of Clinical Monitoring and Computing*, vol.19, pp.427-36, 2005.

[14] J. W. Lebak, J. Yao and S. Warren, "Implementation of a Standards-Based Pulse Oximeter on a Wearable, Embedded Platform," *IEEE Engineering in Medicine and Biology*, vol. 4, pp. 3196-3198, 2003.

[15] M. Galarraga, L. Serrano, I. Martinez and P. de Toledo, "Standards for Medical Device Communication: X73-PoC," *Stud. Health Technol. Inform.*, vol. 121, pp. 242-256, 2006.

[16] I. Martínez, J. Fernández, M. Galarraga, L. Serrano, P. de Toledo and J. García, "Implementation of an End-to-End Standards-based Patient Monitoring Solution," *IET Communications - Special Issue on Telemedicine and e-Health Communication Systems*, vol. 2, pp. 181-191, 2008.

[17] I. Martínez, J. Fernández, M. Galarraga, L. Serrano, P. de Toledo, J. García, "Implementación Integrada de una Plataforma Telemática Basada en Estándares para Monitorización de Pacientes," *Jornadas de Ingeniería Telemática JITEL*, pp. 505-512, 2007.