# HEARTFAID

# D19 – Prototype of data acquisition and transmission infrastructure

**Submission date: 14/09/2007**
**Due date of document: 01/08/2007**

# HEARTFAID

## A KNOWLEDGE BASED PLATFORM OF SERVICES FOR SUPPORTING MEDICAL-CLINICAL MANAGEMENT OF THE HEART FAILURE WITHIN THE ELDERLY POPULATION

| **Project summary** | |
|---|---|
| **Project acronym:** | HEARTFAID |
| **Project identifier:** | IST – 2005 – 027107 |
| **Duration of the Project:** | 01/02/2006 – 31/01/2009 |
| **Project Co-ordinator:** | UNICAL University of Calabria (Italy) |
| **Thematic Priority:** | Information Society Technology |
| **Instrument:** | Specific Targeted Research or Innovation Project |

**Consortium**

- UNICAL - Università della Calabria (Italy)
- UNICZ - Università degli studi  Magna Graecia di Catanzaro (Italy)
- UNIMIB - Università degli studi di Milano Bicocca (Italy)
- JUMC - Jagiellonian University Medical College (Poland)
- VMWS - Virtual Medical World Solutions Ltd (United Kingdom)
- FORTHNET S. A.- Hellenic Telecommunications and Telematic Applications Company S. A. (Greece)
- SYNAP - Synapsis s.r.l. (Italy)
- CNR - Consiglio Nazionale delle Ricerche (Italy)
- FORTH - Foundation for Research and Technology Hellas (Greece)
- RBI - Rudjer Boskovic Institute (Croatia)
- AUXOL - Istituto Auxologico Italiano (Italy)

# D19 – Prototype of Data acquisition and transmission infrastructure

| **Document summary** | |
|---|---|
| **Document title:** | D19 – Prototype of Data acquisition and transmission infrastructure |
| **Document classification:** | Derivable D19 |
| **Dissemination level:** | PU |
| **Submission date:** | 14 September 2007 |
| **Due date:** | 01 August 2007 |
| **Authors:** | Christos Biniaris – VMWS<br>Marianna Dikaiakou – Forthnet<br>Manolis Stratakis - Forthnet<br>Stelios Louloudakis – Forthnet<br>Sergio Di Bona – SYNAP<br>Andrzej Kononowicz – JUMC<br>Franco Chiarugi - FORTH |
| **Work package:** | WP2 – BIOMEDICAL DATA IDENTIFICATION AND COLLECTION |
| **Report version:** | 2.0 |

**Short description**

This document describes the functionalities of the HEARTFAID data acquisition and transmission infrastructure after the completion of the milestone MS2.2 "Technological infrastructure for the acquisition and transmission of the relevant biomedical data".

**Change record**

| **Version number** | Changes | Release date |
|---|---|---|
| **0.1** | Template | 10/07/2007 |
| **0.2** | Preliminary Draft | 23/07/2007 |
| **0.3** | First Draft | 08/08/2007 |
| **1.0** | Advanced Draft | 31/08/2007 |
| **1.5** | Final Draft | 07/09/2007 |
| **2.0** | Final Version | 10/09/2007 |

# Table of contents

## Executive Summary

The main objective of the HEARTFAID project is to devise, develop and validate an advanced and innovative technological platform of services and end-user applications aiming to contribute towards the optimization of the clinical management of HF and the reduction of the economic and social costs, by collecting, integrating and processing all types of the above mentioned biomedical data and information. In particular, the early detection of HF related signs and symptoms and the appropriate identification and acquisition of biomedical data from myocardial tissue and organ, may contribute to delay the hospitalization and to improve both the quality of life and survival in pathologic patients.

The identification and acquisition of the relevant biomedical data in the scope of the HEARTFAID project takes place in WP2 – Biomedical Data Identification and Collection and Task 2.2 of this WP is related to the Design and Development of the Data Acquisition and Transmission Infrastructure, which will enable the biomedical data acquisition as well as the transmission of these data to the HEARTFAID platform.

The biomedical data Acquisition and Transmission infrastructure, therefore, provides the required mechanisms in order

- to acquire the biomedical data from the relevant sensors in all the identified healthcare environments and
- to transmit these data to the HEARTFAID platform

The Data Acquisition and Transmission Infrastructure comprises a set of elements in order to fulfill the data acquisition and transmission process. These elements range from sensors which will perform the measurements, to communication modules in order to communicate the measurements to other elements of the infrastructure, devices to collect the data, software modules, APIs, data tansmission modules and data storage mechanisms.

The purpose of the current Deliverable is to describe the functionality of these elements based on the requirements and specifications defined in D14.

More specifically, the organisation of the document is the following:

After the introductory section (Section 2), an overview of the prototype is presented in Section 3, aiming to demonstrate the elements of the infrastructure at a glance. In Section 4, these elements are presented in detail, starting from the description of the medical devices in all which produce the biomedical data. Next, the data acquisition applications are presented in detail. These applications are executed on the data acquisition devices in all the identified healthcare environments in order to communicate with the medical devices and to acquire the data from them. After the data acquisition phase, the data transmission mechanisms and systems are presented including the description of the XML messaging schema for the transmission of the biomedical data, the functionality of

the intermediate gateway used in the on-the-move scenario in order to generate the XML messages based on the data received from mobile devices over wireless cellular networks and finally the alert notification system which is responsible to dispatch alert messages via email and SMS. In Section 5 the testing of the prototype is demonstrated while in Section 6 the limitations of the prototype compared to the final infrastructure are identified. The deliverable closes with the concluding remarks in Section 7.

# 1 Glossary of terms

| TERM | DEFINITION |
|---|---|
| **3G** | Third generation mobile network |
| **ABP** | Ambulatory Blood Pressure |
| **AmI** | Ambient Intelligence |
| **API** | Application Programming Interface |
| **ASCII** | American Standard Code for Information Interchange |
| **ASPX** | Microsoft Active Server Pages 2.0 |
| **BTSPP** | Bluetooth Serial Port Profile |
| **CDA** | Clinical Document Architecture |
| **DBP** | Diastolic Blood Pressure |
| **DICOM** | Digital Imaging and Communications in Medicine |
| **EAV** | Entity-Attribute-Value schema |
| **ebXML** | Electronic Business using eXtensible Markup Language |
| **ECG** | Electrocardiogram |
| **eCRF** | Electronic Case Report Form |
| **EHR** | Electronic Healthcare Record |
| **GPRS** | General Packet Radio Service |
| **GUI** | Graphical User Interface |
| **HF** | Heart Failure |
| **HL7** | Healthcare Level 7 |
| **HR** | Heart Rate |
| **HTTP** | HyperText Transfer Protocol |
| **HTTPS** | HTTP over Secure Socket Layer |
| **IHE** | Integrating Healthcare Environments |
| **IP** | Internet Protocol |
| **IR** | Infrared |
| **ISO** | International Standardisation Organisation |
| **J2EE** | Java 2 Enterprise Edition |
| **J2ME** | Java 2 Micro Edition |
| **JAR** | Java ARchive |
| **JDK** | Java Developer's Kit |
| **JRE** | Java Runtime Environment |
| **JSP** | Java Server PAges |
| **LOINC** | Logical Observation Identifiers Names and Codes |
| **MVC** | Model-View-Controller (MVC) architectural design pattern |
| **OS** | Operating System |
| **PC** | Personal Computer |
| **QRS** | A structure on the ECG that corresponds to the depolarization of the ventricles |
| **RDA** | Remote Data Acquisition |
| **RDBMS** | Relational database management system |

| | |
|---|---|
| **RR** | R-to-R Interval (the cycle between two consecutive R-waves) |
| **SBP** | Systolic Blood Pressure |
| **SCP-ECG** | Standard communications protocol for computer assisted electrocardiography |
| **SMS** | Small Messaging Service |
| **SMTP** | Simple Mail Transfer protocol |
| **SNOMED-CT** | Systematized NOmenclature of MEDicine Clinical Terms |
| **SOAP** | Simple Object Access Protocol |
| **SSL** | Secure Socket Layer |
| **TCP** | Transport Control Protocol |
| **TLS** | Transport Layer Security |
| **UML** | Unified Modeling Language |
| **USB** | Universal Serial Bus |
| **UUID** | Universally Unique Identifier |
| **WWW** | World Wide Web |
| **XDS** | Cross-Enterprise Document Sharing |
| **XML** | Extensible Markup Language |
| **XSD** | XML Schema Definition |
| **XSLT** | Extensible Stylesheet Language |

# 2 Introduction

As stated in the executive summary, the purpose of this document is to describe the functionality of these elements that comprise the Data Acquisition and Transmission Infrastructure, based on the requirements and specifications defined in D14.

The infrastructure will facilitate the collection of biomedical data from the HEARTFAID patients in all the identified healthcare environments and subsequently the transmission of these data to the HEARTFAID platform.

According to the requirements and specifications defined in D14, the devices, components and sub-systems required to achieve these goals can be outlined as follows:

i) the medical sensors
ii) the data acquisition devices and applications
iii) the data transmission mechanisms
iv) the data storage mechanisms

In this document, those building elements are initially described in detail and their features and characteristics that play major role for their integration into the Data Acquisition and Transmission Infrastructure are hilighted.

Regarding the medical sensors and devices, a complete list of devices is presented, based on the device inventory creation activity. This list contains connectable and not connectable devices. The connectable devices support some means of connectivity (e.g. via the implementation of their communication protocol or via the export of their data into some storage area in a known format), while the non connectable devices do not support connectivity, consequently they can be integrated only manually (via the manual completion of the measurements into web forms and the transmission of the data to the platform). For every connectable device, the device characteristics (environment of use, connectivity etc), the measured parameters as well as the format of the output data are described in detail, demonstrating in this way the device integration on the HEARTFAID platform.

Next, the data acquisition applications are described in detail. These applications acquire the data created by the devices in each identified healthcare environment or entered manually (using the Electronic Case Report Form application). Four data acquisition applications have been developed for the identified healthcare environments (clinical, home and on-the-move). Regarding the clinical environment two applications were developed, one application is related to the collection of data from the medical devcies (executed on a PC), while the second (eCRF application) is a web application accessed via a network connected computer. In the home environment, the application is destined to be executed on a PC, while in the on-the-move environment the application is executed in a mobile device carried by the user.

Regarding the data transmission mechanisms, the adopted XML messaging scheme for the data integration of the medical devices and sensors to the platform is described. This messaging scheme provides a flexible and extensible integration strategy. Also, considering the on-the-move scenario, as it is mentioned in D14, although the selected smart phones may have the ability to pack the acquired data into XML messages and transmit them, due to performance issues an alternative solution will be followed according to which the collected data are sent to an intermediate gateway, where the XML messages are generated and finally sent to HEARTFAID platform. The functionality of this gateway is also described in detail. Since the dispatch of alert is considered as data transmission process related closely to the measurements that enter the platform, a description of the alert notification system is also given. Finally, the Electronic Patient Record is presented, which enables the reception of data by the platform and their proper storage in order to be accessible by other components of the platform.

After having described in detail the elements of the infrastructure, some tests of these elements are presented. It has to be noticed that apart of these tests, a preliminary demonstration of the data acquisition and transmission infrastructure showing an end-to-end data integration from the devices to the platform as well as alert dispatching was carried out during the STAB meeting in Zagreb, Croatia 21-23 June, 2007.

Finally, some limitations of the prototype compare to what is envisaged as the final infrastructure are presented. The main limitations are mainly related to the complete integration of sensors, secure data transmission via HTTPS, as well as the integration of subjective remarks that describe the state of the patient (e.g. Minnessota questionnaire). Furthermore the XML interface of the eCRF, is still in its early development stages in the current prototype and intensive collaboration among project partners in order to make the integration smooth is needed.

## 3 Overview of the prototype

The prototype of the data acquisition and transmission infrastructure is shown in the following figure.

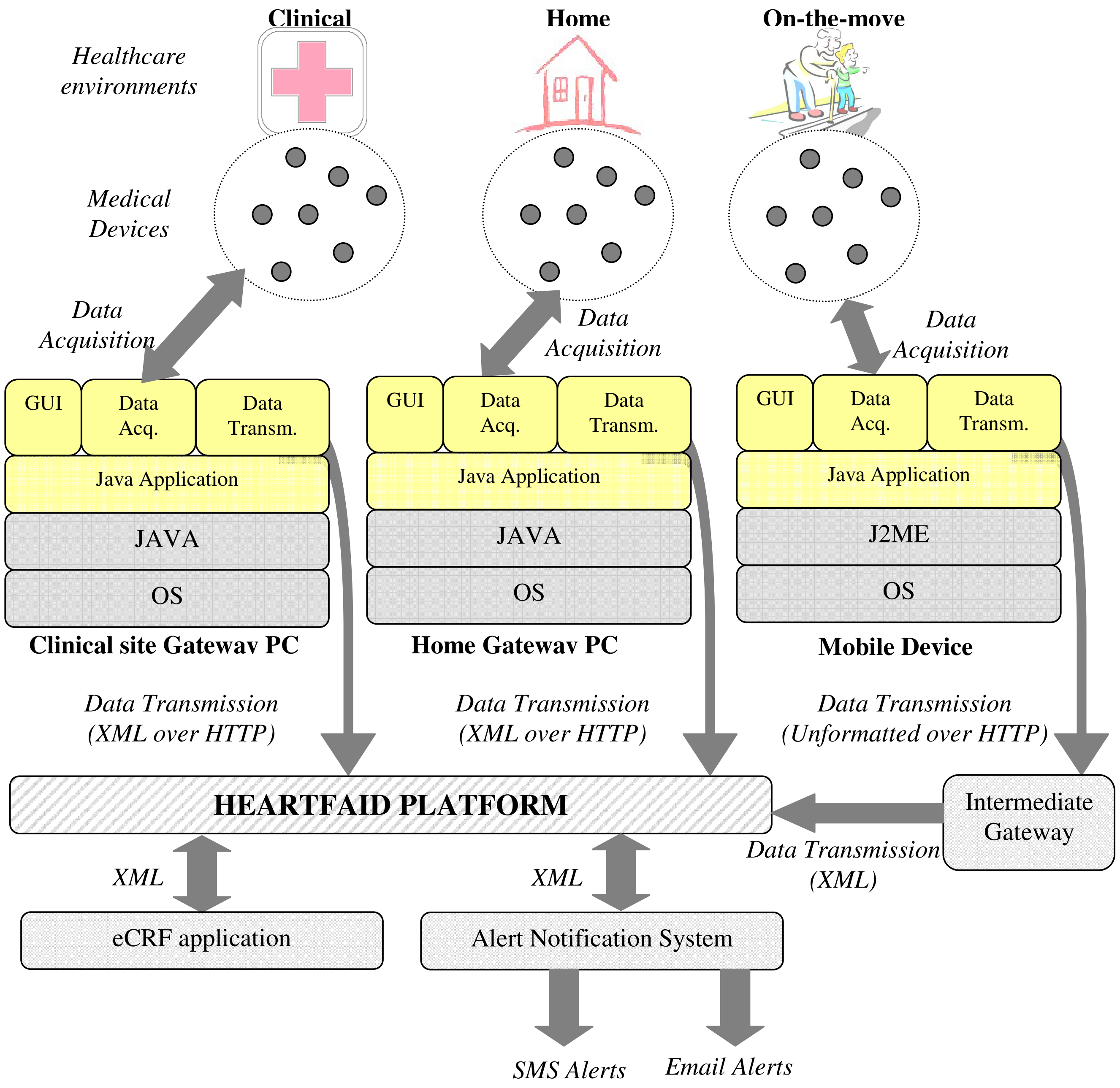


**Figure 1: Overview of the prototype of the Data Acquisition and Transmission Infrastructure**

At the top of this figure, one can see the three identified healthcare environments, namely the clinical, the home and the on-the-move environment. The first one represents the hospitals and the primary health centers, while the other two others form the so-called personal environment. In each one of the environments the medical devices and sensors provide the biomedical data. These sensors transmit these data to the respective data acquisition devices via the data acquisition process depending on the characteristics, communication features and connectivity of each device. In the Clinical and Home environment, the data acquisition devices are PCs located at the clinical and home site respectively, while in the on-the-move environment the data acquisition device is a mobile device carried by the user.
The applications responsible to acquire the data from the medical devices and transmit them to the platform were built depending on the characteristics of the data acquisition device. For the clinical and home environments Java applications were developed, while for the on-the-move environment the application was developed in J2ME a lightweight version of Java executed in mobile devices with limited memory and processing power capabilities.
Each of the applications has three main blocks, the GUI used for the interaction with the user, the Data Acquisition module which performs the acquisition of the biomedical data (raw data or data stored in files – depending on the characteristics and communication capabilities of each given device) and the Data Transmission module, which is responsible to transmit the acquired data to the HEARTFAID platform.
Furthermore, the eCRF application is a web application, used by medical personnel only (nurses, caregivers and doctors) due to the fact that not all medical devices can be automatically integrated with the platform as it has been described in D19/4.2.1.1. Hence, the manual instertion of information to the platform takes place vie the eCRF.
In the clinical and home environment the transmission of data takes place over HTTP by forming HTTP requests to the platform, containing XML as messages as body, while in the on-the-move environment (in order to reduce the application memory footprint and communication volume) the XML messages are created by an intermediate gateway, which receives the data from the mobile via HTTP request and constructs the actual XML message to be sent over HTTP to the platform.
Depending on the values of the measurements, several alarms may be raised after the processing of these data by the platform. If an alarm is fired in the middleware, the notification process is triggered. The appropriate message is assembled in the middleware platform, including the measurements that caused the alarm and other assistive data. The message may be sent by SMS or e-mail, depending on the means selected by the patient or the care-taker.
The SMS notification module, developed in work package 6, offers a clear-cut XML-based communication interface, through which authorized clients may submit requests. The whole process is fully automated and commences with the reception of a well-formed XML document, containing the message subject, the

message body and the message recipients. The SMS notification module manages the XML unmarshaling, the SMS marshaling and the dispatch and monitoring of the SMS. When the message arrives to its destination, the patient, care-taker or clinician is able to view the informative description on their mobile phone quickly and easily, and proceed to perform the necessary actions.

# 4 Elements of the prototype

## 4.1 Medical Devices

### 4.1.1 Introduction

During the STAB meeting in Pisa, it was decided to construct an inventory of the medical devices (useful for the HEARTFAID project) already present in the clinical partners' departments. The construction of this inventory, started in November 2006, was run till May 2007 with continuous exchange of information among clinical partners and technical partners.
Furthermore, based on the list of data & signs (D9), a selection of new devices to be purchased by the validation sites was performed in order to complete the acquisition of the missing data & signs. The approach used was to perform "best effort" integration for all the selected medical devices. All the devices not supporting connectivity or supporting proprietary connectivity without a disclosed protocol and data format by the vendor can be considered as manually connected devices.

### 4.1.2 Devices used in the clinical environment (Hospitals & Primary Health Care Centers)

The devices used in the clinical environment in the scope of the HEARTFAID project can be sub-divided in connectable and non connectable devices. Regarding the connectable devices, an automatic integration can be achieved, while the integration of the non connectable devices can be achieved only by explicitly typing the data (e.g. the measurement values seen on the display of the device) on a form and transmitting them to the HEARTFAID platform.

#### 4.1.2.1 Connectable Devices

The following table includes the list of the connectable devices in the scope of HEARTFAID

| Device Model | Device Type |
|---|---|
| A&D UA-767PC | OSCILLOMETRIC BLOOD PRESSURE MEASURING DEVICE |
| A&D TM-2430 | 24H AMBULATORY BLOOD PRESSURE MONITOR |
| Spacelabs 90207 | 24H AMBULATORY BLOOD PRESSURE MONITOR |
| Spacelabs 90217 | 24H AMBULATORY BLOOD PRESSURE MONITOR |
| ArchiMed 4210 | Resting ECG |
| CardioScan 2 Premier | 24H Holter ECG monitoring |
| Finometer Pro | Non-invasive beat-to-beat blood pressure monitoring |

#### 4.1.2.2 Manually Connected Devices

The following table includes the list of the manually conned devices in the scope of HEARTFAID

| Device Model | Device Type |
|---|---|
| Nellcor PBN-180 | Pulse Oximeter |
| Oxycon Mobile | Cardiopulmonary Test |
| V Max 229 | Cardiopulmonary Test |
| Nihon Kohden | Resting ECG |
| Delmar-Reynolds | 24H Holter ECG |
| SPIROBANK-COSMED | Spirometer |
| CASE 601 | Cardiopulmonary Test on Treadmill |
| EC 5R | Plethysmograph |
| SPHYGMOCOR | Tonometer |
| OMRON M5-I | Oscillometric Blood Pressure Measuring Device |
| V Max 29c | Cardiopulmonary Test |
| Marquette Case 15 | ECG Stress Test |
| Ascard B56 | Resting ECG |
| Bioscan 916 | Body Impedance Analyzer |
| AKERN, BIA 101 | Body Impedance Analyzer |

### 4.1.3 Devices used in the personal environment (home environment and on-the-move environment)

The connectable devices for the personal environment can be seen in the following table:

| Device Model | Device Type |
|---|---|
| Nonin 4100 | Pulse Oximeter |
| FRWD Sports computer | Heart Rate |
| A&D UA-767PBT | Oscillometric Blood Pressure Monitor |
| A&D UC-321PBT | Electronic Scale |
| MagIC | Multiparametric Monitoring Vest |

All the devices in the table above are characterised as connectable, while the only non connectable device is the BF-906 – Body Impedance Analyzer.

### 4.1.4 Interoperability Issues

As stated before, lack of interoperability is the main issue which was to be faced during the integration of the medical devices in the HEARTFAID platform. This lack of interoperability can be distinguished in:

- *Lack of implementation of interoperability standards*: In this case, the communication protocols as well as the data format do not comply with any standards.
- *Wrong implementation of interoperability standards*: In this case some effort towards following the standards has been made by the vendors, but either the implementation is wrong or it is does not completely follow the standard.
- *Implementation of proprietary and non-disclosed data format and communication protocols*: In this case, the communication protocols and the data format are not open to third parties for development.

Due to the above problems, the integration of each device has to be analyzed case-by-case and the possible processing of the data on the platform side depends strongly on the information that has been possible to transfer in the HEARTFAID repository. This information has also to be analyzed device-by-device

### 4.1.5 Raw data acquisition

#### 4.1.5.1 A&D UA-767PC Oscillometric Blood Pressure Measuring Device

***i. Device Description:*** This device is connectable to a PC via RS-232 cable, thus its measurements can be transferred to a computer. The device is usable in hospital, home and on-the-move environments but requires the presence of a gateway with a RS-232 port for transferring the measurements to the HEARTFAID platform. This can be easily done in hospital and at home, but it is complex for on-the-move situations.

It was decided to use the device only in hospitals and primary health care centers. Only in these places there will be a RS-232 HEARTFAID gateway.

***ii. The device output data:*** After connecting the device to the PC, the data can be stored in an ASCII file using the "export functionality" of the vendor software. The following figure depicts a sample ASCII file containing the output data.

```
PAS,PAD,FC,ANNO,MESE,GIORNO,ORA,MINUTI,CODICE
188,164,107,2003,2,20,16,33,00000
134,87,69,2003,2,20,16,36,00000
136,96,65,2003,2,20,16,37,00000
124,91,65,2003,2,20,16,39,00000
136,116,56,2003,2,20,16,40,00000
108,54,82,2003,2,21,21,42,00000
101,52,86,2003,2,21,21,43,00000
103,48,84,2003,2,21,21,44,00000
128,92,69,2003,2,24,8,8,00000
132,84,75,2003,2,25,9,52,00000
117,57,89,2003,3,3,11,42,00000
114,54,88,2003,3,3,11,44,00000
118,80,80,2003,3,3,15,36,00000
```

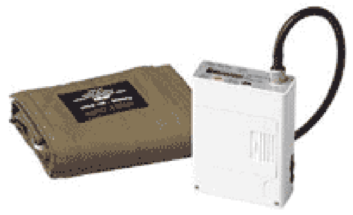
**Figure 2: A&D UA-767PC output data format**

Each line in the file (data coloured in red) represents the following values:

***SBP, DBP, HR, YEAR, MONTH, DAY_OF_MONTH, HOUR, MINUTE***

By parsing this ASCII file, Date, Time, SBP, DBP and HR can be sent to the HEARTFAID repository. It has to be noted that the ASCII file can contain data related to a single acquisition or a stream of values stored in the devices internal memory. However in this case, where the device is to be used by more than one user, a proper mapping between the patient and the performed measurement has to be applied in order to ensure that the measurement refers to the specific person. The most efficient way of achieving this, is to consider only the most recent measurement (the last measurement in the file) as the one to be transferred to the HEARTFAID platform.

#### 4.1.5.2 A&D TM-2430 ambulatory blood pressure monitor



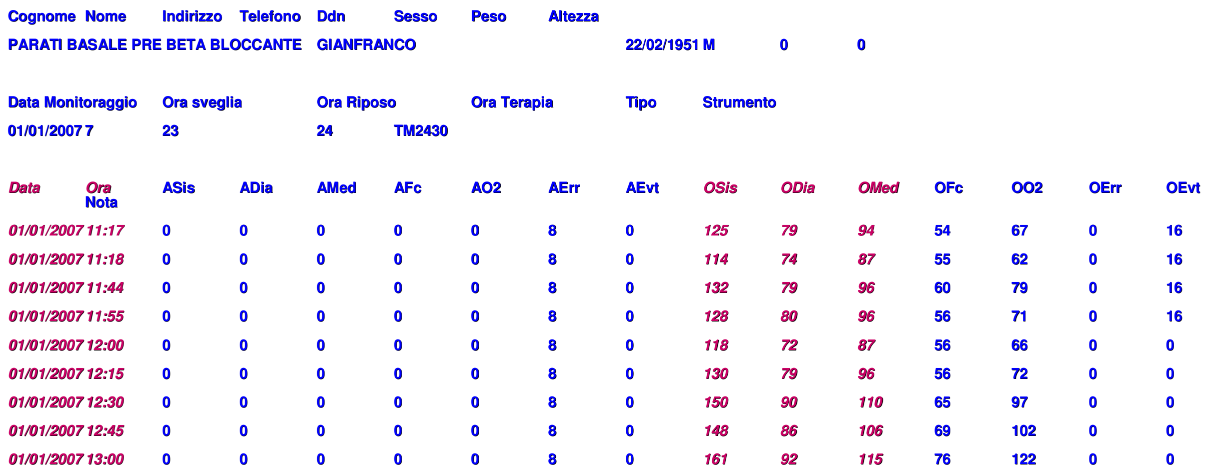
***i. Device Description:*** This device is a data logger that is installed on the patient in the hospital premises and after 24 hours the patient has to go back to the hospital where the device is removed from the patient.
During the time when the patient has the device on it the device does not transfer any information to any PC, only when the device is removed by medical personnel is then connected to the reading station by serial cable. The reading station is

located in the hospital premises, and only specialized medical personnel works on it.
The information is transferred to the HEARTFAID platform only in the hospital premises by connecting the device to a PC where the A&D software for the reading station is installed. The reading station will export a file (under the supervision of the specialized medical personnel) for the HEARTFAID platform in a standard or disclosed format.

***ii. The device output data:*** This device makes available an ASCII file containing the 24H measurements. From the ASCII file, the following values are transferred (in the proper time order) in the HEARTFAID repository:

***Date Time SBP DBP ABP***

In the following figure, the output ASCII file containing the A&D TM-2430 exported data can be seen.



| Cognome | Nome | Indirizzo | Telefono | Ddn | Sesso | Peso | Altezza | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| PARATI BASALE PRE BETA BLOCCANTE | | | | GIANFRANCO | | | | 22/02/1951 M | | 0 | 0 |

| Data Monitoraggio | Ora sveglia | Ora Riposo | | Ora Terapia | Tipo | Strumento |
|---|---|---|---|---|---|---|
| 01/01/2007 7 | 23 | 24 | TM2430 | | | |

| Data | Ora Nota | ASis | ADia | AMed | AFc | AO2 | AErr | AEvt | OSis | ODia | OMed | OFc | OO2 | OErr | OEvt |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 01/01/2007 | 11:17 | 0 | 0 | 0 | 0 | 0 | 8 | 0 | 125 | 79 | 94 | 54 | 67 | 0 | 16 |
| 01/01/2007 | 11:18 | 0 | 0 | 0 | 0 | 0 | 8 | 0 | 114 | 74 | 87 | 55 | 62 | 0 | 16 |
| 01/01/2007 | 11:44 | 0 | 0 | 0 | 0 | 0 | 8 | 0 | 132 | 79 | 96 | 60 | 79 | 0 | 16 |
| 01/01/2007 | 11:55 | 0 | 0 | 0 | 0 | 0 | 8 | 0 | 128 | 80 | 96 | 56 | 71 | 0 | 16 |
| 01/01/2007 | 12:00 | 0 | 0 | 0 | 0 | 0 | 8 | 0 | 118 | 72 | 87 | 56 | 66 | 0 | 0 |
| 01/01/2007 | 12:15 | 0 | 0 | 0 | 0 | 0 | 8 | 0 | 130 | 79 | 96 | 56 | 72 | 0 | 0 |
| 01/01/2007 | 12:30 | 0 | 0 | 0 | 0 | 0 | 8 | 0 | 150 | 90 | 110 | 65 | 97 | 0 | 0 |
| 01/01/2007 | 12:45 | 0 | 0 | 0 | 0 | 0 | 8 | 0 | 148 | 86 | 106 | 69 | 102 | 0 | 0 |
| 01/01/2007 | 13:00 | 0 | 0 | 0 | 0 | 0 | 8 | 0 | 161 | 92 | 115 | 76 | 122 | 0 | 0 |

**Figure 3: A&D TM-2430 output data format**

Out of the above data, the red values are inserted in the HEARTFAID repository producing a 24H stream of Date Time SBP DBP ABP.

### 4.1.5.3 Spacelabs 90207 and 90217 ambulatory blood pressure monitors

***i. Devices Description:*** These devices are data loggers that are installed on the patient in the hospital premises and after 24 hours the patient has to go back to the hospital where the device is removed from the patient.
During the time when the patient

has the device on it the device does not transfer any information to any PC. Only when the device is removed by medical personnel, is then connected to the reading station by serial cable and transmits all the information. The reading station is located in the hospital premises, and only specialized medical personnel works on it.
The information is transferred to the HEARTFAID platform only in the hospital premises by connecting the device to a PC where the Spacelabs software for the reading station is installed. The reading station will export a file (under the supervision of the specialized medical personnel) for the HEARTFAID platform in a standard or disclosed format.

***ii. The device output data:*** This devices make available an RPT file (ASCII) containing the 24H measurements with some additional information.

```
.....
9, 48, 76, 64, 59, 62,"",""
 10, 17, 85, 68, 59, 66,"",""
 10, 47, 90,  0,  0,  0,"EE",""
 10, 50, 83, 72, 63, 64,"",""
 11, 17, 96, 79, 69, 67,"",""
 11, 47, 98, 79, 69, 67,"",""
 12, 17, 93, 80, 75, 73,"",""
 12, 47, 98, 75, 66, 58,"",""
 13, 17, 99, 73, 57, 65,"",""
 13, 47, 88, 63, 51, 64,"",""
 14, 17,109, 84, 73, 69,"",""
 14, 47,113, 88, 72, 67,"",""
 15, 17,113, 88, 77, 67,"",""
 15, 47,124, 98, 84, 64,"",""
 16, 17,125, 97, 82, 60,"",""
 .....
```

**Figure 4: Spacelabs output data format**

In the above figure, the format of the output data (RPT file) of the Spacelabs Devices can be seen. Lines marked with “EE” are discharged, while the red values are inserted in the HEARTFAID repository producing a 24H stream of:

***Date Time SBP ABP DBP HR***

### 4.1.5.4 Archimed 4210 resting ECG

***i. Device Description:*** ESAOTE Archimed is a family of cardiographs manufactured by ESAOTE (Italian company). The model 4210 is the basic model without any diagnostic program able to export the ECG examinations on a floppy disk. This device is only used in hospital premises under the supervision of medical or paramedical personnel.

***ii. The device output data:*** The data format of the file stored on the floppy disk (files with SCP extension) is a dialectal implementation of the SCP-ECG standard (EN 1064:2005+A1:2007 and under approval by ISO). The content of the files is clearly understandable by the FORTH ECG viewer.

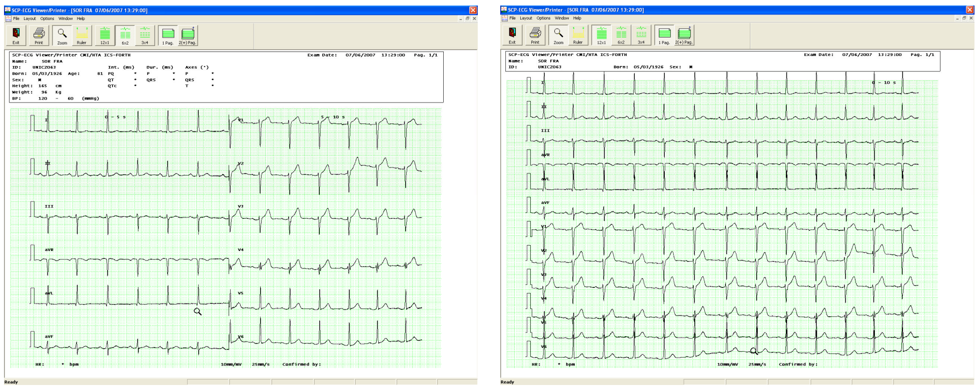


**Figure 5: FORTH ECG Viewer on an ArchiMed 4210 ECG**

### 4.1.5.5 CardioScan 2 Premier

***i. Device Description:*** This device is composed of a data logger (DMS-300) that is installed on the patient in the hospital premises and after 24 hours the patient has to go back to the hospital where the device is removed from the patient.

During the time when the patient has the device on it the device does not transfer any information to any PC, only when the device is removed by medical personnel is then connected to the reading station by a flashcard reader. The reading station is located in the hospital premises, and only specialized medical personnel works on it.

The information is transferred to the HEARTFAID platform only in the hospital premises by connecting the device to a PC where the CardioScan 2 Premier software for the reading station is installed. The reading station will export a file (under the supervision of the specialized medical personnel) for the HEARTFAID platform in a standard or disclosed format.

***ii. The device output data:*** The raw ECG is not exported by the reading station. There are 3 exported files for each examination (p_name.ID, p_name.TXT, p_name.NSV). The ID file (ASCII) contains some information about Date/Time of the examination. The TXT file (ASCII) contains the time occurrences of the QRS with the RR interval related to the previous QRS. The NSV file (binary) contains the time occurrences of the QRS with their classification.

#### 4.1.5.6 Finometer Pro Non-invasive beat-to-beat blood pressure monitor

***i. Device Description:*** This device will be used in hospital setting for research workflow. It provides further possibility of the analysis of autonomic function (i.e. blood pressure variability, baroreflex function) and non-invasive haemodynamic parameters assessment.
Data are trasferred to a PC through direct serial line connection.

***ii. The device output data:*** The exported files are marker.txt, .acq and .txt. The .txt file is in ASCII format: the first column refers to blood pressure value, the second corresponds with ECG, and the third refers to respiratory movements (sampling frequency 200Hz).
The marker.txt file (ASCII) contains markers which are indicating some events of the study, for example: end of the supine recording, start of standing, or indicates the phase of recording performed under controlled breathing of the patients, which is important for further analysis limited for these particular phases. The .acq file (binary) is a device specific format that is opened using brand software. ACQ format in fact it is a direct output from the brand software which can be transformed into ASCII file.

| BP | ECG | RM |
|---|---|---|
| 107.391 | -0.00152588 | 1.90125 |
| 105.225 | 0 | 1.9046 |
| 104.034 | -0.0125122 | 1.90796 |
| 102.783 | -0.00976563 | 1.91498 |
| 101.776 | -0.00183105 | 1.92291 |
| 101.318 | -0.00427246 | 1.9342 |
| 101.074 | -0.0106812 | 1.94763 |

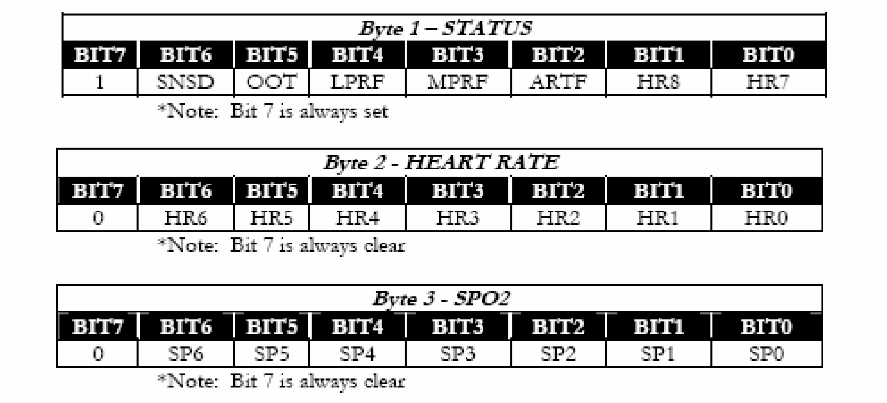
**Figure 6: The FINOMETER Pro txt output file**

The content of the TXT file and the content of the marker.txt file has to be stored (as stream) in the HEARTFAID repository.

### 4.1.5.7 Nonin-4100 Pulse Oximeter

***i. Device Description:*** Nonin 4100 Digital Pulse oximeter allows SpO2, Heart Rate and plethysmographic data to be transmitted wirelessly through a Bluetooth radio to a Bluetooth-enabled device. The benefit of Nonin-4100 is that it removes the cable connection from the sensor cable to the display device giving patients the ability to move freely-without being hindered by cables. Nonin-4100 includes a class II Bluetooth radio which is specified for a range of about 30 feet and battery life of approximately 5 days in power saving mode.
Nonin-400 will be used in home and on-the-move settings, transferring real-time HR and SpO2 measurements via Bluetooth to a PC or mobile device. The communication protocol (even if proprietary) is disclosed by the manufacturer and well-documented.
Nonin-4100 is compatible with two types of sensors: a finger-clip sensor for short term monitoring and a flex sensor for long term monitoring.

***ii. The device output data:*** The device transmits three bytes of data every second.



*Byte 1 – STATUS*

| BIT7 | BIT6 | BIT5 | BIT4 | BIT3 | BIT2 | BIT1 | BIT0 |
|---|---|---|---|---|---|---|---|
| 1 | SNSD | OOT | LPRF | MPRF | ARTF | HR8 | HR7 |

*Note: Bit 7 is always set

*Byte 2 - HEART RATE*

| BIT7 | BIT6 | BIT5 | BIT4 | BIT3 | BIT2 | BIT1 | BIT0 |
|---|---|---|---|---|---|---|---|
| 0 | HR6 | HR5 | HR4 | HR3 | HR2 | HR1 | HR0 |

*Note: Bit 7 is always clear

*Byte 3 - SPO2*

| BIT7 | BIT6 | BIT5 | BIT4 | BIT3 | BIT2 | BIT1 | BIT0 |
|---|---|---|---|---|---|---|---|
| 0 | SP6 | SP5 | SP4 | SP3 | SP2 | SP1 | SP0 |

*Note: Bit 7 is always clear

The first byte is the status byte, which indicates the status of the device using several flag bits (BIT2 to BIT6). Possible cases are: Sensor Disconnect (SNSD), Out Of Track (OOT), LPRF (Low Perfusion), MRPF (Marginal Perfusion), ARTF (Artifact).
HR0 to HR8 indicate Heart Rate (Standard 4-beat average values not including display holds) and SP0 to SP6 indicate SpO2 (Standard 4-beat average values not including display holds).
When SpO2 and HR cannot be computed, the system will send a missing data indicator. For missing data, the HR equals 511 and the SpO2 equals 127.

#### 4.1.5.8 FRWD Sports Computer – B600

*i. **Device Description:*** The FRWD device comprises a heart rate transmitter belt and a recorder unit. The heart rate transmitter belt wirelessly transmits the heart rate to the unit, which in turn transmits it over Bluetooth to a computer or mobile phone.

***ii. The device output data:*** Every 1 second, the heart rate measurement is transmitted over bluetooth by the recorder unit. The communication protocol is not disclosed to public, thus no information on the structure of the transmitted data can be given in this document.

#### 4.1.5.9 A&D UA-767PBT Oscillometric Blood Pressure Monitor

*i. **Device Description:*** UA-767PBT was designed based on the UA-767 *Plus*, which is clinically proven in accuracy. It has inherited some advanced features from the original model, but has extended capability since it is equipped with Bluetooth® wireless communication technology, giving it extended and advanced capabilities.

***ii. The device output data:*** After the performance of a measurement, the device transmits the single acquisition measurement (Date, Systolic Blood Pressure, Diastolic Blood Pressure and Heart Rate) over bluetooth to a bluetooth connected device. The communication protocol is not disclosed to public, thus no information on the structure of the transmitted data can be given in this document.

#### 4.1.5.10 A&D UC-321PBT Weight Measurement Scales

*i. **Device Description:*** UC-321PBT was designed based on the UC-321PL, which has a 200kg (450lb) capacity with a fine resolution of 100g (0.2lb). It shares some characteristics with the UC-321 Precision Personal Health Scale series. Extended capability of wireless communication is provided by Bluetooth®.

***ii. The device output data**:* After the performance of a measurement, the device transmits the single acquisition measurement (Date, Body Weight) over bluetooth to a bluetooth connected device. The communication protocol is not disclosed to public, thus no information on the structure of the transmitted data can be given in this document.

#### 4.1.5.11 Multiparametric Monitoring Vest

***i.*** ***Device Description:*** The MagIC device is a textile-based wearable system, developed by Fondazione Don Gnocchi (Italy) for the unobtrusive recording of ECG, respiration and motion signals during spontaneous behaviour in ambulant subjects. The system is composed of a vest, including textile sensors for ECG and breathing frequency detection, and a portable electronic board for motion assessment, data storage and remote Bluetooth transmission.

***ii.*** ***The device output data:*** During recording data can be locally stored on a memory card and /or transmitted to a remote computer via a Bluetooth connection through a serial protocol. Raw data are packed according to a proprietary format that will be disclosed to the Heartfaid Consortium. A decoding routine is also available to transform raw data in a .txt file.

## 4.2 Data Acquisition Applications

### 4.2.1 Clinical Environment

#### 4.2.1.1 Data Acquisition Application

*1. Description of the application functionality*

The devices used in this prototype in the clinical environment are connected to a reading station (a PC located at the clinical site). In the reading station, the vendor reading software is installed, which is responsible to communicate with the given medical device and read the data from it. The communication protocol is usually a proprietary protocol which is not disclosed by the vendor.

Given that vendor software has been used in order to acquire the data from a device, these data are exported and saved into a file on the hard disk of the reading station.

The purpose of the application described in this section is to integrate the devices located in clinical sites to the HEARTFAID platform via the parsing of the stored data, construction of the appropriate messages to be sent to the platform and, finally, transmission of these messages to the platform over an IP connection.

*2. Installation of the application*

The application is implemented in Java and compiled into a JAR file. It is distributed in a bundle containing the application JAR file, an additional JAR file (containing Netbeans GUI swing elements) and the Java Runtime Environment, which enables the execution of the application in PCs that do not have a Java Developer's Kit (JDK) installed. The following figure depicts the folder structure after copying the files into the PC's hard disk.

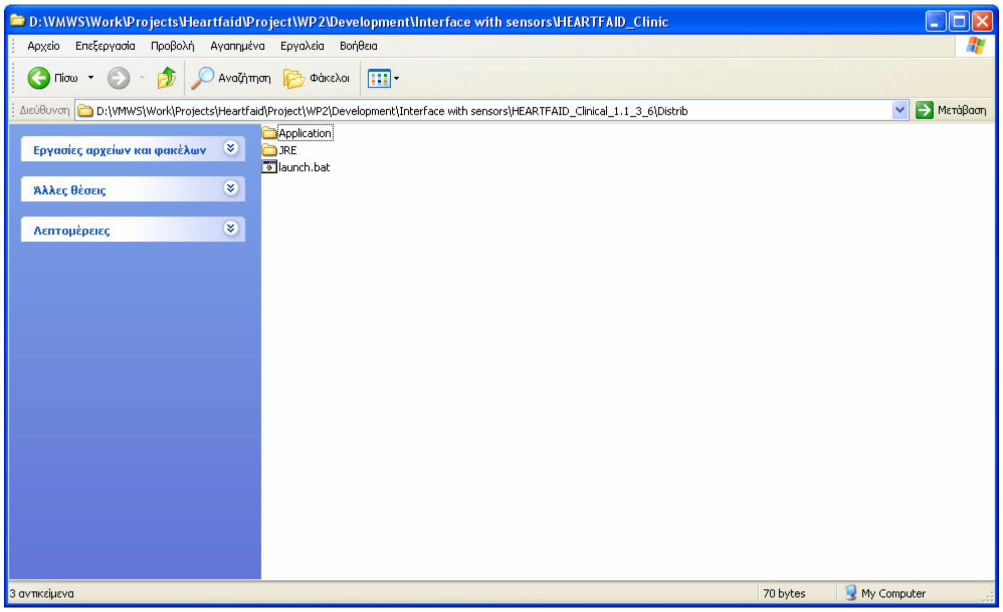

D:\VMWS\Work\Projects\Heartfaid\Project\WP2\Development\Interface with sensors\HEARTFAID_Clinic
Αρχείο
Επεξεργασία
Προβολή
Αγαπημένα
Εργαλεία
Βοήθεια
Πίσω
Αναζήτηση
Φάκελοι
Διεύθυνση
D:\VMWS\Work\Projects\Heartfaid\Project\WP2\Development\Interface with sensors\HEARTFAID_Clinical_1.1_3_6\Distrib
Μετάβαση
Εργασίες αρχείων και φακέλων
Άλλες θέσεις
Λεπτομέρειες
Application
JRE
launch.bat
3 αντικείμενα
70 bytes
My Computer

**Figure 7: The folder structure after the installation of the application**

The Application folder contains the application JAR file, as well as any additional JAR files needed. The JRE folder contains the Java Runtime Environment, in order to execute the application. In order to run the application, the user has to execute the "launch.bat" batch file, which calls the runtime environment as follows:

```
./JRE/jre1.6.0_01/bin/java" -jar ./Application/HEARTFAID_Clinical.jar
```

## 3. *Application User's guide*

The application start screen is depicted in the following figure.

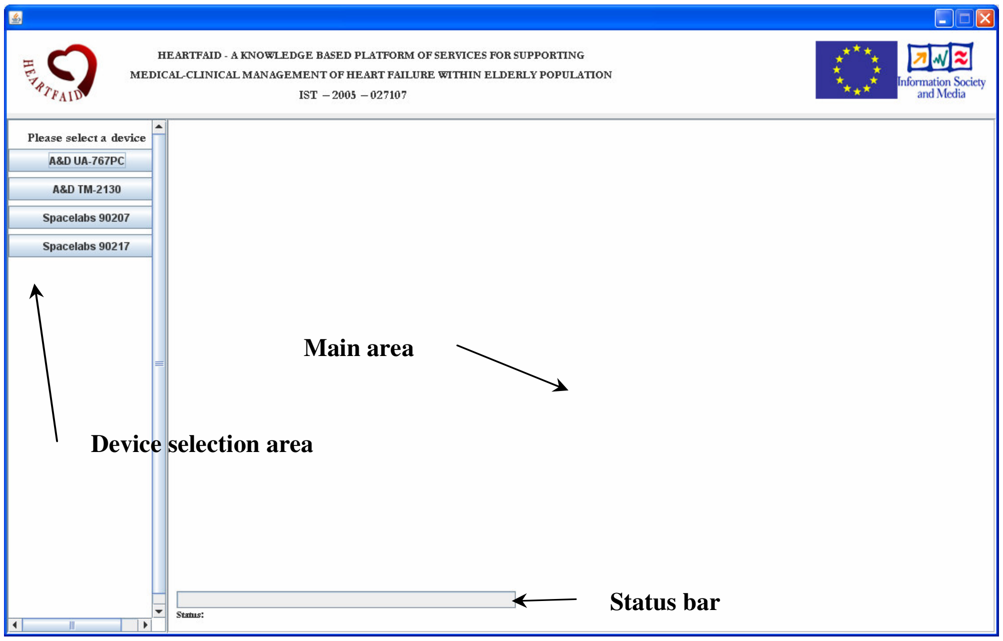

HEARTFAID - A KNOWLEDGE BASED PLATFORM OF SERVICES FOR SUPPORTING
MEDICAL-CLINICAL MANAGEMENT OF HEART FAILURE WITHIN ELDERLY POPULATION
IST – 2005 – 027107
Information Society and Media
Please select a device
A&D UA-767PC
A&D TM-2130
Spacelabs 90207
Spacelabs 90217
Main area
Device selection area
Status bar
Status:

**Figure 8: The application initial screen**

On the left part of this screen, in the "Device selection area", the user can select the device to be used. Once a device is selected, the relative form corresponding to the device appears in the main area.

Upon the initialisation of the application, a working folder named "Work" is created at the top level of the directories structure. Since the creation of the XML files is considered as an intermediate step during the device integration procedure, all the created XML messages can be considered as temporary data which are stored in the "Work" folder. During the application shut down, all the files in the work directory, as well as the directory itself are deleted.

For the sake of simplicity, the functionality of the application is demonstrated using the form for the A&D UA-767PC Blood Pressure Monitor. The procedure followed for the other devices is exactly the same.

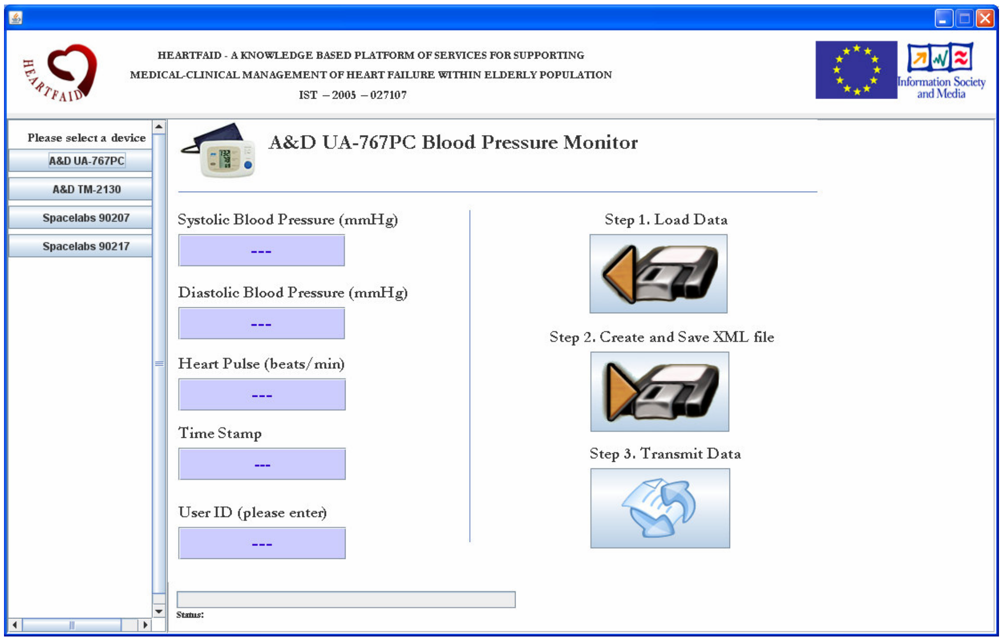

HEARTFAID - A KNOWLEDGE BASED PLATFORM OF SERVICES FOR SUPPORTING
MEDICAL-CLINICAL MANAGEMENT OF HEART FAILURE WITHIN ELDERLY POPULATION
IST – 2005 – 027107
Information Society and Media
Please select a device
A&D UA-767PC
A&D TM-2130
Spacelabs 90207
Spacelabs 90217
A&D UA-767PC Blood Pressure Monitor
Systolic Blood Pressure (mmHg)
---
Diastolic Blood Pressure (mmHg)
---
Heart Pulse (beats/min)
---
Time Stamp
---
User ID (please enter)
---
Step 1. Load Data
Step 2. Create and Save XML file
Step 3. Transmit Data
Status:

**Figure 9: The form representing the A&D UA-767PC device**

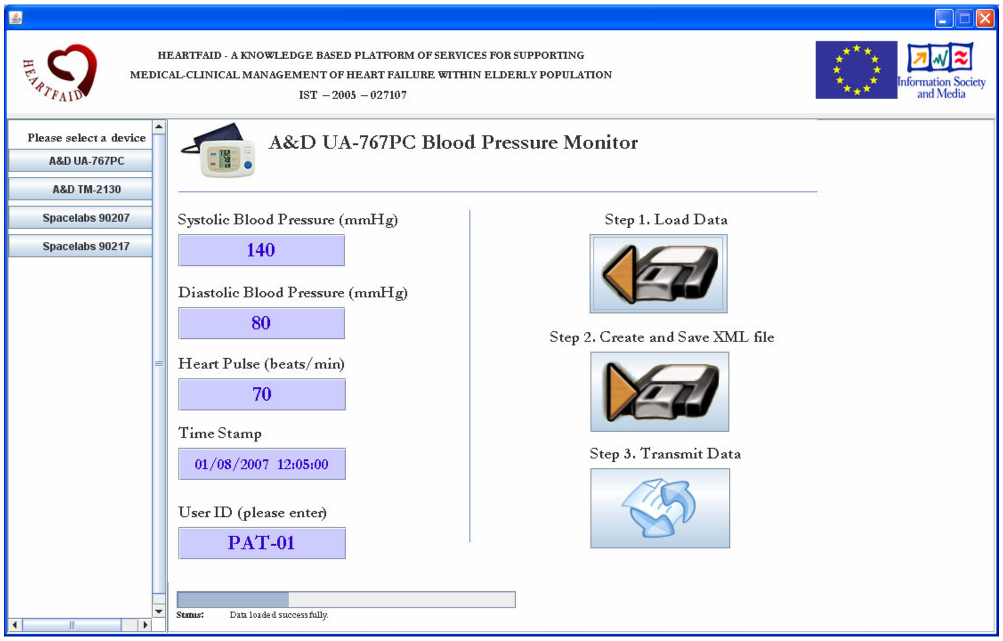
On the left part of this form, the text fields which will display the values read by the file can be seen. Since no data are loaded at this moment, the fields contain the default value “---“. On the right part of the form the three buttons corresponding to the steps of data loading, XML message creation and data transmission can be seen.

i. Definition of the user ID

In the beginning, the user has to enter the User ID field in order to specify the patient which has performed the measurement. Considering that a device can be used by more than one patient and that the stored ASCII data by the vendor’s software do not contain any information about this user, the definition of the UserID is mandatory at the stage of the data parsing, since it is the only way to map the patient to the specific measurement.

ii. Loading the data from a file

When the “Load Data” button is pressed, a dialogue box appears which prompts the user to select the file to load the data from. After the selection of the file, the main screen is as follows (Figure 10). Also, in the status bar, the successful loading of the data is indicated by the phrase “Data Loaded Successfully”. In case of an error during the procedure of data loading, the phrase “ERROR parsing the file” is displayed in the status bar.

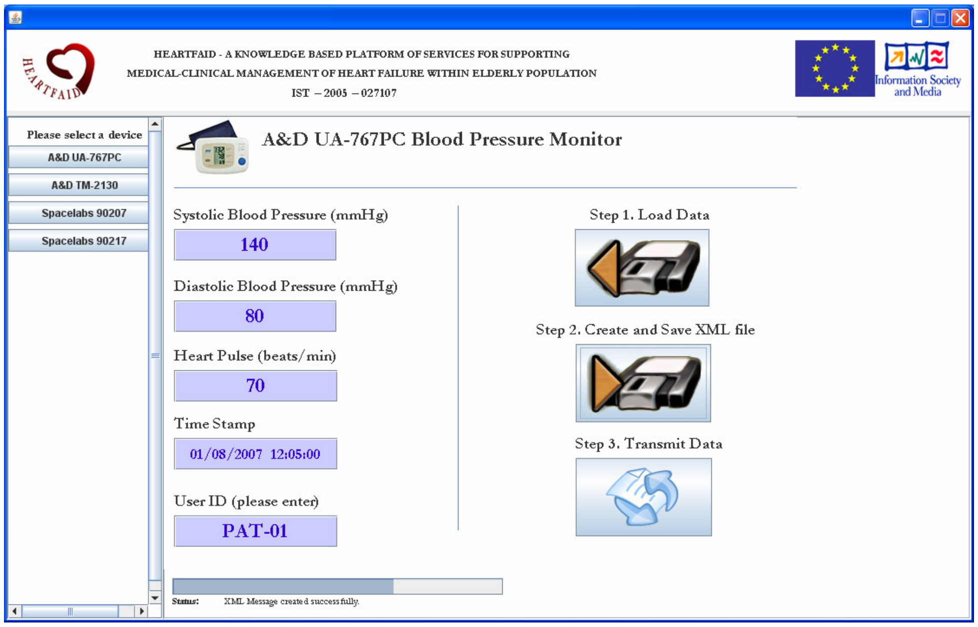

HEARTFAID - A KNOWLEDGE BASED PLATFORM OF SERVICES FOR SUPPORTING
MEDICAL-CLINICAL MANAGEMENT OF HEART FAILURE WITHIN ELDERLY POPULATION
IST – 2005 – 027107
Information Society and Media
Please select a device
A&D UA-767PC
A&D TM-2130
Spacelabs 90207
Spacelabs 90217
A&D UA-767PC Blood Pressure Monitor
Step 1. Load Data
Step 2. Create and Save XML file
Step 3. Transmit Data

Systolic Blood Pressure (mmHg): 140
Diastolic Blood Pressure (mmHg): 80
Heart Pulse (beats/min): 70
Time Stamp: 01/08/2007 12:05:00
User ID (please enter): PAT-01
Status: Data loaded successfully.

**Figure 10: The main screen after loading the data from a file**

iii. XML message creation

After loading the data, the user has to press the "Create and save XML" button. By pressing this button, the XML message which will be sent to the platform is created and stored in a file in the "Work" folder. After the successful creation and storage of the message, the phrase "XML message created successfully" is displayed in the status bar.

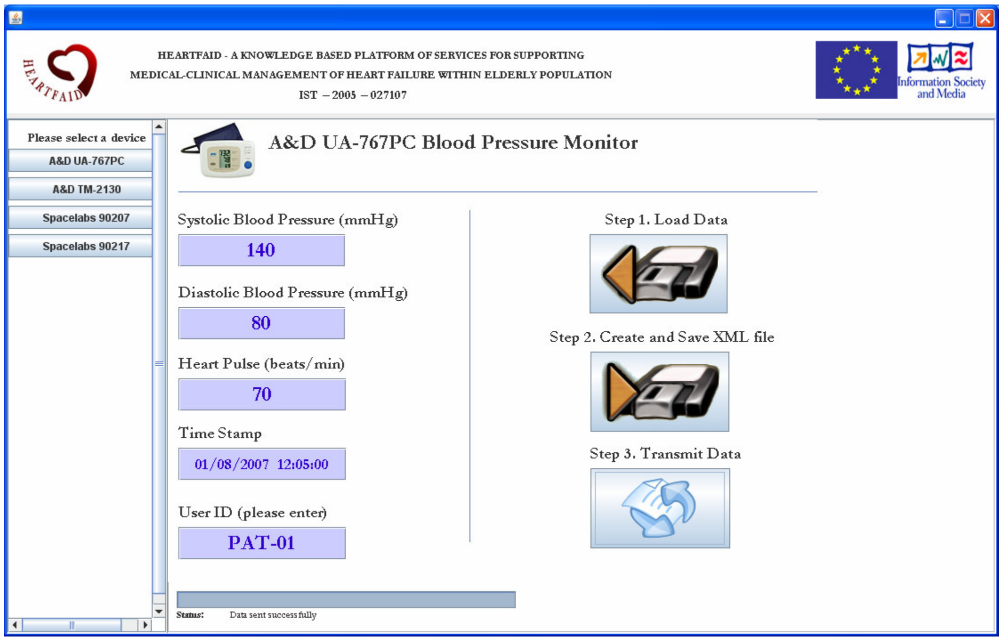

HEARTFAID - A KNOWLEDGE BASED PLATFORM OF SERVICES FOR SUPPORTING
MEDICAL-CLINICAL MANAGEMENT OF HEART FAILURE WITHIN ELDERLY POPULATION
IST – 2005 – 027107
Information Society and Media
Please select a device
A&D UA-767PC
A&D TM-2130
Spacelabs 90207
Spacelabs 90217
A&D UA-767PC Blood Pressure Monitor
Step 1. Load Data
Step 2. Create and Save XML file
Step 3. Transmit Data

Systolic Blood Pressure (mmHg): 140
Diastolic Blood Pressure (mmHg): 80
Heart Pulse (beats/min): 70
Time Stamp: 01/08/2007 12:05:00
User ID (please enter): PAT-01
Status: XML Message created successfully.

**Figure 11: The main screen after creating and saving the XML message**

iv. XML message transmission

Finally, after having created the XML message to be sent to the HEARTFAID platform, the user presses the "Transmit Data" button and the data transmission process begins. The transmission takes place over an http connection to the platform. Upon the successful transmission of the data, the status bar displays the phrase "Data transmitted successfully". If an error occurs during the transmission, the phrase "Transmission error" is displayed in the status bar.


HEARTFAID - A KNOWLEDGE BASED PLATFORM OF SERVICES FOR SUPPORTING
MEDICAL-CLINICAL MANAGEMENT OF HEART FAILURE WITHIN ELDERLY POPULATION
IST – 2005 – 027107

Please select a device
A&D UA-767PC
A&D TM-2130
Spacelabs 90207
Spacelabs 90217

A&D UA-767PC Blood Pressure Monitor

Systolic Blood Pressure (mmHg): 140
Diastolic Blood Pressure (mmHg): 80
Heart Pulse (beats/min): 70
Time Stamp: 01/08/2007 12:05:00
User ID (please enter): PAT-01

Step 1. Load Data
Step 2. Create and Save XML file
Step 3. Transmit Data

Status: Data sent successfully

**Figure 12: The main screen after successful transmission of the XML message**

## 4. *UML Model of the application*

The UML model of the application is depicted in Figure 13. For the sake of simplicity and aiming to demonstrate the application from the scope of software design, the figure shows only the classes related to the UA-767-PC device.

In figure, one can see three packages, namely the GUI package which contains classes related to the graphical elements of the application, the http package which contains elements ralative to data transmission and the UA-767 package which contains elements related to the UA-767 device.

The GUI package contains two classes: the MainFrame, which is the visual representation of the application and the FileChooser, which enables the user to select the appropriate file to be processed by the application. The FileChooser has a reference to the main frame (the callBack variable), which enables it to set the correct environment in the MainFrame once the file is selected.

The http package contains the class named XMLTransmitter, which is responsible to transmit the XML data to the remote location. The main interface of this class is the method setXML(String s) which sets the XML data to be transmitted and the method sendXML(), which performs the data transmission. The MainFrame has a reference to an object of this class in order to be able to transmit the XML data upon the user's request in the GUI.

Finally, the UA767 package contains three classes:

a. class UA767MeasurementData, which represents the data values when the parsing of a file generated by the UA767 device is finished.

b. class UA767Parser, which performs the parsing of the file and returns an object of the class UA767MeasurementData via the method doParse()

c. class UA767XmlMessageGenerator, which is responsible to generate the XML message to be sent to the HEARTFAID platform. This class has two main variables, the patientID which represents the unique ID of the patient entered through the GUI and the String variable fileName which points to the file to be created in order to store the XML message. This class takes as input the generated UA767MeasurementData (which is created upon the proper parsing of the file) in order to construct the XML message. The message is created and stored to the local disk upon the invocation of the createMessage() method.

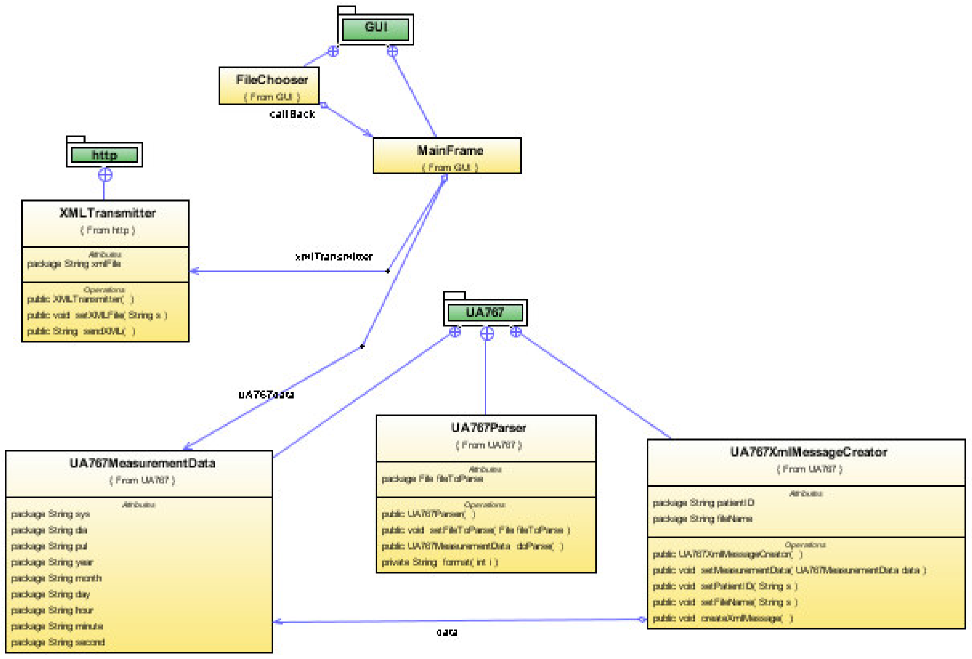


**Figure 13: UML Model of the application**

#### 4.2.1.2 The HEARTFAID eCRF

*I. Description of the application functionality*

The HEARTFAID electronic Case Report Form (eCRF) is part of the web-based portal component of the platform. Its usage is restricted to medical personnel only (nurses, caregivers and doctors) – it is not indented to be accessible for home users. The eCRF is required because not all medical devices can be automatically integrated with the platform as it has been described in D19/4.2.1.1. Because patient's data will be collected also in the research workflow a further important reason for developing the eCFR is the fact that not all required data are present in the routine EHR.

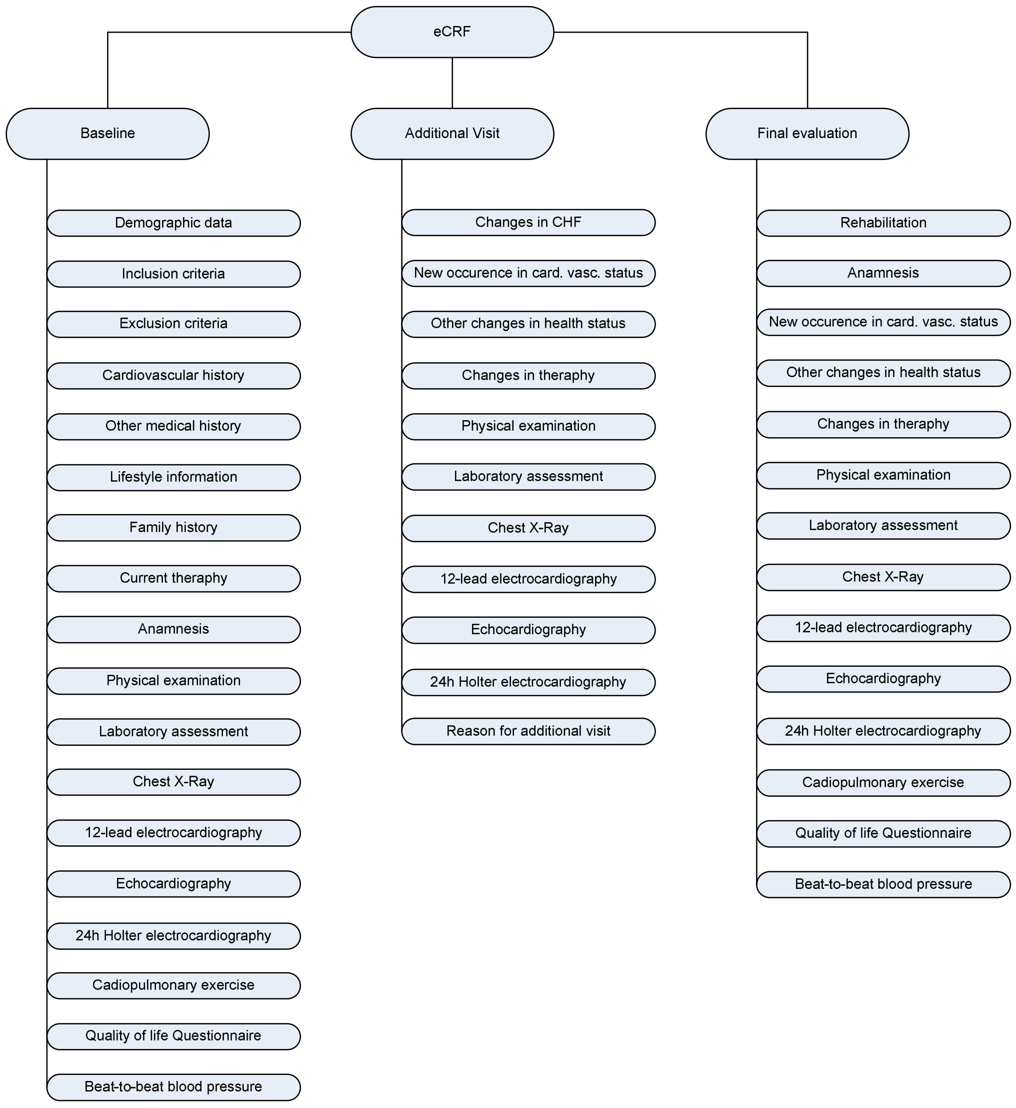


**Figure 14 Schema of the eCRF's prototype**

The prototype of HEARTFAID eCRF is divided into three subforms:

- Baseline Evaluation Form
- Additional Visit Form
- Final Evaluation Form

Each of the subforms is further split into pages which are dedicated to specific medical subjects as Chest X-Ray, 24h Holter Cardiography or Beat-to-beat Blood Pressure Monitoring (Figure 14). The Baseline Evaluation Form is created after patient's admission to the HEARTFAID programme. If the patient contacts the doctor again in the course of the treatment, Additional Visit Forms may be appended to the patient's record. In the situation of patient's leaving the HEARTFAID pathway the Final Evaluation Form is created.

Pages in the eCRF comprise of questions, groups and tables (Figure 15). Groups may contain questions, tables and further groups. Tables are sets of questions (cells) in which fields are arranged in rows and columns. The eCRF form contains questions of different types, ranging from simple types like integer to more complex as e.g. the applied medication component. Entered values are validated against predefined rules. Additionally, question groups may be activated or deactivated based on the values entered by the users.

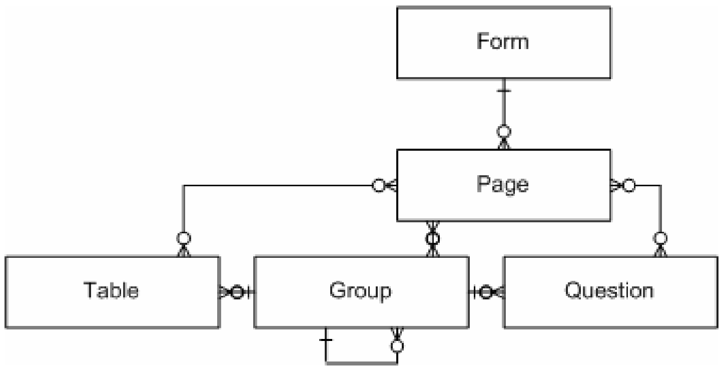


**Figure 15 General schema of the form structure in an eCRF (Crow's Feet notation)**

## *II. Implementation*

The HEARTFAID eCRF is a J2EE web application. Its development process has been accelerated by using the Spring Framework [Johnson2005]. This open-source application framework simplifies programming of all J2EE tiers. The central element of this framework is a configurable *Inversion of Control* (IoC) container which manages the application's objects. The container takes over control of diverse aspects of the application's life cycle. Setting up the environment is done by configuration files (usually in XML format). Thanks to this feature the whole eCRF form structure is not hard-coded into the application and can be altered without the need of recompiling the project.

Web access module of eCRF is implemented using the Spring's Web MVC package. This library facilitates structuring the code following the Model-View-Controller (MVC) architectural design pattern. Spring MVC is very flexible as it allows the integration of many popular web presentation technologies. The HEARTFAID eCRF web user interface is for instance a combination of JSP, JavaScript and XSLT combined together using the Spring's View components. Spring provides the programmer with predefined extendable controllers for on-line forms. Such features as page changing, form validation or form submission are implemented by tailoring predefined abstract methods from the framework.

Spring simplifies also the integration of many J2EE technologies. The HERATFAID eCRF data persistence layer makes use of Hibernate data abstraction layer which is seamlessly integrated into the Spring Framework.

The data model of the eCRF is organized according to the EAV (Entity-Attribute-Value) schema [Nadkarni1997] [Nadkarni1999] [Duftschmid2002]. In conventional database design each parameter of interest is stored in a separate column. The EAV, in contrast, assumes that parameters are kept in rows with three columns. The first column ("entity") identifies the object of the property (for instance the patient), the next column "attribute" identifies the parameter (in our case the question id), and finally the last column contains the value of the question. The EAV approach proved to be useful in applications with large number of parameters and a structure which is likely to change frequently. Because the number of questions in the eCRF is quite large (approx. 700 fields) and its structure is prone to be modified often, it seems that the EAV structure is the right choice for this purpose.

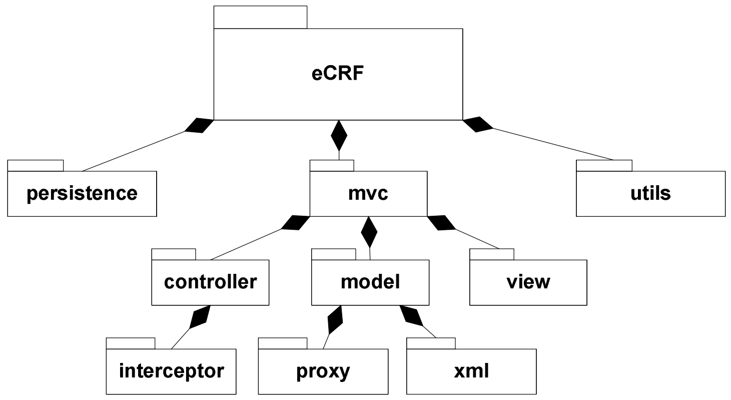


**Figure 16 UML package diagarm of the eCRF prototype**

A diagram of the eCRF prototype code structure is presented in the figure above. The `persistence` subpackage contains classes responsible for connecting the eCRF application with Hibernate. Hibernate is a free, open-source framework, which maps object-oriented schema to a relational database structure. This package provides a data abstraction layer which assures database portability of the project. It has also features for data query and retrieval. The `mvc` package contains three subpackages: `controller`, `model`, `view` - corresponding to the MVC design pattern. The `controller` package comprises of classes processing and dispatching messages in the eCRF service. Classes from the `interceptor` package monitor the dataflow in the application and in case it is needed they may alter it. This feature is for instance used to control the validity of the user session. The `model` package contains code for domain object representation. In order to separate the business logic from the data persistence, proxy classes are used (`proxy` package). The `xml` package groups together classes helpful in building XML messages which are used in the presentation layer by XSLT transformation or are sent to other components of the platform. The `view` package is composed of classes generating the WWW graphical user interface. Finally, the `utils` package contains helper classes useful in various parts of the application.

Many smaller, open source libraries other than Spring and Hibernate, have been used to support diverse parts of the prototype. Among them is JAVS, an open-source framework supporting the EAV data model programming in Java. The data input controls come form a Java Script library written by Jason Moon.

The prototype of the eCRF application has been deployed on an Apache Tomcat (version 5.5.12) application server running under the control of a Windows 2003 Server operating system. For data storage a MySQL (version 5.0.24) RDBMS server has been chosen.

*III. Application User's Guide*

Since the eCRF application is considered as an end user application in the scope of WP6, the readed can find information of the application's user's guide in deliverable D23, Section 7 (User Interface).

### 4.2.2 Personal Environment

#### 4.2.2.1 Data Acquisition Application for Home Scenario

*1. Description of the application functionality*

The Data Acquisition in the Home Environment is accomplished with a light-weight, specialized application, developed in Java 5. The requirements analysis illustrated the need for an extensively user-friendly, highly configurable application that would require the least possible involvement from the user in order to complete its tasks. The software was designed according to these guidelines and version 1.0, which is the current product version, complies to the specifications.

The software architecture followed an object-oriented, modular design, which was assisted by the choice of Java as the implementation language. The Model-View-Controller pattern was the core of the design, but it was enhanced with multithread support and a very basic workflow engine. As depicted in Figure 17, the main modules of the software are the controller, the user interface (view) and the measurements data (model).

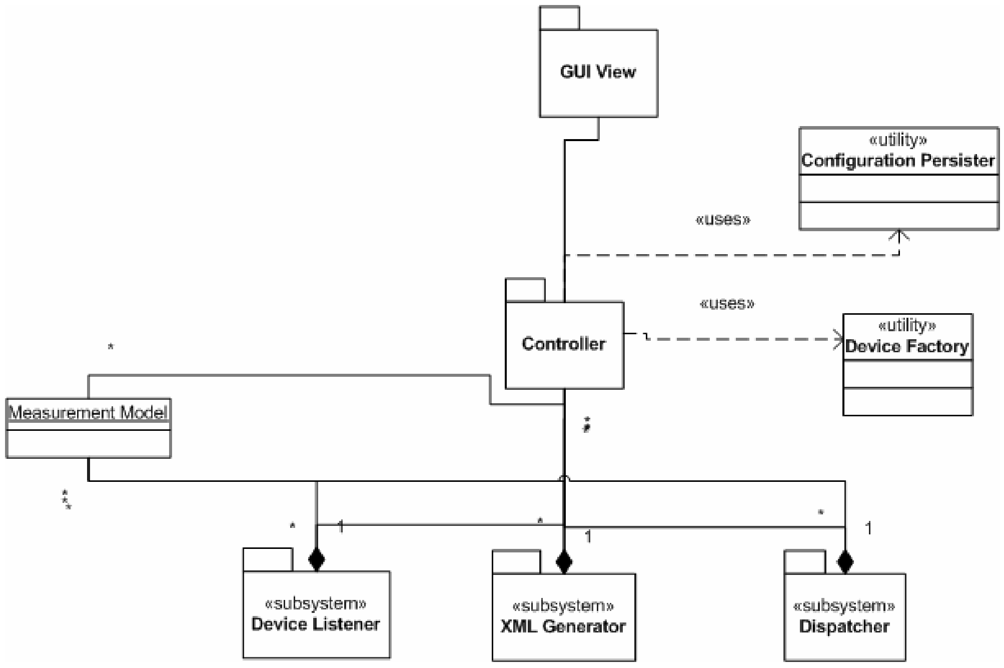


**Figure 17: Static Structure of Home Environment Architecture**

The Graphical User Interface consists of three screens, which encapsulate the three main functions of the software. The three screens are presented in distinct tabs, to allow for easy and intuitive navigation between them. The functionality included in each of the three screens is:

a. The "Activity View" screen features a summarized survey of all the information and control elements necessary for supervising and operating the application. In effect, the patient needs to interact with no other screen than this and still take full advantage of the software.
b. The "Personal Information" screen is arranged as an input form for the patient's identification and some rudimentary demographics data.
c. The "Configuration" screen provides certain options to set up the data acquisition and transmission.

As mentioned above, the Home Client is designed specifically for people with little exposure and familiarity with sensors, computers and high-tech devices in general. Therefore, the software's execution has been structured to be as automated as possible, require little to no user intervention and provide feedback on the processing status.

The software has three modes of operation, "configuration-error", "stand-by" and "running". The first of the three occurs when at least one of the mandatory input fields in the "Personal Information" or "Configuration" screen is missing. While on this mode, the user only has two options in interacting with the software:

a. Apply a modification in the configuration parameters. The state of the application is re-evaluated after each modification, and may change the mode to stand-by if all parameters are valid or maintain the configuration-error mode.

b. Terminate the application.

The software may run continuously in the client machine in stand-by mode and initiate the data retrieval and dispatch whenever the patient starts taking measurements. The application is already configured according to user preferences and server settings when installed, but the default options may be changed both command line and through the GUI. At start-up, the application enters stand-by mode and may stay idle for any length of time, since it consumes practically no resources. While on this operational status, the user may perform a number of actions, which are:

a. Apply a modification in the configuration parameters, similar to the action specified above.
b. Manually initiate the data acquisition process, which brings the application to the running mode.
c. Terminate the application.

“Running” is the third mode of operation for the software, which incorporates three phases, acquisition, marshalling and dispatch. The application may enter this mode either by manual selection or automatically, when a measurement is received from one of the connected sensors. During the first phase – data acquisition – the system “listens” for input from the connected sensors, and performs periodic polling of the sensors that have not responded with a measurement up to that point. The system “listens” until all sensors have produced measurements, or a preconfigured time-out is reached, or one of the following manual commands interrupts the process:

a. Stop listening and forward the measurements that have been retrieved up to that point for marshalling and dispatching.
b. Quit the current process.
c. Terminate the application.

The phase following listening is transforming the data retrieved from the sensors into an XML document that conforms to the schema agreed upon by the partners. Once the document is complete, processing passes to the third phase of the running mode, which is dispatching the data. Throughout these phases, informative messages appear to a designated area of the GUI to keep the user informed that processing is taking place and it advances without problems.

An overview of the software’s control flow is displayed in the activity diagram of Figure 18

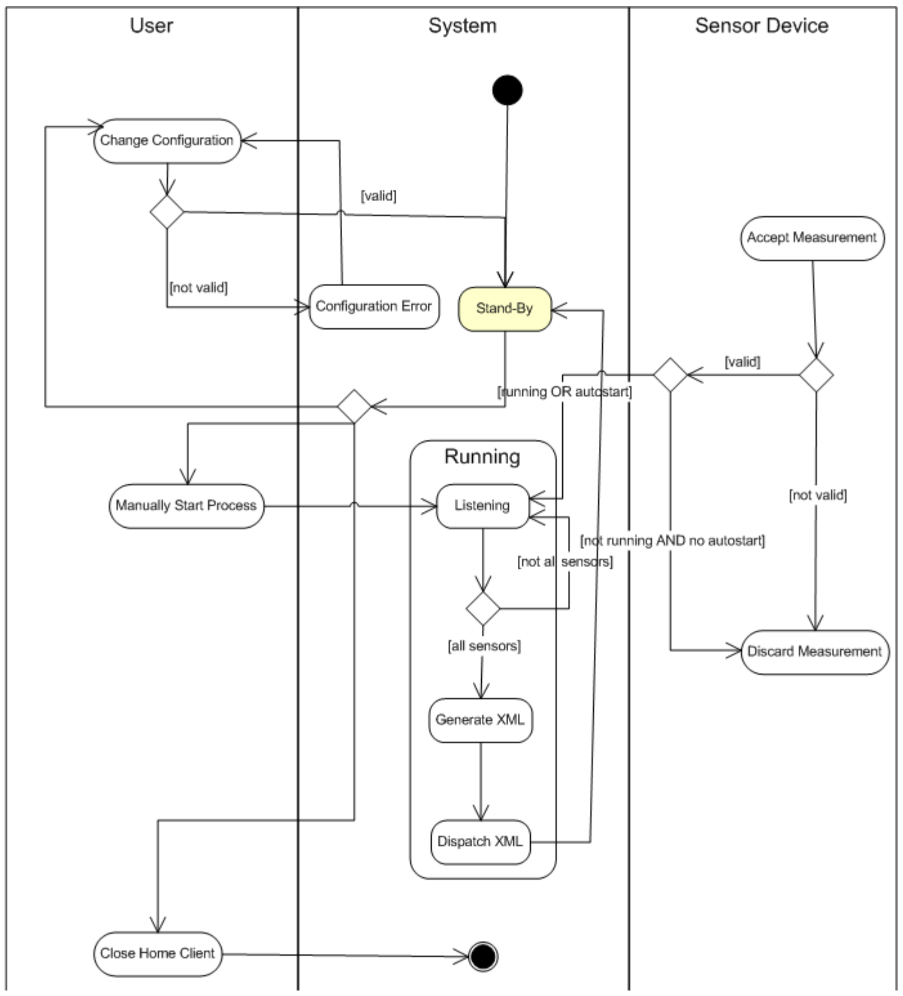


**Figure 18: Activity Diagram for Home Environment**

## 2. Installation of the application

Installing the client of the home environment is a very easy process. All the necessary software – application, external libraries and device drivers – is bundled in one compressed file. The software is written in Java 5, so a compatible JRE, version 1.5.01 and above, should be present in the client PC.

Once the software is copied to the client PC, the following steps should be performed:

1. Uncompress the bundle to a folder of choice. A new folder, named **heartfaid** is created.
2. Enter folder **heartfaid** and open file **heartfaid_exe.bat** for editing.
   a. Set variable HEARTFAID_HOME to the absolute path for the application's root folder, for example HEARTFAID_HOME=C:\Program Files\heartfaid
   b. Set variable JAVA_HOME to the absolute path for the JRE, for example JAVA_HOME=C:\Program Files\Java\jre1.5.0_12

3. Save and close file **heartfaid_exe.bat**. Optionally, create a desktop shortcut to the file, for easier access to the application.

   With these steps, installation is complete. The user may double-click the file **heartfaid_exe.bat** – or the shortcut – to start the application. All configuration parameters are saved in a properties file, named **nurseathome.properties**, which is also situated in the application's root folder. The user may edit the parameters in that file directly, or use the GUI once the application is started.

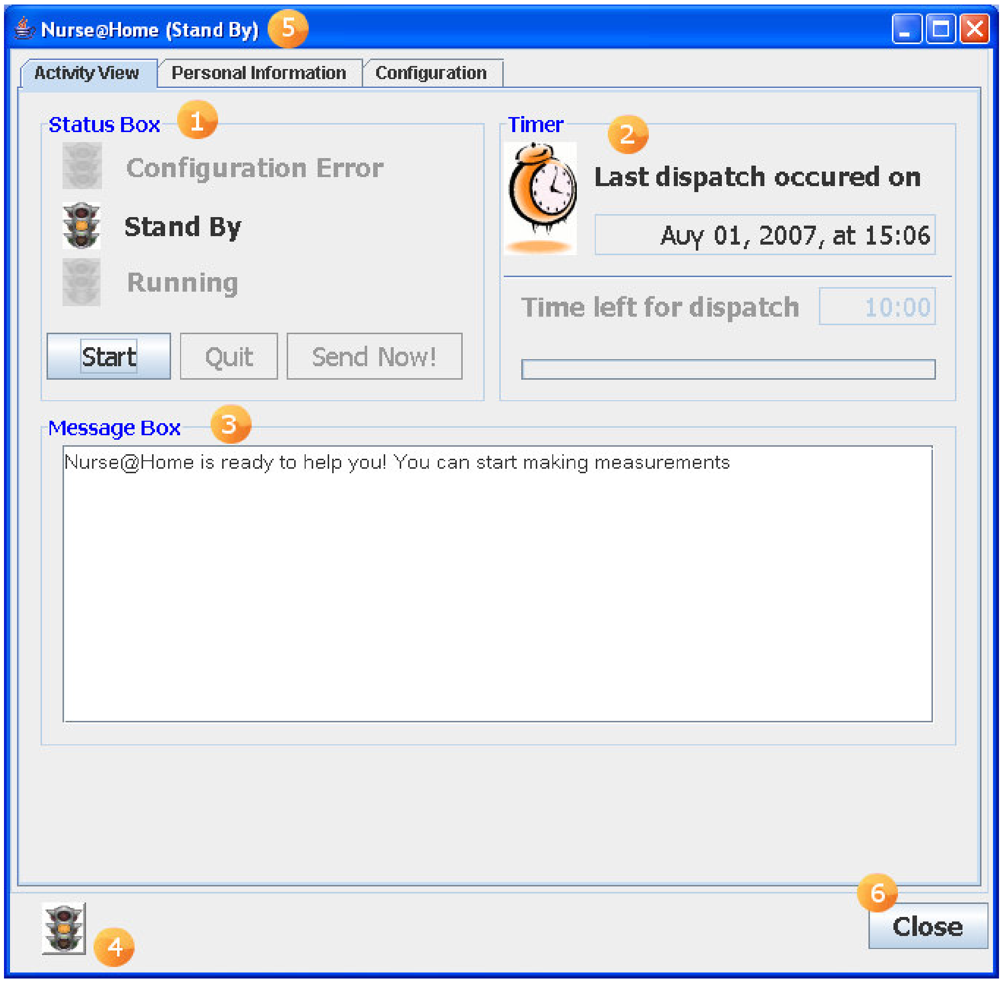
## 3. Application User's guide

The application is designed to perform its basic tasks of assembling and dispatching medical measurements, without the click of a single button. The various functions are divided into three high-level conceptual sections and arranged in a tabular format. The layout and utility of each tab is detailed below.

i. Activity View tab

Once the application starts in Stand-by mode, the user views the main activity screen, which consists of three areas, as presented in Figure 19:

a. The Status Box, denoted with (1) in the figure. This area displays the current mode of the application in textual form - "Configuration Error", "Stand By", "Running" - and using the metaphor of a traffic light. This graphical representation was chosen both because it is acknowledged universally as a "Stop-Go" indicator and to enhance the aesthetics of the user interface. Recognizable symbols and metaphors help decrease the learning curve of a system, especially for novice users.
   The Status Box also contains a button panel with three buttons that control the main actions of data acquisition. Button "Start" initiates the acquisition process, whereupon the other two buttons on the panel become enabled. Button "Quit" stops the process immediately and discards the measurements obtained. Button "Send Now!" stops waiting for measurements and dispatches the ones assembled up to that point.
b. The Timer, denoted with (2) in the figure. This area encloses the temporal related information of the application. The first field displays the date and time, when the last batch of medical measurements was sent to the server.
   Below, is a field that becomes enabled only during the application's "Running" mode and presents a countdown to the time-out. The time-out is the maximum time that the application will wait for input from the sensors. If the time-out is reached, the application will encapsulate and send whichever measurements have been gathered so far, even if not all sensors have responded.
   Under the time-out indicator is a progress bar, which is also enabled only during "Running" mode. It constitutes a visual aid for the user, relevant to the performance and stage of the application.
c. The Message Box, denoted with (3) in the figure. This area is used to display informative messages to the user as to the state and progress of the application.

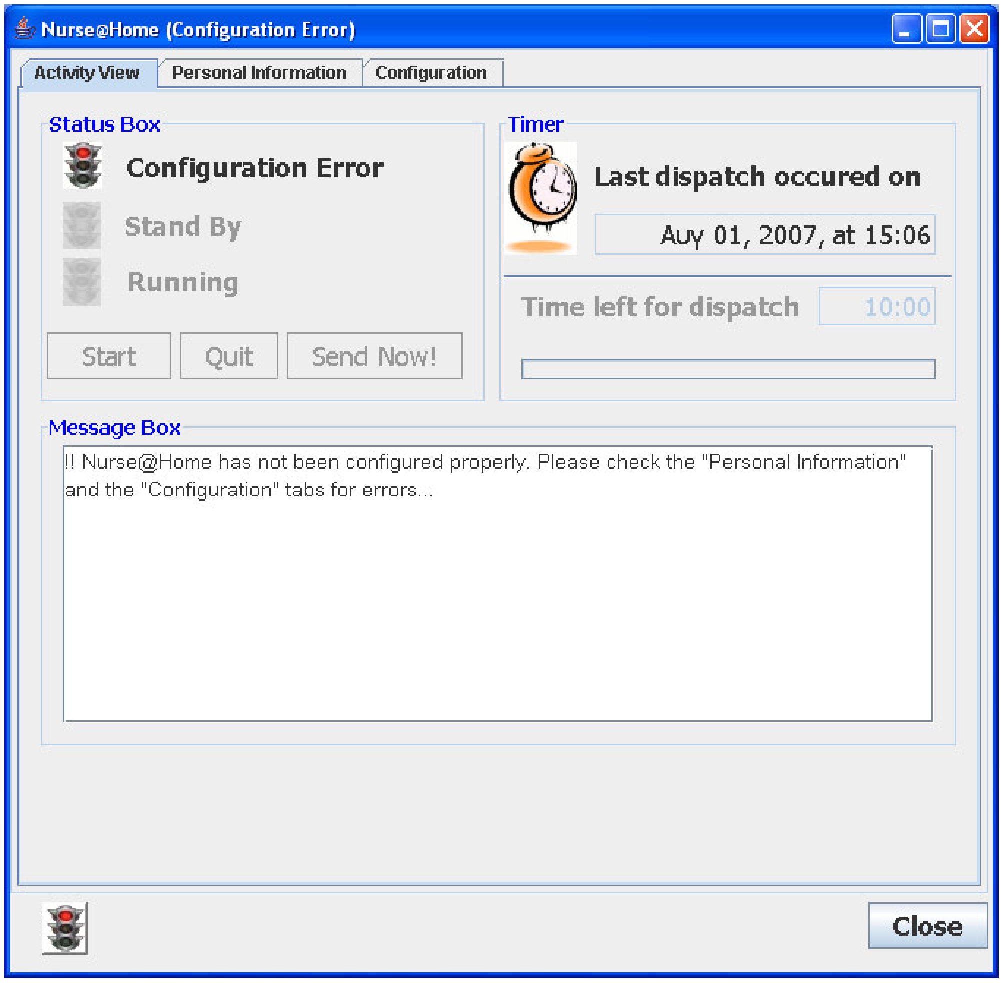


Nurse@Home is ready to help you! You can start making measurements

**Figure 19: "Activity View" in Stand-By mode**

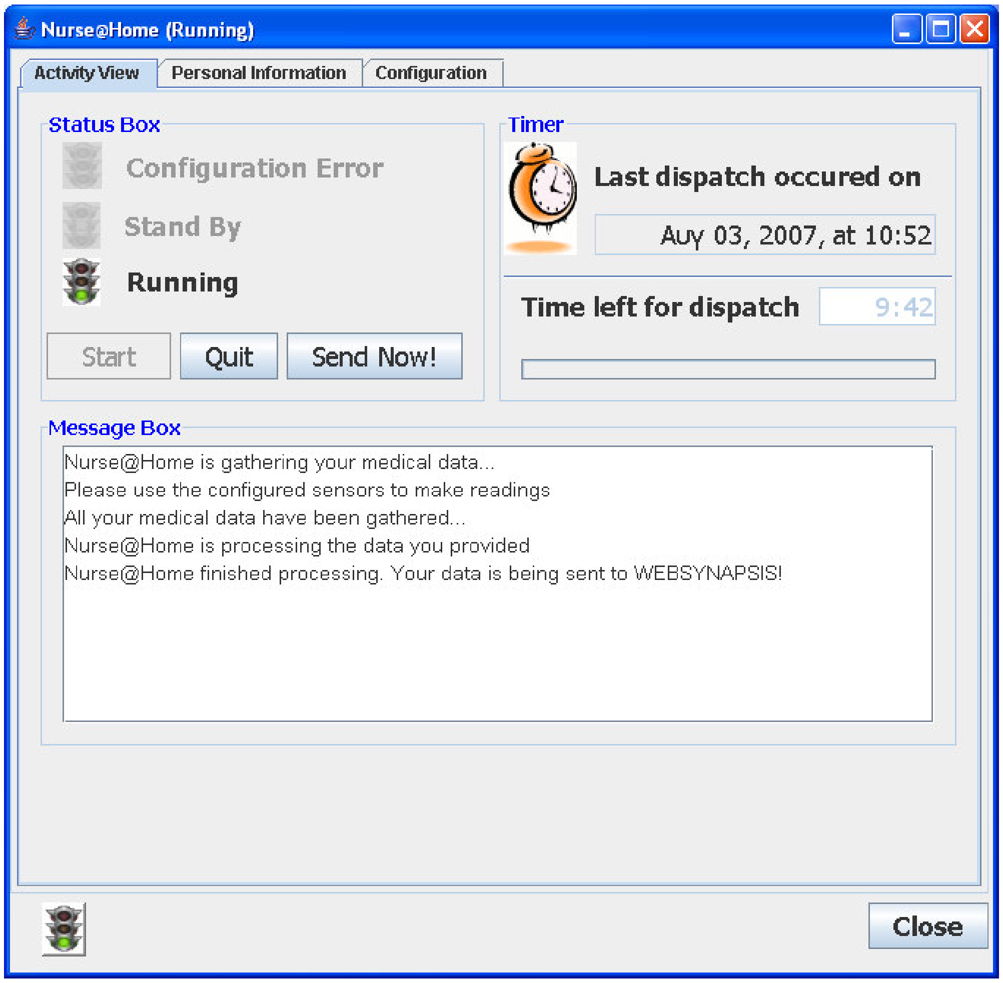
The GUI contains a few further assistive elements. The mode indicated in the status box, is also displayed at the bottom of the interface panel – pointer (4) – below the tabbed pane, to be viewable from all tabs. It is also part of the title bar – pointer (5) – which allows it to be viewable in the OS task bar, even when the application is minimized.

The "Close" button – pointer (6) – which closes the application, irrelevant of its state, is situated below the tabbed pane, so that it can be accessed from all tabs. The "Activity View" tab contains all features a patient will need to run and control the application. Figure 20 and Figure 21 show the tab in the other two modes of operation, "Configuration Error" and "Running".



!! Nurse@Home has not been configured properly. Please check the "Personal Information" and the "Configuration" tabs for errors...

**Figure 20: "Activity View" in Configuration Error mode**

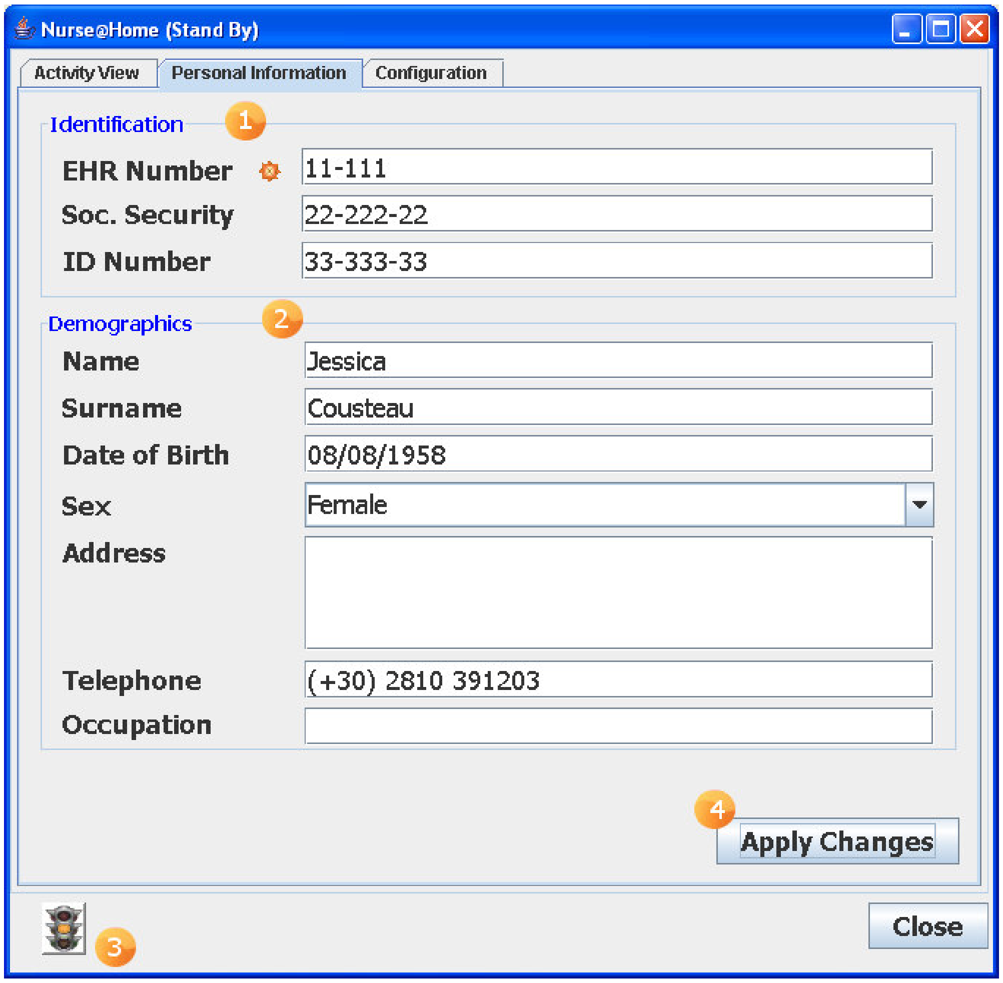


Nurse@Home is gathering your medical data...
Please use the configured sensors to make readings
All your medical data have been gathered...
Nurse@Home is processing the data you provided
Nurse@Home finished processing. Your data is being sent to WEBSYNAPSIS!

**Figure 21: "Activity View" in Running mode**

ii. Personal Information tab

The second tab of the application, Figure 22, which is somewhat more administrative, is the "Personal Information" tab and comprises the means for user identification and for optional demographic input. The patient does not need to interact with this screen, as long as the correct EHR number has been set by an administrator during installation.



Identification

| | |
|---|---|
| EHR Number ✱ | 11-111 |
| Soc. Security | 22-222-22 |
| ID Number | 33-333-33 |

Demographics

| | |
|---|---|
| Name | Jessica |
| Surname | Cousteau |
| Date of Birth | 08/08/1958 |
| Sex | Female |
| Address | |
| Telephone | (+30) 2810 391203 |
| Occupation | |

**Figure 22: "Personal Information" tab**

The tab is divided into the following two sections:

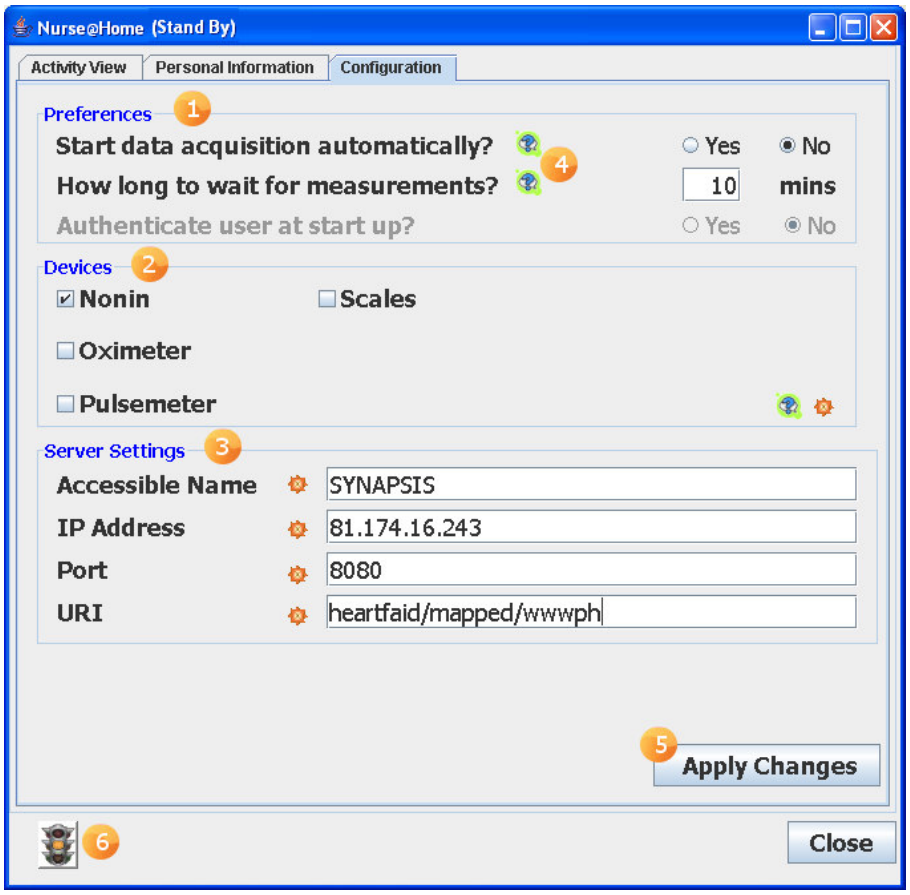
a. The "Identification" area, denoted with (1) in the figure. The field marked with an asterisk is mandatory – the only one in the form – and the application will pass to the Configuration Error mode if this field is left empty.

b. The "Demographics" area, denoted with (2) in the figure. All fields in this area are optional and serve solely to collect some demographical and statistical data for the patient.

An additional noteworthy element of this screen is the graphical status indicator – pointer (3) – which is located under the tabbed pane to be visible from all tabs, as mentioned above. If the user makes any changes in the input fields, s/he clicks the "Apply Changes" button – pointer (4) – to persist the changes.

iii. Configuration tab

The third tab comprises all administrative settings related to the sensor and remote communication configuration. Users will not need to interact with this screen at all, since all contained values should be set prior to installation and do not need editing. A sample screenshot of the tab is displayed in Figure 23.



Preferences
Start data acquisition automatically? Yes [✓ No]
How long to wait for measurements? 10 mins
Authenticate user at start up? Yes [✓ No]

Devices
[✓ Nonin] Scales
Oximeter
Pulsemeter

Server Settings
Accessible Name: SYNAPSIS
IP Address: 81.174.16.243
Port: 8080
URI: heartfaid/mapped/wwwph

**Figure 23: The "Configuration" tab**

Following the paradigm of the other two screens, this tab is also segregated into conceptual areas.

a. The "Preferences" area, denoted with (1) in the figure. The user preferences comprise of two parameters, the auto-start and the time-out period. The coloured question mark next to each question – pointer (4) – is a tool-tip that displays an explanatory text, when the mouse hovers over it.
Auto-start refers to initiating the data acquisition and dispatch process whenever a valid measurement arrives, without the need to press the "Start" button in the "Activity View" tab. When this setting is ON, the patient does not need to interfere at all with the application, not even to click a single button. S/he only needs to use the sensors at the specified times. By default it is set to OFF.
The time-out period is the time – in minutes – that the application will wait for data from the sensors, once it enters "Running" mode, i.e. once the data acquisition starts. When this time passes, the application stops

listening to sensors and forwards to the server the data received so far. By default, this value is set to 10 minutes.

b. The “Devices” area, denoted with (2) in the figure. This is a list of all devices supported by the software. The selected items take part in the data acquisition during the “Running” mode.
c. The “Server Settings” area, denoted with (3) in the figure. The details of the network communication are configured in this section.

When any of these settings is altered, the “Apply Changes” button – pointer (5) – should be pressed for them to take effect. Once again, one should note the status indicator – pointer (6) – on the bottom left corner. If the settings are modified and an invalid state occurs, the indicator will turn to a red light, which notifies the user of the error immediately.

#### 4.2.2.2 Data Acquisition Application for On-the-move Scenario

*1. Description of the application functionality*

The devices used in this prototype in the on-the-move environment are connected to a reading station (a mobile phone carried by the patient). In the reading station, the reading software (a Java MIDlet) is installed, which is responsible to communicate with the medical devices and read the data from them over a wireless Bluetooth channel. The communcation protocols used for the communication with each one of the devices usually are proprietary protocols which are disclosed by the vendors, enabling the development of the reading software on the mobile phone.
The purpose of the J2ME application described in this section is to integrate the devices located in the patient’s personal environment to the HEARTFAID platform via the acquisition of the relevant data and their transmission to the intermediate gateway over an IP connection (GPRS/3G). As it was described in D14 this intermediate gateway is responsible for the creation of the XML messages and their final transmission to the platform.

*2. Installation of the application*

The application is implemented in J2ME and compiled into a JAR file. In order to install this JAR file on the mobile phone, the user has to transfer the JAR file from the PC to the mobile phone. The transfer procedure will depend on the mobile phone. There are several options to do this:

- Explicit physical connection via USB cable
- Direct InfraRed (IR) connection
- Direct Bluetooth connection

The first option obviously requires a USB cable, which will normally be provided by the phone manufacturer, and will come with the phone itself, together with some software. More information can be found in the phone's manual or the phone manufacturer's web site.

If the phone and the PC both have InfraRed (usually the phone will have, but the PC won't unless it's a laptop), the file can be transferred via InfraRed. The procedure is quite simple in Microsoft XP, and more details on establishing a connection can be found either on the PC's help file, or at Microsoft's web site.

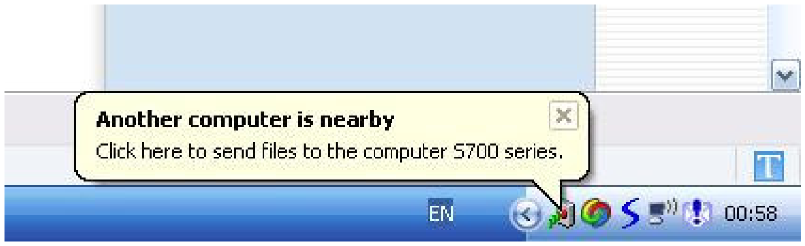

Another computer is nearby
Click here to send files to the computer S700 series.

**Figure 24: Notification of connection over infrared (WindowsXP)**

Once the connection is established, the files can be easily sent to the mobile phone simply by lining up the mobile phone with the PC's IR receiver, and the PC should suddenly notify the user that "another computer is within range" (see image, above). By clicking the pop-up message, a window will open asking what file the user likes to transfer. The user has to locate the midlet (the .jar file), click on it, click 'Send' and the file will be sent. Once successfully uploaded, the phone will usually ask what the user wants to do with it. Normally it will give an option to save it in a specific location (usually a folder name). The user has to choose an appropriate location, and the file will be saved. To use it, the user has to select it from the phone's menu, and it will be run by the phone's J2ME implementation.

In case of a Bluetooth enabled mobile phone, a connection has to be established between the mobile phone and PC. However, although most laptops these days come with Bluetooth pre-installed (certainly most of the wireless Centrino-based ones do), desktop PCs usually do not. This problem can be solved by buying a USB Bluetooth adapter, which is a simple USB pen-drive-style dongle that sits in the USB port and instantly adds Bluetooth to the PC (note, however, that unlike USB pen-drives the supplied Bluetooth drivers have to be installed first in order for the dongle to work).

Once both devices are Bluetooth-enabled, the connection can be established. This is easily done, as Bluetooth devices can be made to search for one another. Each device has to be given a name beforehand, then the user selects “search for device” in either the phone or PC, and that device will search for other Bluetooth devices within the local area (between 10m - 100m depending on the Bluetooth specification being used). All devices located will be presented to the user, and the user can then select the one he needs. Depending on the security settings selected, either the PC or phone will ask for a passcode. Enter a number (any number will do), and the other device being connected to will then also ask for a passcode. Again, enter the same number (the two numbers must match, but do not need to be specified in advance). Once the passcodes have been entered, the connection is established, and file transfer can begin.

To transfer files from the PC to the phone, the user can use Windows Explorer from PC (see image, below). The user has to simply locate the file, right click on it, click 'Send To' and then select 'Bluetooth' and the device will then appear as a menu item. The user has to select it, the file will be uploaded to the phone. Note that in some circumstances, depending on the settings, the file can be sent this way without establishing a connection first - the devices will establish the connection themselves automatically.

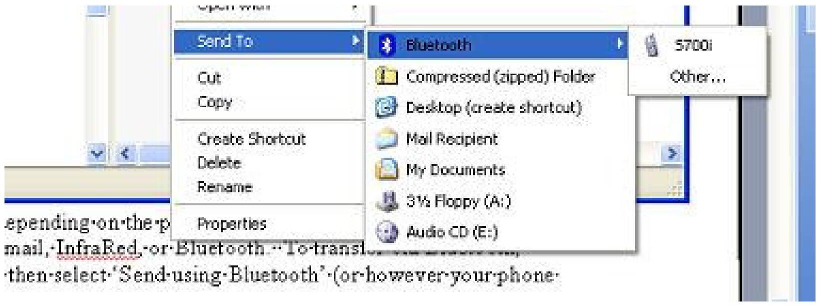


**Figure 25: Transferring the file from the PC to the phone over Bluetooth**

## 3. *Application User's guide*

The application's operations can be demonstrated through the flow design which is depicted in the following Figure.

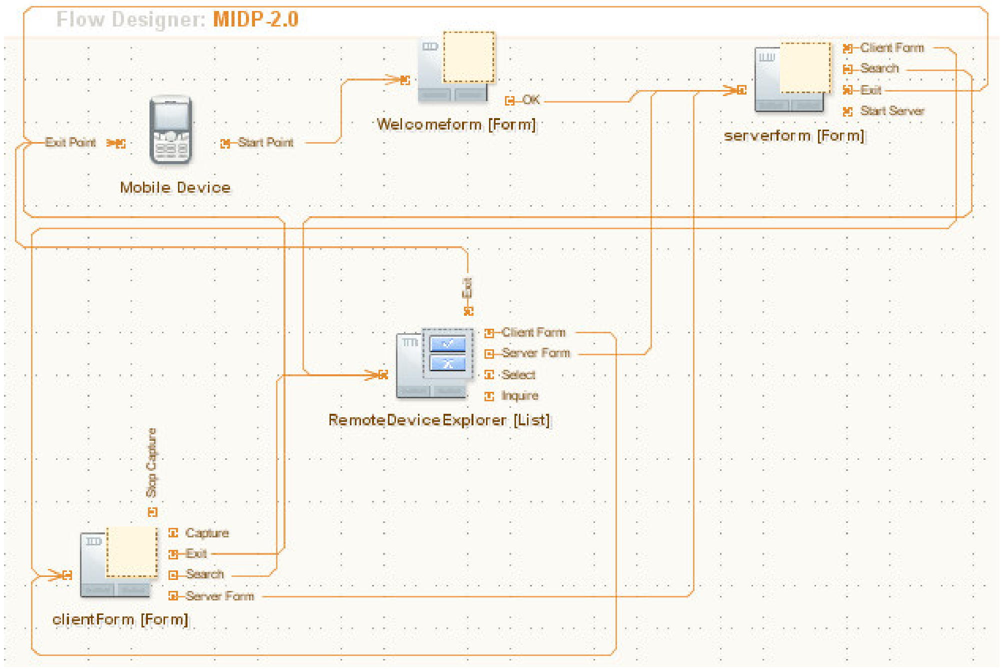


**Figure 26: Flow design of the J2ME application for the on-the-move environment**

When the execution of the application starts, a welcome screen (Welcomeform) is presented to the user prompting him/her to enter the patient ID. This patient ID will be used thereafter for the transmission of any data to the HEARTFAID platform.
Once the ID is inserted the core part of the application starts executing. This comprises two parts:

1. The server part (managed by the serverForm), which enables the mobile device to act as a server to the bluetooth communication. According to this way of operation, the mobile device sets up a bluetooth btspp (Bluetooth Serial Port Profile) service described by a unique identifier UUID (in order to be traceable by client devices) and waits for incoming connections.
2. The device and service discovery part (managed by RemoteDeviceExplorer), which enables the mobile device to search for other devices and services in the bluetooth environment.
3. The client part (managed by the clientForm), which enables the mobile device to act as a client to the bluetooth communication. According to this way of operation, the mobile device can discover and connect to bluetooth services in the neighbourhood.

Upon the insertion of the ID, the serverForm appears which has the following options (Commands in the J2ME terminology):
*Client Form:* When this command is selected, the clientForm appears
*Search:* When this command is selected, the RemoteDeviceExplorer List appears
*Start Server:* When this command is selected, the bluetooth server on the mobile device is started and the device is ready to accept client connections. Once a measurement is read via the bluetooth connection, it is transeferred over an http connection to the intermediate gateway.
*Exit:* When this command is selected, the application exits.

As mentioned before, the clientForm enables the device to act as a client to other devices in the bluetooth environment. In order to use the functionality of the client form, the user has to first select a device in the environment. In order to do this:

1. the user selects the *Search* command from the clientForm.
2. the RemoteDeviceExplorer List which will provide the functionality for the selection of a device.
3. the user selects *Incquire* from the RemoteDeviceExplorer options and the device inquiry process begins. The user has to wait for some time in order for the device inquiry process to be completed. Upon its completion, the available devices are displayed as items in the RemoteDeviceExplorer List.
4. the user selects a device in the List and selects the *Select* option from the RemoteDeviceExplorer options. The device is now selected.
5. the user selects the *Client Form* command and switches to the clientForm.

Once the device is selected following the procedure described above, the mobile device can start the communication with it as a client in order to acquire the data. This is achieved using the *Capture* command in the clientForm. Once this command is selected, the mobile device communicated with the selected device and the mobile phone reads in the data according to the protocol supported by the selected device. The reading process can be stopped using the *Stop Capture* command from the clientForm. Once a measurement is read via the bluetooth connection, it is transeferred over an http connection to the intermediate gateway.

## *4. UML Diagrams of the Application*

The main conceptual diagram of the application is depicted in the following figure.

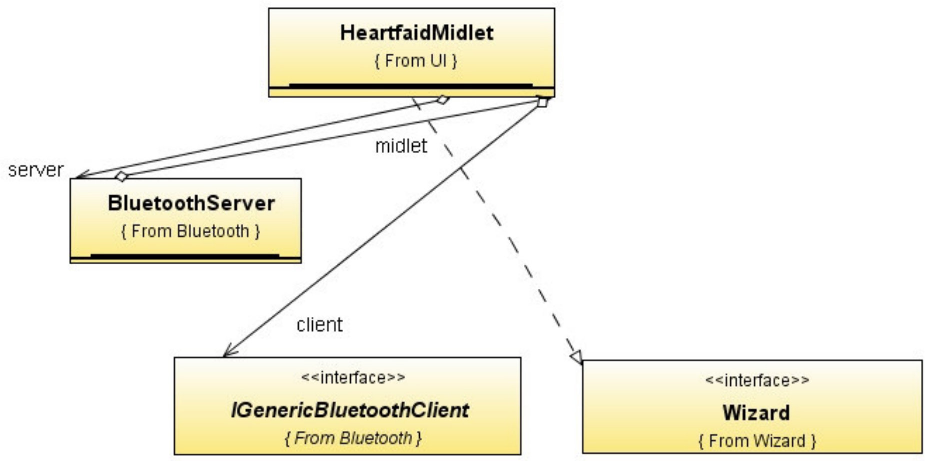


**Figure 27: Main UML diagram of the application**

The main class of the application is the HeartfaidMidlet class. This class includes the User Interface functionality of the application and communicates with other classes and packages in order to execute the logic of the application. The main classes which the HeartfaidMidlet communicates with is the BluetoothServer class, which implements the server side bluetooth logic in order to enable remote bluetooth devices to connect to the mobile device and transfer data to it. The IgenericBluetoothClient interface represents the abstract functionality of a client in the bluetooth channel and consequently it has to be implemented by every class that is destined to act as a client to a medical device in order to acquire data from it. The abstract class Wizard (which implements the interface IWizard) represents the functionality of needed in order to acquire data from a medical device when the mobile device acts as a server and has to be implemented into by every class that will handle the data received by the BluetoothServer.

In the following figure, a more detailed view of the bluetooth classes can be seen. These classes are organised in a package called Bluetooth. The most important class of this package is the *BluetoothManager* class, which is responsible to perform al the Bluetooth-related functionality (e.g. start the inquiry for remote devices and services, search for the available services running on a remote device etc.). As mentioned above, the interface IgenericBluetoothClient describes the abstract functionality of the client in the bluetooth environment. The most characteristic operation is the method *Send*, which is responsible to send a stream of bytes to a remote device to the service defined by a specific *ServiceRecord*. Finally the class *BluetoothServer* implements all the functionality of the bluetooth server. Characteristic methods of this class are the methods: *defineService* which initiates the service, *updateServiceAvailability* which manages the availability of this service taking into account the maximum number of clients allowed to connect to it, *acceptClientConnections* which blocks waiting for a client to connect, *readData* which reads the incoming data, *activateWizard* which instantiates the appropriate wizard that will handle the incoming data and *endComm* which ends the communication with the connected client.

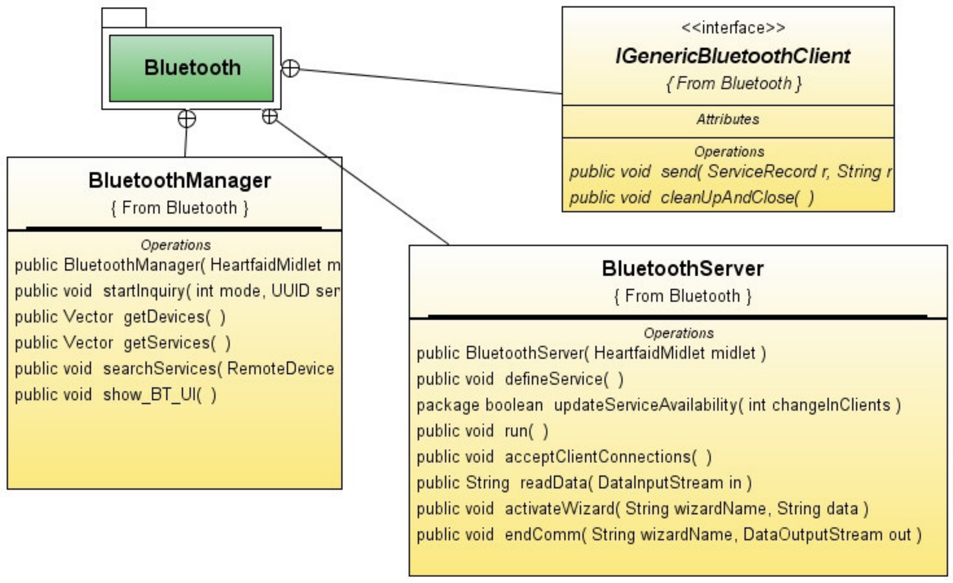

Bluetooth
<<interface>>
IGenericBluetoothClient
{ From Bluetooth }
Attributes
Operations
public void send( ServiceRecord r, String r
public void cleanUpAndClose( )
BluetoothManager
{ From Bluetooth }
Operations
public BluetoothManager( HeartfaidMidlet m
public void startInquiry( int mode, UUID ser
public Vector getDevices( )
public Vector getServices( )
public void searchServices( RemoteDevice
public void show_BT_UI( )
BluetoothServer
{ From Bluetooth }
Operations
public BluetoothServer( HeartfaidMidlet midlet )
public void defineService( )
package boolean updateServiceAvailability( int changeInClients )
public void run( )
public void acceptClientConnections( )
public String readData( DataInputStream in )
public void activateWizard( String wizardName, String data )
public void endComm( String wizardName, DataOutputStream out )

**Figure 28: UML Diagram of generic bluetooth classes**

Figure 29 illustrates the way that device specific bluetooth clients can be developed by implementing the *IGenericBluetoothClient* interface. In the figure, the class diagram containing the two bluetooth client classes (for the Nonin pulse oximeter and for the FRWD sports computer) is depicted. Each client class contains device specific methods, as well as implementation of the *IGenericBluetoothClient* interface methods (*send* method which sends the data and *cleanupAndClose* method which finalises the connection from the client side (terminates all the threads, closes the input and output streams of the connection and finally closes the connection).

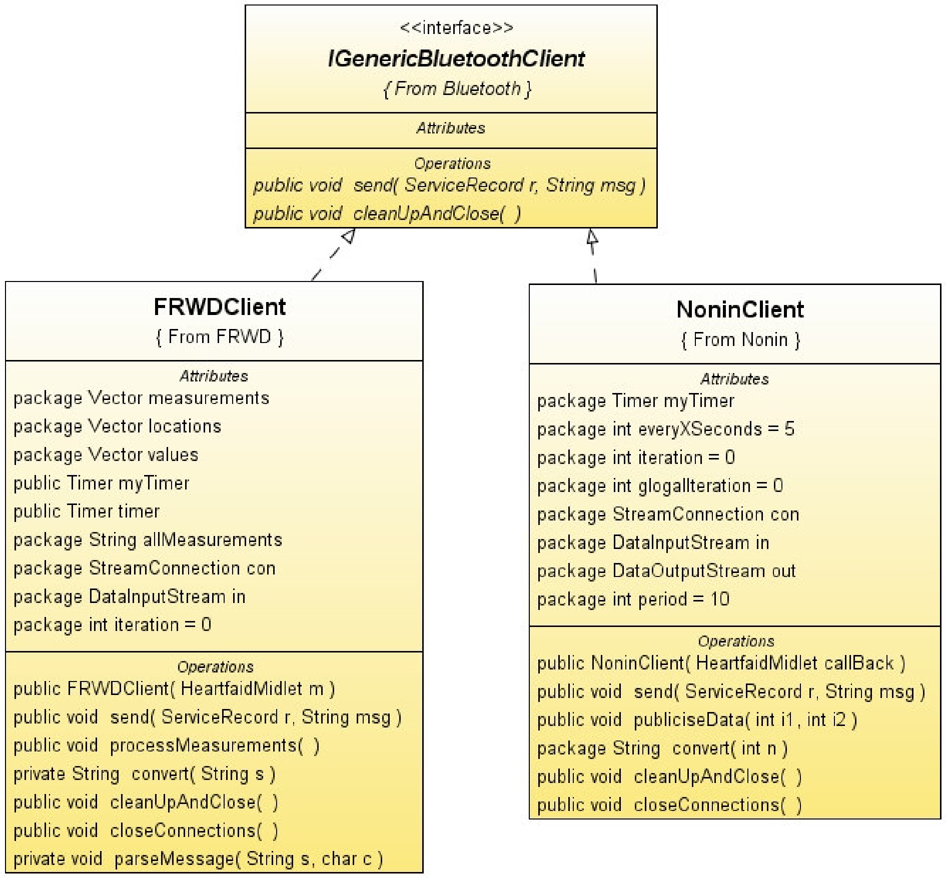

<<interface>>
IGenericBluetoothClient
{ From Bluetooth }
Attributes
Operations
public void send( ServiceRecord r, String msg )
public void cleanUpAndClose( )
FRWDClient
{ From FRWD }
Attributes
package Vector measurements
package Vector locations
package Vector values
public Timer myTimer
public Timer timer
package String allMeasurements
package StreamConnection con
package DataInputStream in
package int iteration = 0
Operations
public FRWDClient( HeartfaidMidlet m )
public void send( ServiceRecord r, String msg )
public void processMeasurements( )
private String convert( String s )
public void cleanUpAndClose( )
public void closeConnections( )
private void parseMessage( String s, char c )
NoninClient
{ From Nonin }
Attributes
package Timer myTimer
package int everyXSeconds = 5
package int iteration = 0
package int glogalIteration = 0
package StreamConnection con
package DataInputStream in
package DataOutputStream out
package int period = 10
Operations
public NoninClient( HeartfaidMidlet callBack )
public void send( ServiceRecord r, String msg )
public void publiciseData( int i1, int i2 )
package String convert( int n )
public void cleanUpAndClose( )
public void closeConnections( )

**Figure 29: UML Diagram for FRWD and Nonin client bluetooth modules**

Finally, Figure 30 demonstrates the way that device specific data processing logic in case of bluetooth server can be developed in terms of classes extending the class Wizard. In the figure two classes (for UA767-PBT and for UC-321PBT) are shown. Once the data are received through the bluetooth channel, the HeartfaidMidlet fowards them to the appropriate Wizard via the invocation of the *setData* method. The implementation of each Wizard is device specific in the sense that it implements the *handleData* method according to the medical device communication protocol.

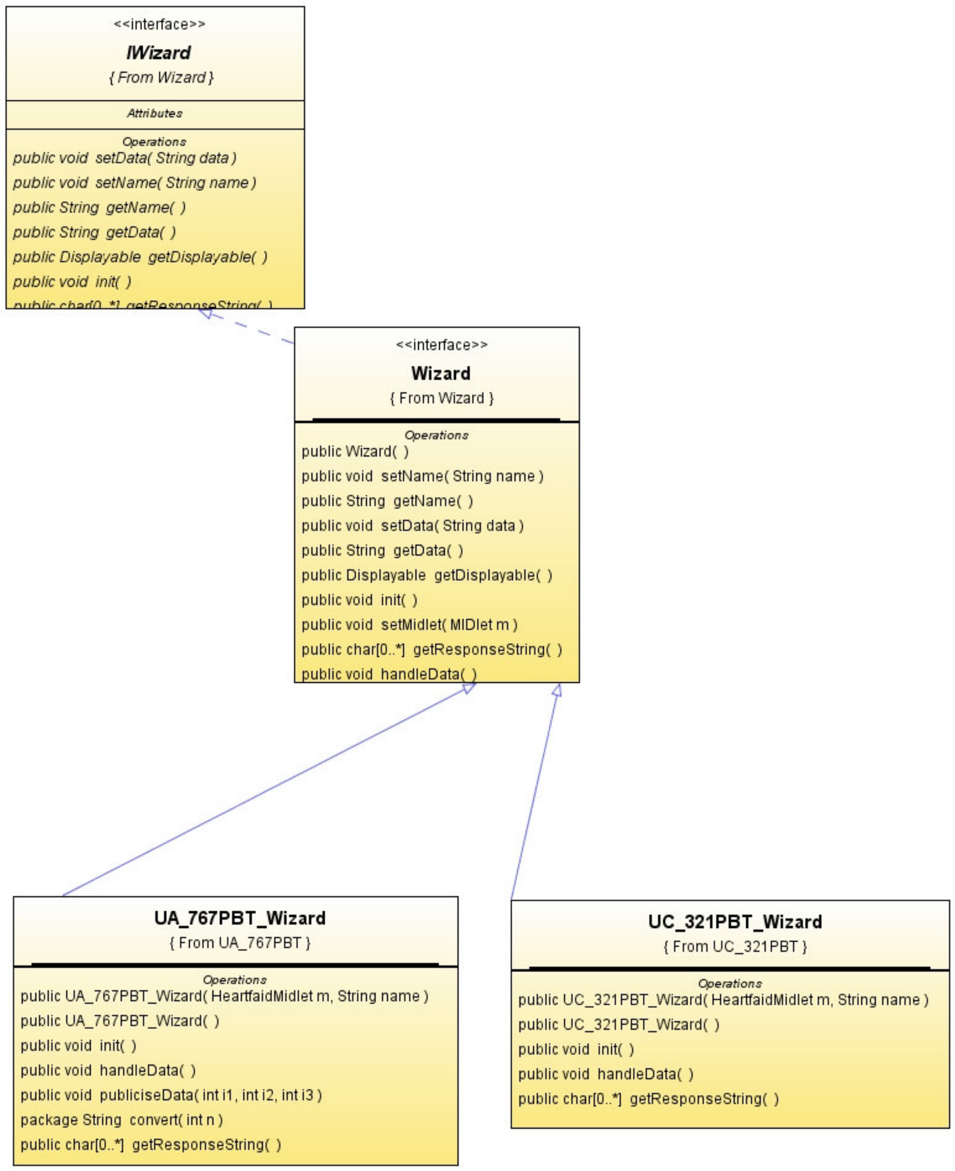

<<interface>>
IWizard
{ From Wizard }
Attributes
Operations
public void setData( String data )
public void setName( String name )
public String getName( )
public String getData( )
public Displayable getDisplayable( )
public void init( )
<<interface>>
Wizard
{ From Wizard }
Operations
public Wizard( )
public void setName( String name )
public String getName( )
public void setData( String data )
public String getData( )
public Displayable getDisplayable( )
public void init( )
public void setMidlet( MIDlet m )
public char[0..*] getResponseString( )
public void handleData( )
UA_767PBT_Wizard
{ From UA_767PBT }
Operations
public UA_767PBT_Wizard( HeartfaidMidlet m, String name )
public UA_767PBT_Wizard( )
public void init( )
public void handleData( )
public void publiciseData( int i1, int i2, int i3 )
package String convert( int n )
public char[0..*] getResponseString( )
UC_321PBT_Wizard
{ From UC_321PBT }
Operations
public UC_321PBT_Wizard( HeartfaidMidlet m, String name )
public UC_321PBT_Wizard( )
public void init( )
public void handleData( )
public char[0..*] getResponseString( )

**Figure 30: UML Diagram of Wizard interfaces and classes for server side bluetooth modules (UA767BPT and UC321PBT)**

## 4.3 Data Transmission Mechanisms and Systems

### 4.3.1 Description of the XML Messaging schema

The XML messaging schema includes measurements or sets of measurements in the form of "observations". In order to conclude to a flexible and extensible schema, the "composite observation" type was introduced according to the Composite Design Pattern. The composite observations are actual placeholders

(Substitution-Groups in xsd terminology) of the observation elements and can carry other observations inside them. The carried observations can be themselves composite or not thus creating the typical recursive structure of a composite object. Any meta information (besides the one already encoded like address, timestamp etc) can be supplied in a completely unstructured way by adding name/value parameter elements to any observation (composite or not). These parameters have an attribute name and their value is expressed as the node text. Another attribute type can be used to give a hint on the type to be used for parsing the parameters (possible values are string|int|float|boolean). Only leaf (non composite) observations can carry measurements and a gwph-message structure can contain either a composite or a non-composite observation.

The above description can be explained in an easier way in the following figure. The gwph-message can contain a *compositeObs* element or an *obs* element. The *compositeObs* element can contain other *compositeObs* and/or *obs* elements. Only the *obs* elements can carry measurements to be transferred to the HEARTFAID platform. In Appendix C, various kinds of valid example XML messages are given.

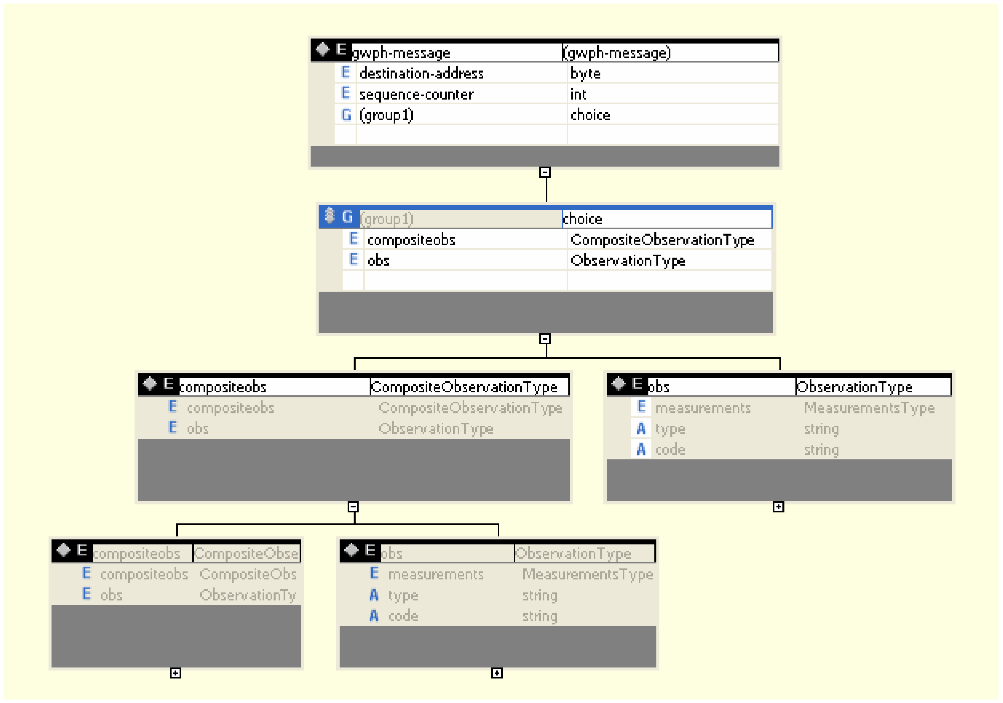

E gwph-message (gwph-message)
E destination-address byte
E sequence-counter int
G (group1) choice
G (group1) choice
E compositeobs CompositeObservationType
E obs ObservationType
E compositeobs CompositeObservationType
E compositeobs CompositeObservationType
E obs ObservationType
E obs ObservationType
E measurements MeasurementsType
A type string
A code string
E compositeobs CompositeObse
E compositeobs CompositeObs
E obs ObservationTy
E obs ObservationType
E measurements MeasurementsType
A type string
A code string

**Figure 31: Block diagram of the gwph-message structure.**

### 4.3.2 Functionality of the intermediate gateway for on-the-move scenario

As it has been mentioned before, the intermediate gateway supports the XML message creation and transmission to the HEARTFAID platform for the case of the on-the-move scenario. There are a number of small, open source XML parsers available including NanoXML, kXML and TinyXML. The NanoXML and TinyXML are both natively tree-based parsers while kXML is a more sophisticated pull/event-based parser. kXML is much larger than NanoXML and TinyXML. All these parsers are becoming more complete over time with features such as DTD validation and Java interfaces for both tree and event-based parsing; of course all these features come at a price: memory footprint. Moreover, the volume the transmitted data increases significantly in the case of direct transmission of XML messages from the mobile device to the HEARTFAID platform.
For the above two reasons, the approach of an intermediate gateway was followed. This gateway executes ASPX service components which are responsible to generate XML messages based on the data sent to them from the mobile devices and transmit these messages to the HEARTFAID platform. This feature will be demonstrated in the following figure.

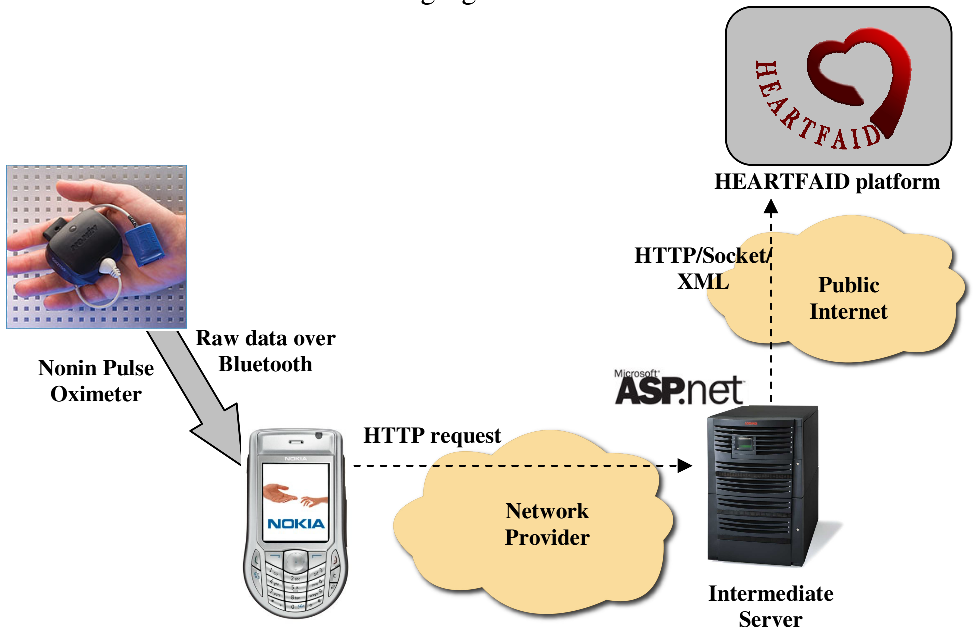


**Figure 32: The functionality of the intermediate gateway**

In the figure above, we assume that the mobile device receives over Bluetooth data form the Nonin Pulse Oximeter. This medical device produces a combined Heart Rate and SpO2 measurement. The data acquisition application running on the mobile device transmits these measurements over HTTP as parameters in an HTTP request. The following snippet of code demonstrates the sending of this request.

```
…..
StreamConnection c = null;
InputStream s = null;
try {
        StringBuffer b = new StringBuffer();
        String url;
        url="http://<address>/heartfaid_gw.aspx?phone=1&bt=1&ts="+ts+"&me
        asurements=2&a1=-&t1=PUL&v1="+i1+"&u1=mmHg&a2=-
        &t2=OXY&v2="+i2+"&u2=percent";
        c = (StreamConnection)Connector.open(url);
        s = c.openInputStream();
        int ch;
        // wait for the gateway response
        while((ch = s.read()) != -1) {
           b.append((char) ch);
        }
}catch(IOException ioe){}
```

In the above code, the mobile device opens a connection to the address of the gateway and provides a list of parameters in the HTTP request. The “*phone”* and “*bt”* parameters serve as parameters that will define the unique address of the sensor in the side of the platform, while the *ts* is calculated a the mobile phone side and represents the timestamp of the measurement. The parameter “*measurements”* represents the number of the measurements. After this parameter, a set of parameters follow *aX, tX, vX, uX*, which represent the alarm (in case it is generated at the mobile devices side, otherwise the default is “-“, or it may not be transferred), type, value and units of each measurement.
When this request reaches the intermediate gateway, the aspx component parses the above parameters and creates the following XML message containing a simple observation (see Appendix C), to be sent to the HEARTFAID platform as a message in an HTTP request or as a stream of bytes over a plain Socket connection:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<gwph-message xsi:noNamespaceSchemaLocation="GWPH-protocol-
v1.10.xsd" xmlns="http://www.synapsis.com/heartfaid/ph/dom"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <destination-address>0</destination-address>
  <sequence-counter>0</sequence-counter>
  <obs type="" code="0">
    <address descr="xml-gw" id="1">
      <address descr="phone" id="1">
        <address descr="bt" id="1" />
      </address>
    </address>
    <timestamp>2007-06-08T09:49:13</timestamp>
    <measurements>
      <measurement>
        <alarm>-</alarm>
        <type>PUL</type>
        <value>99</value>
        <units>mmHg</units>
      </measurement>
      <measurement>
        <alarm>-</alarm>
        <type>OXY</type>
        <value>98</value>
        <units>percent</units>
      </measurement>
    </measurements>
  </obs>
</gwph-message>
```

### 4.3.3 Alert notification system

The alert notification system was developed to offer sophisticated end-user services and a medium of immediate feedback to patients and clinicians. The SMS gateway, developed for the notification system, is a middleware application that accepts requests on a TCP port. The gateway was developed as an external module to HeartFAID's middleware with the intention to disengage the two and run them on separate boxes, if so required. To this end, the module accepts XML-based requests, instead of using a language specific API.

The SMS routing gateway is a multi-threaded server. As mentioned previously, it can be accessed through a TCP socket, on a predefined host and port. The XML-based protocol that was adopted for communication allows any number of clients to interact with the gateway. Each client is responsible for obtaining credits for the gateway, which define the amount of SMS messages the client may send.

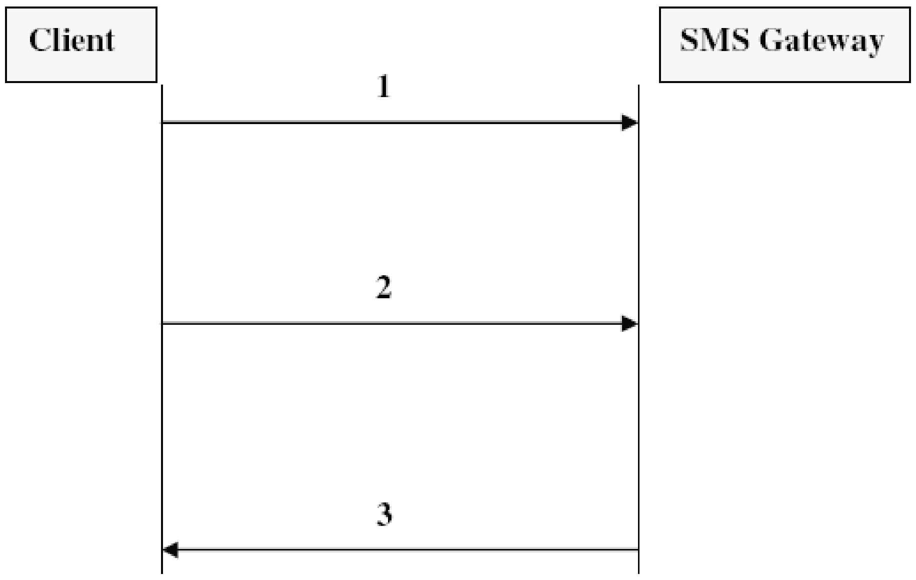


**Figure 33: Client / Server communication on SMS Gateway**

Figure 33 illustrates the interaction between the gateway server and a client as a 3-stage process:

a. The client opens a TCP/IP socket connection with the gateway.
b. The client sends through the socket, a request in XML format.
c. The gateway sends and XML-based response to the client.

The request that clients submit to the SMS gateway has the syntax displayed in Listing 1. The complete XML schema definition is presented in Appendix A.

```
<XML>
   <REQUEST>alphanumeric</REQUEST>
   <CUSTOMER>
      <USERNAME>alphanumeric</USERNAME>
      <PASSWORD>alphanumeric</PASSWORD>
   </CUSTOMER>
   [request-body]
</XML>
```

***Listing 1:*** *Request to SMS Gateway*

Elements XML, REQUEST and CUSTOMER are mandatory in every request. The [request-body] depends on the value of element REQUEST.
In order for the SMS gateway to process a request it must first authenticate the client that submits it. The credentials of the client consist of a **username** and a **password** that are issued for every valid client of the gateway. The credentials are part of the client's request, enclosed ender element CUSTOMER, in child-elements USERNAME and PASSWORD.
The requirements of the project specified the need for two types of server requests, one for dispatching an SMS and one for retrieving the number of available credits, for the requesting client. The value of element REQUEST defines the type of instruction submitted to the gateway. The types of requests and a short description for each one are displayed in the table below.

| **Request Type** | **Description** |
|---|---|
| SENDMESSAGES | Sends the message specified in the request body to a list of recipients. |
| GETCREDITS | Returns the number of available credits for a specific client. |

## 4.4 HEARTFAID Electronic Health Record (SYNAPSIS)

An **Electronic Health record** (EHR) will be the comprehensive view on all the health information available for the patient. The EHR is not a single and complex application but the result of a series of data **acquisition** and **transformation activities**. In particular, in the Heartfaid EHR the data will be gathered from different and heterogeneous producers such as specific and specialistic Electronic Medical Record (EMR), Remote Data Acquisition networks (RDA) connected to Ambient Intelligence (AmI) platforms used for telemonitoring and other components of the Healthcare Information Systems involved in the Heartfaid platform. Transformations activities are performed in order to produce one or more **reports** that show data in a more suitable and "easy to view, read and interpret" form.
The outputs of these activities are collected in repositories and the produced information is registered by its *meta-data* in proper registries that expose search and inquiry services.

### 4.4.1 Definitions and standards

In general, according to the Medical Records Institute[1], the EHR, is compared to another more general classification: the Electronic HealthCare Record (EHCR). The institute also divide the EHCR in five levels that can be distinguished in:

- The Automated Medical Record (AMR): paper-based record with some computer-generated documents.
- The Computerized Medical Record (CMR): makes the AMR documents electronically available.
- The Electronic Medical Record (EMR) restructures and optimizes the documents of the previous levels ensuring inter-operability of all documentation systems.
- The Electronic Patient Record (EPR) is a patient-centred record with information from multiple institutions.
- The Electronic Health Record (EHR) adds general health-related information to the EPR that is not necessarily related to a disease.

To guarantee interaction among the different levels of the EHCR and the interoperability between the EHCRs and the other components of the Healthcare Information System, since the nineties, a very large set of standards, work-flows and integration guidelines have been introduced by healthcare related actors: healthcare organizations, healthcare institutes and standardization institutions in general. Some, and the most important, of these standard are represented by **Healthcare Level 7** organization (HL7), the **Digital Imaging and Communications in Medicine** (DICOM) standard from American College of Radiology (ACR) and National Electrical Manufacturers' Association (NEMA); in addition we can found the **Integrating Healthcare Environments** (IHE) initiative, leaded by healthcare professionals and industry to improve the way computer systems in healthcare share information. This initiative uses standards

1 Medical Records Institute: http://www.medrecinst.com/ - July 25, 2006 Paper

defined by other actors and defines the guidelines in one or more documents that are known as *Technical Frameworks*. One technical framework serves as a consensus document of how to think about, discuss, and successfully overcome medical information system integration problems by using existing standards and tools such as HL7, DICOM and so on. Each year the new version of the technical frameworks defines some new *Integration Profiles* in order to realize key interoperability tasks between systems that are identified as **actors** for the profile itself.

Following these guidelines, the technological architecture that we propose for the implementation of the EHR, involves the following key actors:

1. A centralized patient identity management system that is able to manage and resolve patients identifications between different systems: a **Master Patient Index** (MPI) with **Patient Identity Cross Referencing** (PIX) capabilities (In the following simply called MPI).
2. Distributed information **Repository** that contains information, that can be heterogeneous.
3. A centralized **Registry** archive where all patient related information *meta-data* are stored, with also a reference to the real information, that can be queried and permits to retrieve these information as needed.

The IHE profile that describe the technologies, standards and the transactions to obtain this goal is **Cross-Enterprise Document Sharing** (XDS and the newest XDS for Medical Summaries, XDS-MS) that enables a number of healthcare delivery organizations belonging to a clinical affinity domain (e.g. a community of care) to cooperate in the care of patient by sharing clinical records in the form of a document as they proceed with their patients' care delivery activities. This profile is based upon ebXML standard of the Registry, SOAP messages, HTTP and SMTP protocols for data exchanging.
By using the two key actors any healthcare related system can, according to transactions that are been defined in the interoperability standard, produce (**Producer**) and retrieve (**Consumer**) all needed information.

#### 4.4.1.1 Producer

In this scenario all systems that act as information producer:

- Should use the MPI to assure that the produced information belongs to the entity that is *shared* by all the others systems, data and mainly the unique identifier.
- Should store on one or more Repository the produced information with its *meta-data* using the entity's unique identifier[2].
- Should register in the unique Registry the information's *meta-data* and the location of the information in the Repository[3].

2 Please note that the uniqueness is mandatory in this version of the profile and not represents any limitation.

#### 4.4.1.2 Consumer

In the same scenario a system that wants to retrieve information:

- Should use the MPI to assure that know the unique identifier of the entity.
- Should query the Registry by using one or more information's *meta-data* constraints in order to obtain zero or more links, as URLs, to documents that are stored into the repositories.
- Should retrieve the information contacting directly the repository with the URL.

### 4.4.2 Security

It's evident that the use of secure communications between all the actors represents an important requirement in order to assure the confidentiality of the data exchanges. This can be easily done by assuring that all communications are made with the use of a secure channel of communication over the TCP/IP protocol: the well known Secure Socket Layer Version 3 (SSLv3) or the **Transport Layer Security** (TLS). The communications are made by using the version with almost best security defined by the protocol, also knows as *mutual authentication*, when both the communications' endpoints are recognized and trusted with its personal and unique certificate.
It's also evident that it is fundamental to assure the trust of the originator (authentication), the integrity and the non-reputability of this information. For these reasons, for healthcare related information production and consuming, we will plan the use of the digital signature of the information. This requirement will be also enforced by IHE in the next years in new versions of the Technical Frameworks.

### 4.4.3 Structured data

Another fundamental aspect in the realization of the EHR is represented by the need to use structured data for the healthcare information production. Structured documents, in fact, can be easily processed by a *Consumer actor* in order to discover, elaborate and manage the information contained in that documents. This kind of data has been defined by HL7 with the Clinical Document Architecture (CDA) standard, that use the XML standard for the data representation; in particular, HL7 uses the level 2 of the CDA standard (CDA2): this version requires that data must be structured and coded. With the use of the XML as information representation language for the CDA, the goal of HL7 is ensuring both the human readability and machine processability of the content. In fact it is evident that the use of coded data to represent concepts is necessary for assure the portability of the information. This can be done by using healthcare coding

3 In effect the last operation, as defined by IHE integration profile, is done directly from the Repository that register the information's meta-data subsequently to a store operation initiated by the information's source.

systems for the information such as these provided by **Logical Observation Identifiers Names** and **Codes** (LOINC) and by **Systematized NOmenclature of MEDicine Clinical Terms** (SNOMED-CT).

# 5 Testing of the prototype

## 5.1 Tests related to Clinical Environment

### 5.1.1 Clinical Environment Data Acquisition application

HEARTFAID's Clinical Environment client application was implemented having in mind the ease of use by the clinical personnel. The basic functionality of the software which was tested consists of three actions, namely the loading of the data from files stored in the hard disk of the PC, creation and storage of the XML messages to be sent to the HEARTFAID platform and finally the transmission of these messages to the platform. The tests that were performed in the Clinical Environment Client are the following:

1. Creation and Deletion of the Temporary "Work folder"

**Methodology**: When the application starts, a working folder named "Work" is created at the top level of the directories structure in order to store the XML messages. During the application shut down, all the files in the work directory, as well as the directory itself are deleted.

**Results**: The "Work" folder is created in the correct location. When a message is created, it is stored under this folder and during the application shut down all the messages as well as the folder are deleted.

2. Loading of the biomedical data from files

**Methodology**: The purpose of this test is to verify that the data stored in the respective files are parsed successfully. In order to achieve this, various messages are reported in the status bar at the bottom of the application's window (see Figure 8 ). Moreover, for each device, the relevant data are entered in the respective fields in the device's screen. In order to cope with erroneous situations, corrupt version of the original test files were created, so in these cases the parsing of the files could not be successfully performed.

**Results**:

a) When the data are loaded successfully, the values are displayed on the screen in the respective text fields. Also, the phrase "Data Loaded Successfully" is displayed in the status bar.
b) When there is an error in data loading procedure (mainly due to incorrect parsing of the file) the message "ERROR parsing the file" is displayed and no values are places in the text fields.

3. Creation and storage of the XML messages

**Methodology**: The purpose of this test is to verify that the XML messages to be transmitted are sucessfully generated and stored. For these reason the messages are stored in the "Work" folder and are verified towards the defined XML schema definition.

**Results**: The XML messages are successfully generated and stored in the "Work" folder. The phrase "XML message created successfully" is displayed in the status bar. For each one of the files, the verification against the defined XML schema was successful.

4. <u>Transmission of the XML messages to the platform</u>

**Methodology**: The purpose of this test is to detect whether the XML messages are successfully transmitted to the platform and, in erroneous cases, to report the errors to the user. In order to achieve this, various messages are reported in the status bar at the bottom of the application's window (see Figure 8 ). In case of successful transmission, the HEARTFAID platform's response should be printed out in the console of the application, and the phrase "Data Transmitted successfully" should be seen in the status bar. In case of an error, the phrase "Transmission Error" should be seen in the status bar. In order to achieve this situation, the transmitting PC is disconnected from the network or the platform is disconnected; hence the connection to the platfrom cannot be established.

**Results**: Upon successful transmission of the messages to the platform, the phrase "Data transmitted successfully" is displayed in the status bar and the response of the HEARTFAID platform is printed out in the console of the application. In case of transmission error (e.g. disconnected client), the phrase "Transmission ERROR" is displayed on the status bar.

### 5.1.2 Clinical Environment eCRF application

The eCRF prototype has already been preliminarily tested by clinicians. Approximately 50 eCRF forms have been entered into the system. Comments on the usefulness of the system are collected and will hopefully lead to a more intuitive user interface.

## 5.2 Tests related to Personal Environment

### 5.2.1 Home Environment tests

HEARTFAID's Home Environment client offers great versatility and contains a number of different interacting modules that seamlessly automate the active procedures. Due to the software's modular design, distinct tests were planned for each of the units. The methodology and results of each test are detailed below.

1. <u>Configuration update</u>

   **Methodology**: It is evident from the descriptions and screenshots of 4.2.2 that the application is fully configurable through the GUI. The application follows the MVC pattern, so changes in the front-end are propagated to the controller, where respective processing takes place and updates are circulated to all components. Items in the front-end were changed one by one to test their effect on the application.

a. Interface update

**Results**: Every control element of the GUI is "wired" into the GUI listener and changes are propagated to the controller. For this phase of the test, emphasis was put on modifying mandatory fields and observing the changes of state. Other than that, the User Preferences were examined individually to observe their effect in the "Activity View".

b. File persistence

**Results**: At application start-up, configuration file **nurseathome.properties** is read and all front-end fields are populated accordingly. The reverse procedure is also completed successfully; whenever the "Apply Changes" button is clicked in the front-end, the persistent configuration file is updated fully.

2. Modes

**Methodology**: In section 4.2, there was an extensive description of the different modes of operation of the software and the significance of their succession. The default mode is assumed to be "Stand By", since that's the initial state of the application after installation. The front- and back-end state was checked at each mode, to validate conformance to specifications.

a. "Stand By" mode

**Results**: The front-end controls to quit and immediate data dispatch are disabled, as expected. The correct status indicator appears on all three positions – status box, bottom left and title bar – and the Message Box displays the respective message. In the back-end, the logs show that the application is active and receiving data from the sensors. If auto-start is set to OFF (default), the data is discarded.

b. Transition to "Configuration Error"

**Results**: If one of the mandatory fields is removed, the application passes to the "Configuration Error" mode, a fact that is immediately reflected to the three status indicators. In the "Activity View", all three buttons of the Status Box are disabled, and a warning message is displayed in the Message Box. Effectively, the user can not use the application; s/he can only apply different configuration settings or shut it down. Since the settings are persisted on permanent storage (Test 1b), re-starting the application does not affect the mode, until the configuration is corrected.

c. Transition to "Running"

**Results**: This transition can only occur from "Stand By", in one of two ways:

ii. clicking on the "Start" button in the Status Box,
iii. selecting "Yes" in the auto-start preference.

In the front-end, all controls outside the "Activity View" are disabled. In the Status Box of the "Activity View", "Start" button is disabled and "Quit" and "Send Now!" are enabled. In the Timer of the "Activity View", count-down commences and in the Message Box, short informative messages appear, as the application skips through the steps of acquisition, encapsulation and transmission.

In the back-end, the application is "listening" for sensor input, and every 60 seconds, polls through the list of sensors that have not yet delivered a measurement and requests input. The measurements are stored in memory for the duration of this mode. The data encapsulation phase may commence after data acquisition in various ways, which are examined in Test 3.

3. Data acquisition interceptors

   **Methodology**: Various methods are employed to pass control from the data acquisition phase to the data encapsulation phase of the "Running" mode. Each of these methods is used repeatedly, to ensure that the behaviour of the application is stable and consistent.

   a. All sensors respond

   **Results**: Through the application's logs, make sure that all sensors take measurements and send them to the application. The data acquisition thread stops gracefully and the controller activates the data encapsulation phase. In the front-end, the message "All your medical data have been gathered…" appears in the Message Box.

   b. "Quit" interceptor

   **Results**: By pressing the "Quit" button, while the data acquisition is running, the thread stops gracefully and signals the end of execution. The application reverts to the "Stand By" mode and the familiar message "Nurse@Home is ready to help you! You can start making measurements" appears in the Message Box.

   c. "Send Now!" interceptor

   **Results**: By pressing the "Send Now!" button, while the data acquisition is running, the thread stops gracefully and control is passed to the data encapsulation thread. The front-end response is identical to the one viewed in Test 3a.

   d. Time-out interceptor

   **Results**: One of the sensors is purposefully blocked from sending input, to exhaust the time-out period without receiving complete feedback. The front-end indicator starts count-down as expected. When the countdown reaches "1 sec", the application logs indicate the interception of the data acquisition thread and activation of the encapsulation thread. The front-end response is identical to the one viewed in Test 3a.

4. Temporal indicators

   **Methodology**: The Timer Box in the front-end publishes two significant pieces of information, the date of last dispatch and the time interval left for taking measurements, while "Running". In order to test how these fields are populated, the whole "Running" process is executed, and the field values are observed. The former should also be persisted in the properties file, a task which is performed successfully in Test 1b.

a. Timer countdown

**Results**: The count-down starts 0-1 second after the application passes to "Running" mode and proceeds second per second, until the phase is interrupted.

b. Dispatch date

**Results**: Once the medical data are successfully sent to the server, the dispatch date in the GUI is updated to reflect the last occurrence. The respective value in the properties file is also updated.

5. <u>Sensor listening process</u>

   **Methodology**: Accepting sensor input is among the most important functions of the software. A dedicated driver has been developed for each sensor, which employs wireless connectivity to communicate and read data from the device. Most drivers incorporate the functionality of creating and maintaining a connection, validating input and sending configuration data to the sensor. The issues examined in this test case are the stability of connection and the efficiency of data retrieval. Accuracy of medical data is considered given, therefore it is not tested.

   **Results**: The device drivers accept data at a specific port, so the same driver may not be instantiated more than once. That is fully acceptable, since the application creates a unique instance of each driver. At start-up, the driver connects to the device and keeps the connection open to receive data whenever it is available. Even if the application specifically requests to create a connection, the driver will only re-connect if the connection was lost or otherwise inactivated, thus ensuring that only one thread will be listening to a specific port. The driver never delivers corrupt data; it is highly efficient in requesting, validating and forwarding a readable, logical data set.

6. <u>XML generation</u>

   **Methodology**: The second phase of the "Running" mode is transformation of all data readings into a well-formed XML document, which conforms to the schema presented in Appendix B. Testing this phase encompasses checks for inclusion of all measurements, correct field population, XML well-forming and XSD conformance.

   **Results**: After retrieval from the sensors, the medical data is stored in memory in a key-value format, which allows for easy interpretation to the XML nested element format. Therefore, the values are always correctly reflected in the resulting document. Popular XML processing libraries are used in the software, which produce well-formed XMLs by default. The only difficulty in this phase is XSD conformance, which caused some initial delays. However, after repeated trials, the software utilizes a custom output module, which renders valid and conformant XMLs.

7. XML dispatch

   **Methodology**: The third phase of the "Running" mode is the dispatch of the XML document. After the document is produced, the application starts the dispatch thread, which first tries to establish an Internet connection. When the connection is established, an HTTP request is sent to the Synapsis server that contains the generated XML as body.

   **Results**: The first concern is to test the application's ability to respond, when no connection may be established or when the server does not respond. The thread makes three attempts to connect, after which an exception is thrown – and logged – and the application returns to the "Stand By" mode, and a message appears in the Message Box, that the data dispatch failed. The XML document is saved in the file system, to be retrieved manually, if this intention arises. Most importantly, the application does not hang or enter an unstable, unpredictable state.

   If the connection is established, the HTTP request completes as expected and an acknowledgment is returned from the Synapsis server. The front-end displays a message of successful completion and the application returns to the "Stand By" mode.

### 5.2.2 On-the-move environment tests

HEARTFAID's on-the-move environment client application was implemented as a MIDlet application executed in a J2ME environment. The functionality of the software was tested regarding the communication with the medical devices/sensors in the Bluetooth neighborhood (client and server modes), data transmission to the intermediate gateway, xml message generation at the gateway side and sockets connectivity between the gateway and the HEARTFAID platform.

1. Bluetooth Modes
   a. Bluetooth Server mode
   **Methodology**: The purpose of this test was to examine the proper functionality of the J2ME application regarding the ability to act as a server that will accept connections over the bluetooth channel in order to receive the data from the client medical devices in the neighborhood. In order to achieve this, initially the service should be detectable from another device (PC or mobile phone in the neighborhood) and then able to receive data from remote devcices.

**Results**: After starting the bluetooth server functionality (by selecting the "Start Server" command in the serverForm of the application the phrase "Server Started" appears in the form. A device and service detection software was run on another device. This software successfully detected the bluetooth service.

b. Bluetooth Client mode
**Methodology**: The purpose of this test was to examine the proper functionality of the J2ME application regarding the ability to act as a client that will connect to remote server devices in order to acquire the relevant data. The basic functionality involved in this test is the devcie discovery within the J2ME application and the capability to connect to each one of the server devices.

**Results**: The J2ME application is capable to detect the devices in the neighborhood by selecting the Inquire command in the RemoteDeviceExplorer list. The detected devices appear in the list. The application can successfully to all the devices.

2. Data acquisition from medical devices
**Methodology**: The purpose of this test is to examine the proper data transmission from the medical devices to the mobile device in both the client and server operating modes. In this way the correct implementation of the communication protocols was verified. For each one of the devcices, the proper data transmission to the mobile device was tested.

**Results**: The mobile device was able to receive data from all the devices. The received data were shown on the server and client forms depending on the remote device.

3. Data transmission to the intermediate gateway
**Methodology**: The purpose of this test was to verify that the data were successfully transmitted to the gateway. For this reason, the J2ME client processes for the result of the HTTPRequest.

**Results**: Upon the successful transmission of the data to the intermediate gateway, a positive HTTPResponse is sent to the J2ME client and is displayed. In case of error in transmission, the error response is sent to the client.

4. XML creation in the gateway
**Methodology**: The purpose of this test was to verify that the XML data were successfully created in the gateway. For this reason, ASPX component locally

stores the XML generated messages into files, which are accessed by an FTP client and are validated towards the defined XML messaging schema.

**Results**: The ASPX successfully created the XML messages and saved them locally into files. The files were then offline validated towards the XML schema and there were found compiant with it.

5. <u>Connectivity between the intermediate gateway and the HEARTFAID platform</u>

**Methodology**: The purpose of this test was to verify that a successful connection and data transmission can take place between the intermediate gateway and the HEARTFAID platform. For this reason, the intermediate gateway processed the response of the platform after the transmission of the data to it.

**Results**: The intermediate gatewayh can successfully transmit XML data over a socket and http connection to the HEARTFAID platform.

# 6 Limitations of the prototype compared to the final infrastructure

The current prototype version of the Data Acquisition and Transmission was designed and implemented according to the specifications of the final, fully-functional product and encompasses the vast majority of its features, even though it is an intermediate release. Software engineering principles where applied in every aspect of development, which resulted to thorough test cases, modular design, adherence to patterns and meticulous documentation.

The following features are identified as limitations of the current prototype compared to the final infrastructure and their implementation is planned for the next release:

i. Integration of additional sensors in the home environment: The intermediate release supports two devices, the Nonin pulse/oximeter and the A&D electronic scale. Drivers have already been developed for two further devices, the A&D blood pressure and the Esaote cardiograph, the integration of which is already in progress. Furthermore, the integration of the MagIC vest is considered as very important in order to compile a detailed medical picture for the patient at home.

ii. Secure transmission. It was agreed among the partners to use HTTP during data transmission for the current software version, where the main point of interest was functionality and compatibility. However, the absolute necessity for secure communication is acknowledged and the use of HTTPS will be adopted in the final version.

iii. Minnesota questionnaire. In an effort to create a complete medical picture, some subjective remarks from the patient or care-taker could prove very useful. A questionnaire that is manually filled-in by the patient, along with the measurements from the medical devices, will provide valuable insight to the patient's state. The questionnaire will be integrated in the Home Client as a separate tab or set of tabs, since the software will already be running in the patient's environment and s/he should not be obliged to run a separate application for it.

Finally, since the structure of CRFs applied in medical research changes often in the course of a project, the configuration of the HEARTFAID eCRF presented in the Figure 14 is not final yet and will, according to expectations, undergo further modification. The current version of the eCRF prototype may export the data as XML messages to other components of the platform. This interface, however, is still under development and needs intensive collaboration among project partners in order to make the integration smooth. A Single Sign-On (SSO) authentication with the platform is currently under development. Similar as it is in the case of other subsystems, the connection with the server has to be secured by SSL.

# 7 Conclusions

In the framework of the work-package WP2 “BIOMEDICAL DATA IDENTIFICATION AND COLLECTION” of the HEARTFAID project, this document has dealt with the detailed description of the prototype related to the Data Acquisition and Transmission. This prototype was built based on the requirements and specifications defined in D14, enhanced also with the alert notification system and the eCRF application, which are implemented in WP6.

Initially an overview of the prototype was given describing its structure and elements, while it continues with the detailed description of them, starting with the connectable medical devices that produce the biomedical data and continuouing with the data acquisition applications for each clinical environment. For every data acuisition application, a detailed description was given, while also the installation instractions and the user’s guide were described in detail.

Having covered the technological infrastructure (devices and applications) related to data acuisition the data mechanisms were presented in detail. The logic behing the design of the XML messaging schema was given, aiming mainly to provide a flexible and extensible form of messaging. Furthermore, the functionality of the intermediate gateway for XMl generation in the on-the-move scenario, as well as the functionality of the alert notification system was presented. The functionality of the HEARTFAID Electronic Health Record was also presented, which allows for the comprehensive view on all the health information available for the patient.

Having described in detail the entire technological infrastructure allowing the creation of the complete chain from the medical deviced to the platform, a set of tests related to validation of the proper functionality of the elements was presented. These tests are standalone tests verifying the correct operation of the elements themeselves, as well as integration driven tests verifying the collaboration among verious components. It has to be noted that a preliminary demonstration of the portotype’s functionality was given in front of all HEARTFAID partners during the STAB meeting in Zagreb, Croatia in June 2007.

Finally, some limitations of the prototype compared to its envisaged functionality were presented. These limitations are mainly related to the completion of devices integration, incorporation of transmission security mechanisms, integration of subjective remarks (e.g Minnessota questionnaire), as well as the finalisation of integration of the eCRF to the platform via XML messaging.

## 8 References

[Duftschmid2002] Duftschmid G., Gall W., Eigenbauer E., Dorda W., *Management of data from clinical trials using the ArchiMed system*, Med Inform Internet, 27(2), pp. 85-98, 2002

[Johnson2005] Johnson R., Hoeller J., Arendsen A., Risberg T, Sampaleanu C., *Professional Java Development with the Spring Framework*, John Wiley & Sons, 2005

[Nadkarni1997] Nadkarni P., *QAV: querying entity – attribute – value metadata in a biomedical database*, Comput Meth Prog Biom., 53, pp. 93-103, 1997

[Nadkarni1999] Nadkarni P., et al., *Organization of Heterogeneous Scientific Data Using the EAV/CR Representation*, J Am Med Inform Assoc., 6(6), pp. 478-493, 1999

[Schleyer2000] Schleyer T., Forrest J., *Methods for the Design and Administration of Web-based Surveys*, J Am Med Inform Assoc., 7(4), pp. 416-425, 2000

[Whatt2000] Wyatt JC., *When to Use Web-based Surveys*, J Am Med Inform Assoc., 7(4), pp. 426-430, 2000

# Appendix A: The complete XML schema definition for the SMS gateway communication

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
<!-- definition of basic simple elements -->
<xs:simpleType name="requestType">
<xs:restriction base="xs:string">
<xs:enumeration value="SENDMESSAGES"/>
<xs:enumeration value="GETMESSAGESTATUS"/>
<xs:enumeration value="GETCREDITS"/>
<xs:enumeration value="SCHEDULESEND"/>
<xs:enumeration value="SENDQUOTA"/>
</xs:restriction>
</xs:simpleType>
<xs:simpleType name="subject">
<xs:restriction base="xs:string">
<xs:pattern value="([a-zA-Z0-9])+"/>
<xs:maxLength value="11"/>
</xs:restriction>
</xs:simpleType>
<xs:simpleType name="phoneNum">
<xs:restriction base="xs:string">
<xs:pattern value="([0-9])+"/>
<xs:maxLength value="16"/>
<xs:minLength value="8"/>
</xs:restriction>
</xs:simpleType>
<xs:simpleType name="smsText">
<xs:restriction base="xs:string">
<xs:maxLength value="160"/>
</xs:restriction>
</xs:simpleType>
<xs:simpleType name="msgID">
<xs:restriction base="xs:string">
<xs:pattern value="([0-9])+"/>
</xs:restriction>
</xs:simpleType>
<xs:simpleType name="percent">
<xs:restriction base="xs:integer">
<xs:minInclusive value="1"/>
<xs:maxInclusive value="100"/>
</xs:restriction>
</xs:simpleType>
<!-- definition of basic complex elements -->
<xs:complexType name="authenticationData">
<xs:sequence>
<xs:element name="USERNAME" type="xs:string"/>
<xs:element name="PASSWORD" type="xs:string"/>
</xs:sequence>
</xs:complexType>
```

```
<xs:complexType name="recipientsList">
<xs:sequence>
<xs:element name="PHONE" type="phoneNum"
maxOccurs="unbounded"/>
</xs:sequence>
</xs:complexType>
<xs:complexType name="messageBody">
<xs:sequence>
<xs:element name="TEXT" type="smsText"/>
</xs:sequence>
</xs:complexType>
<xs:complexType name="scheduleTime">
<xs:sequence>
<xs:element name="DATE" type="xs:date"/>
<xs:element name="TIME" type="xs:time"/>
</xs:sequence>
</xs:complexType>
<!-- definition of request elements and parts-->
<xs:element name="REQUEST" type="requestType"/>
<xs:element name="SUBJECT" type="subject"/>
<xs:element name="CUSTOMER" type="authenticationData"/>
<xs:element name="RECIPIENTS" type="recipientsList"/>
<xs:element name="MESSAGE" type="messageBody"/>
<xs:element name="SCHEDULE" type="scheduleTime"/>
<xs:element name="MESSAGEID" type="msgID"/>
<xs:element name="QUOTA" type="percent"/>
<xs:group name="requestHeader">
<xs:sequence>
<xs:element ref="REQUEST"/>
<xs:element ref="CUSTOMER"/>
</xs:sequence>
</xs:group>
<xs:group name="sendRequestBody">
<xs:sequence>
<xs:element ref="SUBJECT"/>
<xs:element ref="RECIPIENTS"/>
<xs:element ref="MESSAGE"/>
</xs:sequence>
</xs:group>
<xs:group name="scheduleSendRequestBody">
<xs:sequence>
<xs:element ref="SCHEDULE"/>
<xs:group ref="sendRequestBody"/>
</xs:sequence>
</xs:group>
<xs:group name="quotaSendRequestBody">
<xs:sequence>
<xs:element ref="QUOTA"/>
<xs:group ref="scheduleSendRequestBody"/>
</xs:sequence>
</xs:group>
<xs:group name="requestBody">
<xs:choice>
```

```
<xs:group ref="sendRequestBody"/>
<xs:group ref="scheduleSendRequestBody"/>
<xs:group ref="quotaSendRequestBody"/>
<xs:element ref="MESSAGEID"/>
</xs:choice>
</xs:group>
<!-- definition of SMS Gateway request-->
<xs:element name="XML">
<xs:complexType>
<xs:sequence>
<xs:group ref="requestHeader"/>
<xs:group ref="requestBody" minOccurs="0"
maxOccurs="1"/>
</xs:sequence>
</xs:complexType>
</xs:element>
</xs:schema>
```

# 9 Appendix B: The complete XML schema definition for the medical data transmission

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema
        xmlns="http://www.synapsis.com/heartfaid/ph/dom"
        targetNamespace="http://www.synapsis.com/heartfaid/ph/dom"
        xmlns:xsd="http://www.w3.org/2001/XMLSchema"
        elementFormDefault="qualified">

<xsd:element name="gwph-message">
        <xsd:complexType>
                <xsd:sequence>
                        <!-- Destination address. Always 0 if PH is the message receiver. -->
                        <xsd:element name="destination-address" type="xsd:byte"/>
                        <!-- Progressive number of the message. Useful to check message leak -->
                        <xsd:element name="sequence-counter" type="xsd:int"/>
                        <xsd:choice>
                                <!-- either one composite observation or one single observation -->
                                <xsd:element name="compositeobs" type="CompositeObservationType"/>
                                <xsd:element name="obs" type="ObservationType"/>
                        </xsd:choice>
                </xsd:sequence>
        </xsd:complexType>
</xsd:element>

<xsd:complexType name="_AbstractObservationType" abstract="true">
                <xsd:sequence>
```

```
                        <!-- all observations can contain an arbitrary number of parameters, one optional recursive addressing structure,
one optional timestamp -->
                        <xsd:element name="param" type="ParamType" minOccurs="0" maxOccurs="unbounded"/>
                        <xsd:element name="address" type="AddressType" minOccurs="0"/>
                        <xsd:element name="timestamp" type="xsd:dateTime" minOccurs="0"/>
                </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CompositeObservationType">
        <xsd:complexContent>
                <xsd:extension base="_AbstractObservationType">
                <!-- a composite observation can contain an arbitrary number of observations (composite or not) -->
                <xsd:choice minOccurs="0" maxOccurs="unbounded">
                                <!-- either one composite observation or one single observation -->
                                <xsd:element name="compositeobs" type="CompositeObservationType"/>
                                <xsd:element name="obs" type="ObservationType"/>
                        </xsd:choice>
                </xsd:extension>
        </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="ObservationType">
        <xsd:complexContent>
             <xsd:extension base="_AbstractObservationType">
                        <xsd:sequence>
                                <xsd:element name="measurements" type="MeasurementsType" minOccurs="0"/>
                        </xsd:sequence>
                        <xsd:attribute name="type" type="xsd:string" use="optional"/>
                <xsd:attribute name="code" type="xsd:string" use="optional"/>
            </xsd:extension>
        </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="ParamType">
        <!-- a param is a element with character content and two attributes name and type. Type serves as a hint for parsing an can
contain one of the following string constant string (default), number, boolean or date -->
        <xsd:simpleContent>
                <xsd:extension base="xsd:string">
                        <xsd:attribute name="type" type="paramType" default="string"/>
                        <xsd:attribute name="name" type="xsd:Name" use="required"/>
                </xsd:extension>
        </xsd:simpleContent>
</xsd:complexType>

<xsd:simpleType name="paramType">
        <xsd:restriction base="xsd:string">
                <xsd:enumeration value="string"/>
                <xsd:enumeration value="number"/>
                <xsd:enumeration value="boolean"/>
                <xsd:enumeration value="date"/>
        </xsd:restriction>
</xsd:simpleType>

<xsd:complexType name="AddressType">
        <!-- a recursive addressing scheme -->
        <xsd:all>
```

```
          <xsd:element name="address" type="AddressType" minOccurs="0" />
     </xsd:all>
     <xsd:attribute name="descr" type="xsd:string" />
     <xsd:attribute name="id" type="xsd:string" />
</xsd:complexType>

<xsd:complexType name="MeasurementsType">
     <!-- Heartfaid typical measurement structure -->
     <xsd:sequence>
          <xsd:element name="measurement" maxOccurs="unbounded">
               <xsd:complexType>
                    <xsd:sequence>
                         <xsd:element name="alarm" type="xsd:string" minOccurs="0"/>
                         <xsd:element name="type" type="xsd:string" />
                         <xsd:element name="sampleRate" type="xsd:float" minOccurs="0"/>
                         <xsd:element name="value">
                              <xsd:simpleType>
                                   <xsd:list itemType="xsd:float"/>
                              </xsd:simpleType>
                         </xsd:element>
                         <xsd:element name="units" type="xsd:string" />
                    </xsd:sequence>
               </xsd:complexType>
          </xsd:element>
     </xsd:sequence>
</xsd:complexType>

</xsd:schema>
```

# 10 Appendix C: Example valid XML messages containing biomedical data measurements

## 10.1 Simple Observation

```
<?xml version="1.0" encoding="UTF-8"?>
<gwph-message xsi:noNamespaceSchemaLocation="GWPH-protocol-v1.9.xsd"
xmlns="http://www.synapsis.com/heartfaid/ph/dom" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <destination-address>0</destination-address>
  <sequence-counter>0</sequence-counter>
  <obs type="" code="0">
    <address descr="xml-gw" id="2">
      <address descr="phone" id="1">
        <address descr="bt" id="1" />
      </address>
    </address>
    <timestamp>2001-12-31T12:00:00</timestamp>
    <measurements>
      <measurement>
        <alarm>High</alarm>
        <type>SBP</type>
        <value>110</value>
        <units>mmHg</units>
      </measurement>
      <measurement>
        <alarm>Low</alarm>
        <type>DBP</type>
        <value>92</value>
        <units>mmHg</units>
      </measurement>
    </measurements>
  </obs>
</gwph-message>
```

## 10.2 Simple Observation with some parameters like patient-id, name etc.

```
<?xml version="1.0" encoding="UTF-8"?>
<gwph-message xsi:noNamespaceSchemaLocation="GWPH-protocol-v1.9.xsd"
xmlns="http://www.synapsis.com/heartfaid/ph/dom" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <destination-address>0</destination-address>
 <sequence-counter>0</sequence-counter>
 <obs type="" code="0">
   <param name="patient-id" type="string">PAT-1.0</param>
   <param name="patient-name" type="string">John</param>
   <param name="patient-surname" type="string">Doe</param>
   <param name="patient-age" type="number">50</param>

   <address descr="xml-gw" id="192.168.0.33">
     <address descr="phone" id="1">
       <address descr="bt" id="1" />
     </address>
   </address>
   <timestamp>2006-11-22T13:55:55</timestamp>
   <measurements>
     <measurement>
       <alarm>Normal</alarm>
       <type>SBP</type>
       <value>108</value>
       <units>mmHg</units>
     </measurement>
     <measurement>
       <alarm>Normal</alarm>
       <type>DBP</type>
       <value>88</value>
       <units>mmHg</units>
     </measurement>
   </measurements>
 </obs>
</gwph-message>
```

## 10.3 Composite Observation with metadata

```
<?xml version="1.0" encoding="UTF-8"?>
<gwph-message xsi:noNamespaceSchemaLocation="GWPH-protocol-v1.9.xsd"
xmlns="http://www.synapsis.com/heartfaid/ph/dom" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <destination-address>0</destination-address>
 <sequence-counter>0</sequence-counter>
 <compositeobs>
    <param name="patient-id" type="string">PAT-1.0</param>
    <param name="patient-name" type="string">John</param>
    <param name="patient-surname" type="string">Doe</param>
    <param name="patient-age" type="number">50</param>
    <param name="instrument" type="string">TM2430</param>

    <address descr="xml-gw" id="192.168.0.33">
       <address descr="tm2430" id="1"/>
    </address>
    <obs>
     <address descr="xml-gw" id="192.168.0.33">
       <address descr="tm2430" id="1">
         <address descr="sens" id="1" />
       </address>
     </address>
     <timestamp>2006-11-22T13:55:55</timestamp>
     <measurements>
       <measurement>
         <alarm>Normal</alarm>
         <type>SBP</type>
         <value>108</value>
         <units>mmHg</units>
       </measurement>
       <measurement>
         <alarm>Normal</alarm>
         <type>DBP</type>
```

```
        <value>88</value>
        <units>mmHg</units>
      </measurement>
    </measurements>
   </obs>
   <obs>
    <address descr="xml-gw" id="192.168.0.33">
      <address descr="tm2430" id="1">
        <address descr="sens" id="1" />
      </address>
    </address>
    <timestamp>2006-11-22T13:55:55</timestamp>
    <measurements>
      <measurement>
        <alarm>Normal</alarm>
        <type>temp</type>
        <value>36.6</value>
        <units>C</units>
      </measurement>
      <measurement>
        <alarm>Normal</alarm>
        <type>hum</type>
        <value>68</value>
        <units>percent</units>
      </measurement>
    </measurements>
   </obs>
 </compositeobs>
</gwph-message>
```

## 10.4 Simple Observation containing dummy ECG data

```
<?xml version="1.0" encoding="UTF-8"?>
<gwph-message xmlns="http://www.synapsis.com/heartfaid/ph/dom" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xsi:schemaLocation="http://www.synapsis.com/heartfaid/ph/dom src/com/synapsis/heartfaid/ph/GWPH-
protocol-v1.10.xsd ">
      <destination-address>0</destination-address>
      <sequence-counter>0</sequence-counter>
      <obs type="" code="0">
            <address descr="xml-gw" id="2">
                  <address descr="ecg-simulator" id="1">
                        <address descr="ISTI-Myocardial-Electronic-Simulator" id="1"/>
                  </address>
            </address>
            <timestamp>2006-10-27T06:17:55</timestamp>
            <measurements>
                  <measurement>
                        <type>av1</type>
                        <sampleRate>5000000</sampleRate>
                        <value>
                              0.020 0.020 0.040 0.020 0.020 0.020 0.040 0.020 0.040 0.020 0.020 0.020
                              0.020 0.000 0.020 0.040 0.040 0.020 0.020 0.020 0.060 0.020 0.020 0.020
                              0.000 0.000 0.000 0.020 0.020 0.000 0.000 0.020 0.000 0.020 0.020 0.000
                              ................................................................................
                              ................................................................................
                              ................................................................................
                              0.020 0.040 0.040 0.040 0.040 0.020 0.020 0.040 0.020 0.000 0.020 0.040
                              0.040 0.020 0.020 0.040 0.040 0.020 0.020 0.020 0.020 0.040 0.020 0.020
                              0.040 0.020
                        </value>
                        <units>V</units>
                  </measurement>
            </measurements>
      </obs>
</gwph-message>
```