Chapter 10
System Middleware

André Bideaux, Stefan Hey, Panagiota Anastasolpoulou, Alberto Fernandez, Christos P. Antonopoulos, and Nikolaos S. Voros

Abstract This document defines the middleware of a cyberphysical monitoring system for epilepsy and related brain disorders. Taking into account requirements, this document provides insights about the service functionalities of the middleware and the interfaces connecting the different technologies in place within the system middleware.

The document is divided into two main parts, the first part describes the functional, non-functional and security aspects of the middleware, while the second part is all necessary information about the middleware architecture.

The middleware runs within the Home Gateway, it is the ICT part responsible to connect sensor data to upper software layers (like personal health record service and the tele-alarm and messaging manager). Furthermore it provides the necessary infrastructure (thanks to the inclusion of a data stream management system) for the development of on-line multi-parametric data processing. Upper software layers connected are the Tele-alarm and Messaging Manager and the Personal Health Record. All middleware components follow a SOA (service oriented architecture) paradigm that allows easy scalability of the system.

A. Bideaux (✉) • S. Hey • P. Anastasolpoulou
KIT Karlsruhe Institute of Technology, Institute for Information Technology, Karlsruhe, Germany
e-mail: andre.bideaux@kit.edu

A. Fernandez
Sensing and Control Systems S.L., Barcelona, Spain

C.P. Antonopoulos • N.S. Voros
Embedded System Design and Application Laboratory, Computer and Informatics Engineering Department, Technological Educational Institute of Western Greece, Antirio, Greece

N.S. Voros, C.P. Antonopoulos (eds.), *Cyberphysical Systems for Epilepsy and Related Brain Disorders*, DOI 10.1007/978-3-319-20049-1_10

10.1 Middleware Requirements

10.1.1 Introduction

This chapter defines the requirements related to the middleware platform. From sensors devices to electronic health record interface and notification applications, the middleware address the needs of the interoperability that is required among the different technologies in place within cyberphysical monitoring for epilepsy and related brain disorders and its functional goals.

The main source of information used to develop middleware requirements are user and sensor requirements, with main focus on the description of the services to be delivered to the users; patients and healthcare professionals, and the specification of the scenarios and use cases.

Specific privacy and security requirements have been devoted to security issues, extracting main objectives to be accomplished at sensor/WSN and extrapolating the implications to be developed at middleware level.

10.1.2 Functional Requirements

The main purpose of Middleware is to create an isolation layer between the physical world (sensors/field devices) and high level applications.

Due to the heterogeneous nature of the physical world, it is expected to have many different field devices (which likely will include sensors and a communication system) running different protocols and communication interfaces (serial, ip (udp or tcp), etc.).

At high level applications side, the creation of a standardised way of accessing sensor data for online/offline processing is important for the application developers. Application developers are managing the information from the service creation point of view, hence isolating data access from underlying communication and commissioning of field sensors is mandatory.

The communication between physical and application worlds is accomplished at intermediate level between the two defined interfaces (online and offline). In this middle layer several functions supported by a database will run to match the services necessities. The description of the functional requirements is divided into the following four parts: Functional Sensor Requirements, Functional Requirements of Data Stream Management System (DSMS), Functional Requirements of Application programming interfaces (API) and Functional Requirements of Notification services.

10.1.2.1 Functional Sensor Requirements

Within this section, all the functional requirements for/from sensors involved in the whole System and related to the middleware are described. The following requirements are related to acquiring, processing and storing data from sensors described in Chap. 8:

- All the data must be time stamped appropriately, at a sufficient level to perform synchronisation among separate data modalities, e.g. EEG & ECG
- Data from Sensors must be streamed to the middleware
- Middleware drivers must collect the chunks of data from Sensor and pass them to the online data stream management system (DSMS)
- Middleware can pull previously recorded data from the sensors.
- Middleware supports data input from sensors specified in Chap. 8
- Middleware is able to store the data locally on the Home Gateway
- Alarms/Warnings can be originated by sensors (i.e. push button)

10.1.2.2 Functional Requirements of Data Stream Management System (Online Analysis)

The Data Stream Management System (DSMS) is the backbone for online analysis. The following functional requirements describe the methods for online management, fusion and analysis of sensor data:

- Detection of abnormal values: The system must detect abnormal values due to improper use (e.g. motion artefacts) of sensors.
- The system need to use a variety of algorithms, to be able to deal with the streaming nature of data in order to perform the online analysis efficiently.
- The DSMS has to be configured to perform the desired analysis.
- Application errors need to be logged for efficient error handling.
- The online algorithms must be able to detect epileptic seizures and their patterns.
- Alarms/Warnings need to be originated by DSMS.
- Changing the configuration parameters must be possible.
- Detection of modalities including a preset range of normal values (i.e. excessive tachycardia and oxygen level excursions) for each patient individually.
- Use of activity sensor for diagnostic purposes involving cardiogenic triggers of seizures.

10.1.2.3 Functional Requirements of the Application Programming Interface (API)

How do we interact with the middleware? What are the different integrations between systems and middleware for user services? These functional requirements are described as follow:

- Middleware will initiate the uploading of sensor data to the PHR via PHR-API by notifying the upper layers.
- Middleware will send the raw data from sensors to the PHR-API by notifying the upper layers.
- Configuration parameters related to sensors are provided by upper layers (GUI/ PHR).
- Middleware will receive Patient ID from underlayed processes and services which are controlled by the GUI.
- Alarms/Warnings can be originated by the upper layers.

10.1.2.4 Functional Requirements of Notification services

The middleware must ensure the communication between components. All the requirements needed to communicate events, intercommunication processes and send data to other managed systems are described as follow:

- Alarm notification: Middleware must be able to pass alarms to the upper layers.
- Middleware will receive the configuration parameters from the upper layers (GUI).
- Middleware will be able to send the status of operations to the upper layers (Upload progress, performing analysis, data sent, data error, etc.).
- Application errors can be send as notifications to the upper layers (GUI).
- Middleware will receive a ‘send sensor data’ trigger from the upper layers (GUI).
- Middleware will notify upper layers (GUI) whether the function used is in the ON or OFF mode.
- Middleware will receive a ‘capture sensor data’ trigger from the upper layers (GUI).
- Middleware will receive a ‘pause’ and ‘resume’ trigger from the upper layers (GUI).

10.1.3 Non-functional Requirements

Non-functional requirements for the middleware are qualities or standards that the system has, but which are not tasks or features automated by the platform.

Non-functional requirements need to be made precise and actionable. “SMART” requirements [1] have the following characteristics:

- Specific: without ambiguity, using consistent terminology, simple and at the appropriate level of detail.
- Measurable: it is possible to verify that this requirement has been met.
- Attainable: technically feasible.
- Realizable: realistic, given the resources.
- Traceable: linked from its conception through its specification to its subsequent design, implementation and test.

Regarding this set of characteristics, non-functional specifications for the middleware are described following the next areas:

10.1.3.1 Physical Technology

The following technologies are used to ensure correct operation of the middleware and they form the nucleus of all installations within the Home Gateway:

- Windows Web Server 2008 R2 64 bits
- SQL Server 2008 R2
- StreamInsigth™

10.1.3.2 Security

The security aspects of middleware follow the requirements and specifications which are documented in Chaps. 9 and 11. The following points give an overview about the most important requirements:

- Storing the activity for auditing.
- Anonymization of patient data (only the Middleware knows the patient ID).
- Using cryptographic random number generators to generate session IDs.
- Store credentials in a secure manner.
- Use Strong password policies.
- Encrypt communication channels to secure the authentication tokens.
- Authenticated API commands should be supported.

10.1.3.3 Compatibility/Interoperability

Middleware establishes an outgoing applicative connection (TCP, UDP, SOAP, and REST) towards the platform.

10.1.3.4 Reliability/Adaptivity

New types of sensor networks and dynamical (re-) configuration of data sources are supported in the middleware through xAffect and unisens format. Additionally, the middleware provides the basic building blocks for data communication between different components like PHR, DSMS and SHACU. The particularities of the information sent between those components are unknown by the middleware.

10.1.3.5 Robustness

Periodical retrieval/upload of data is automated by the drivers and all the transfers are reliable. Any failure in the system triggers automatic events to the administrator in order to be monitored and recovered.

10.1.3.6 Connectivity

The middleware needs a permanent or part time available, internet connection, to upload the data and event notifications to the PHR.

10.1.3.7 Scalability

The middleware is designed as a SOA (Service Oriented Architecture) platform. The architecture is extremely loosely coupled and well distributed to provide full scalability and efficiency.

10.1.3.8 Deployment

The middleware is tailored to be deployed locally at the Home Gateway. However, the technology used allows it to be ready to be deployed on Fully Managed Web Clusters. Therein supporting Software as a Service (SaaS) model with small modifications, so being ready to scale on the cloud (making use of the massive scalability of the cloud environment), and designed for fault tolerance, including management and monitoring software components.

10.2 Middleware Architecture

The principal objective of this Chapter is presenting an overview of the middleware package. The document will provide insights on the functional and non-functional aspects of the middleware. The architecture is based on ICT components which are described in Fig. 10.1. It highlights the middleware architecture within the overall system.

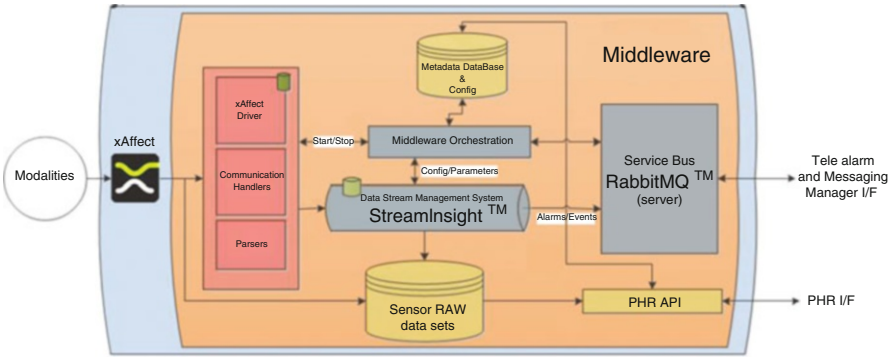

Fig. 10.1 Middleware architecture

As can be interpreted from Fig. 10.1, middleware will be installed in a local personal computer (the Home Gateway). Data from the different modalities (EEG, GSR, SPO2, etc…) will be acquired to the personal computer throughout an encrypted wireless channel. Data will be stored and uploaded to PHR server in a time configurable way (mainly in a daily basis scheme), from where data will be accessible and presented in a friendly manner to authorized persons (doctors, caregivers, etc…). When the on-line processing takes place, an event notification trigger can occur, by pressing the push button or by any event detection algorithm in the DSMS component. Depending on the severity of the event, immediate notification to a caregiver will be triggered using the MS (Management Service—SHACU) and immediate sensor data plus event information will be uploaded to PHR. Notice that to allow upload of data a broadband connection at home is required.

10.2.1 Secure Incoming Data

Security and privacy of patient data is of big concern in these cyberphysical systems which have been addressed in Chap. 9.

Regarding the communication, part of the efforts in the project is being concentrated to secure communications at RF level between the sensors and the home gateway by the provision of encryption/decryption engines.

To provide a secure communication between sensor (modalities) and home gateway, both hardware and software solutions are possible. While Chap. 9 explains the encryption phase, the following subsections describe two alternatives for the decryption of sensor data being sent through RF channels. The encrypted data reaching the middleware needs to be decrypted for online processing. There are two ways of decrypting the data, which are described within this chapter, hardware and software decryption.

Table 10.1 Decryption module processing delay and throughput rate (@200 MHz operating frequency)

Key size (bits)	Processing delay (in clock cycles)	Throughput rate (Mbps)
128	480	53.3
192	582	44.4
256	684	37.4

10.2.1.1 Hardware Decryption Engine

Hardware decryption module follows the same 8-bit architecture presented in Chap. 9. As done in encryption phase, decryption commences by executing the expansion operation in order to produce all intermediate requires keys. After the completion of the key expansion phase the decryption module is ready for the 128-bit data blocks processing. Due to the 8-bit data-path architecture that is used, 16 clock cycles are required to load/unload the 128-bit ciphertext/plaintext block. The input block is decrypted by performing four different byte-oriented transformations, which are executed sequentially and repeatedly (as rounds); the transformations are: Add Round Key, InvSubstitute Bytes, InvShift Rows and InvMix Columns. The resulted intermediate decipher result, known as state, is stored and updated at the end of each round. The number of rounds depends on the size of the key. The actual key size and the number of rounds are configured via the KEY_SIZE input as shown in Table 10.1. Note that, since the key size determines the number of rounds, it also affects the latency of the decryption module. Respective delay and throughput performance is presented in Table 10.1.

10.2.1.2 Software Decryption Engine

A decryption module can be developed within xAffect. Its deciphering process can be abstracted as much as possible providing wide algorithm customization.

This module should be designed to be flexible and adaptable which results a highly reusable ciphering/deciphering component that can be used within all the functionalities that xAffect provides, like data collection, data logging, data sending, etc.

10.2.2 Data Fusion

Data Fusion is a process dealing with the association, correlation, and combination of data and information from single and multiple sources. Multi sensor data fusion refers to the acquisition, processing and synergistic combination of information gathered by various knowledge sources and sensors to provide better understanding of a phenomenon.

The most important part of data fusion is, the data itself. Therefore, it is crucial to take care during the capturing in order to minimize uncertainties. However, there is one aspect that is sometimes left behind and in some cases it is the most relevant: adapt metadata in a common format in order to simplify the process.

The data fusion can be classified as low level fusion of data, high level variable fusion and mixture level fusion. When the data fusion is performed before analysis, it is classified as low level. When the data fusion is performed after some data analysis, it is classified as high level variable fusion. There are some situations where we can fuse data and variables.

The overall system architecture, depicted in Fig. 10.1, shows the two paths where processing of sensor data takes place. On-line multi-parametric data processing and analysis takes place at the Home Gateway, while the off-line multi-parametric data processing takes place in different server/database, by (i) accessing/acquiring sensor data from the PHR (previously captured by the system) or by (ii) accessing/acquiring sensor data from external databases not belonging to the system itself. The data flow is depicted in Fig. 10.2.

An interesting relation in Fig. 10.2 is between off and on line processing. The goal of Off-line processing is to detect relations between different modalities throughout extensive data analysis. Online processing on the other hand is used to detect events of special interest and reduce the data amount by processing the raw data.

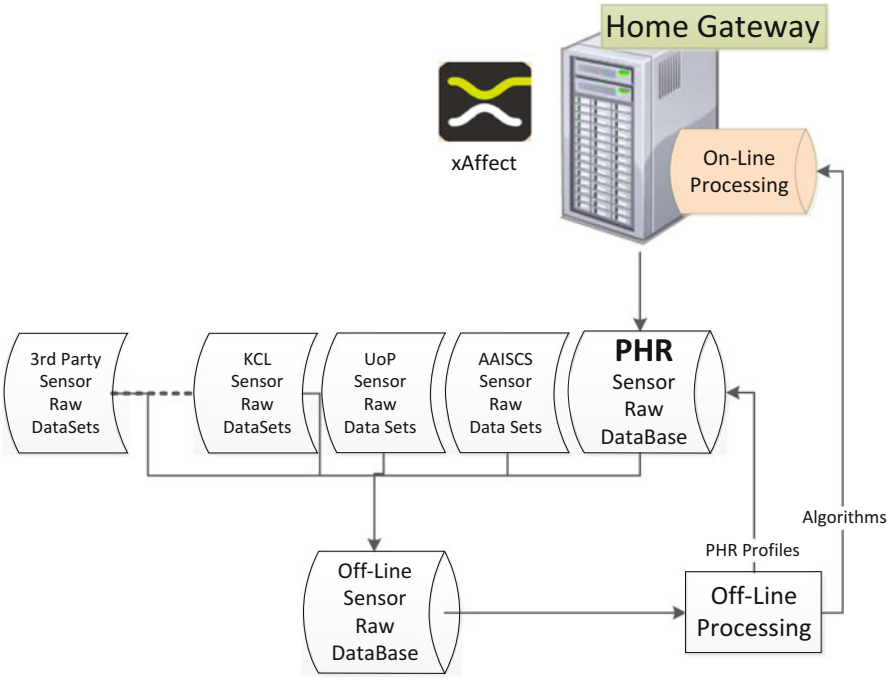

Fig. 10.2 Data fusion within the system

10.2.2.1 xAffect Framework

To connect all the data online, synchronize and handle it a lean and open framework, which can be personalized and reconfigured for each patient, is needed. xAffect is a software framework developed by the Research Center for Information Technology, Karlsruhe, Germany. It was developed in Java to fulfill real-time data processing, easy integration of different data sources, easy integration of algorithms and data logging of raw as well as derived data [2]. Libraries for some common sensors already exist in xAffect. To use a broad spectrum of bio-signals, additional libraries need to be implemented. The data format which is being used is the unisens-format. This is a universal and generic format suitable for recording and archiving sensor data from various recording systems and with various sampling frequencies [3].

The current version of xAffect can be modified in order to customize the interface for the middleware, which is necessary to achieve the performance and the required functionality for the system.

The changes that need to be made are:

- Additional libraries for sensors (To use a broad spectrum of sensors, non-existing libraries had to be written).
- Decryption module (data from sensors need to be ciphered).
- Data acquisition pause/resume to achieve the needs for the control of the sensor data acquisition.
- A custom notification module for communicating xAffect state to middleware DSMS.
- Extended data recording functionalities to provide configurable file splitting (in order to reduce high network consumption during heavy data uploads), alarm signal detection and a communication system with middleware and PHR.
- Extended data streaming functionalities to provide hot-plug client connections and custom xml output data formats (including gzip for network traffic optimization).

10.2.3 Graphical User Interface (GUI)

The Graphical User Interface handles the communication between the components and the User.

The GUI, shown in Fig. 10.3 is an example of how this part could look like. In the GUI, the user enters username and password and the patient-id. With the patient-id, the middleware will download the personalized profile from the PHR. Afterwards the user can press the configure button to initialize the communication with the sensors. Furthermore the profile contains information about the alarm settings. This enables the middleware to set up the DSMS with the customized alarm set [4].

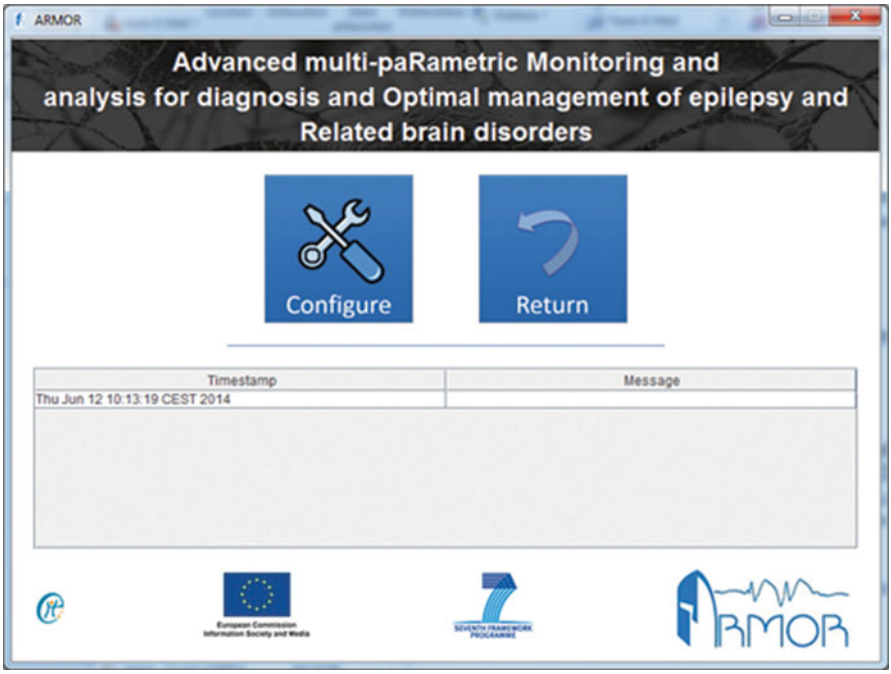

Fig. 10.3 Example of a graphical user interface for a system middleware

When the configuration process finished successfully the user is able to start the measurement by pressing the record button. During recording the data is streamed from the sensors through xAffect towards the DSMS and the storage. The middleware takes the data from the storage and uploads it in junks to the PHR. In case an Event occurs the data will be uploaded immediately. This ensures that when an alarm will be send to a clinician, the data will be ready for downloading and viewing. Furthermore, the GUI allows users to pause or resume the measurement. This allows the subject to interrupt the data acquisition and move out of the systems Bluetooth range.

10.2.4 Data Stream Management System (DSMS)

DSMS function takes place in on-line scenario, where real time processing of modalities is performed. In order to achieve it, minor adaptation of the xAffect™ stream connector of the sensor data being delivered to Middleware has been introduced in order to assure no sensor data is lost.

DSMS is based on Microsoft™ StreamInsight™ platform created for the development and deployment of complex event processing (CEP) applications.

It's a high-throughput stream processing architecture that uses .Framework-based development platform.

The development of DSMS allows sensor data to be received from xAffect™ in real time, synchronize it in a lossless way and feed it to computation algorithms. At the end, a framework is provided that enables the creation of own queries and policies over data (from now on Middleware Framework).

In order to explain data workflow within Middleware, we need to introduce some concepts related to StreamInsight™ as follow:

- **Sources**: They are data providers and can be implemented with adapters, IEnumerable or IObservable objects. They are in charge of collecting data, fit it into a payload, generate an appropriate timestamp (if needed) and redirect it to StreamInsight's core adding appropriate CTI (Current Time Increment) insertions. Sources are IObservable objects built with information coming from xAffect through TCP sockets.
- **Queries**: All data from sources goes directly to StreamInsight core (in any number of streams) where it's processed in order to satisfy one or more queries (LINQ). That will produce an output that will go to the data consumers (observers) where custom operations can be defined. It can also compute different operations depending on query such as aggregation, unions, max/min etc.
- **Consumers**: They are pieces of software which main function is processing data output from StreamInsight™ queries. They are usually implemented with the observer interface so it can be easy to notify them when new data is available.

Lossless stream between xAffect and DSMS can be achieved by using TCP channels. Then synchronized sensor data can be provided by using the built-in architecture of StreamInsight™ available through the highly flexible and configurable Insight framework which allows researchers to gather and analyze customized data structures derived from StreamInsight™ output.

10.2.4.1 Push Button Processing

One of the services provided by middleware is the processing of the marker button in the Bioplux™ device if it is configured as alarm push button.

When the user press the Bioplux™ push button, xAffect™ captures the event and records it. Additionally it sends the push button raw data through the IP connection to Middleware DSMS.

Middleware DSMS canalizes the push button raw data to StreamInsight™ core. The Middleware Framework contains a developed query to detect alarm signals, if it successfully detects an alarm condition. It notifies the event manager which builds and sends a XML message to the upper layers of the software (as can be seen in Fig. 10.1). Also if the user has defined more queries to detect alarms they would produce appropriate output that would be handled within a defined observer. Figure 10.4 depicts the complete data workflow.

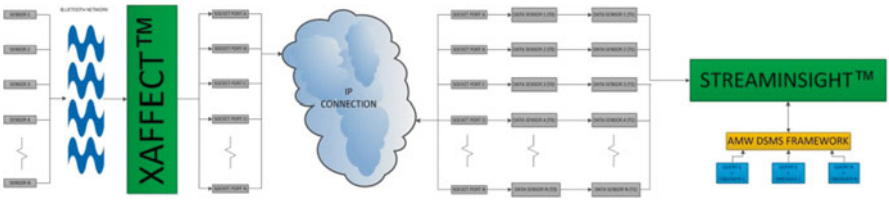

Fig. 10.4 Middlewaredata work flow

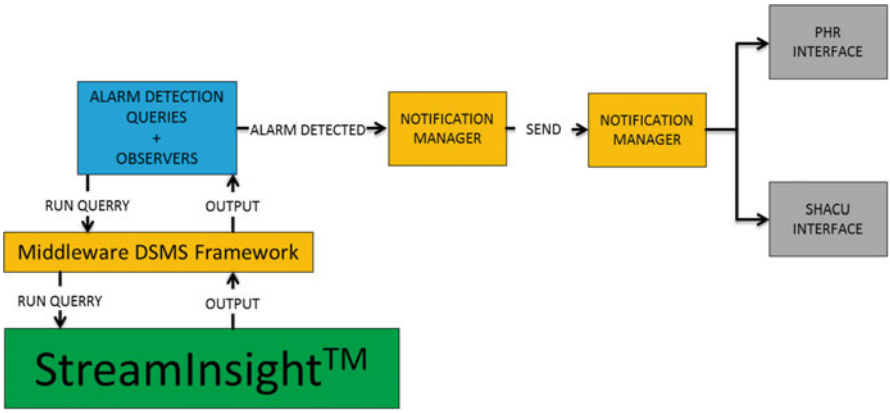

Fig. 10.5 Middleware DSMS notification manager module workflow

10.2.5 Event Notification

One of the objectives of Middleware DSMS is to be able to deliver alarm/warning events to high level applications. One of these events is the push button detection. Further events are for example alpha rhythm detection or seizure detection. The workflow of the event notification is shown in Fig. 10.5.

The module that communicates events within the middleware is called the Notification Manager which is explained in the next subsection.

10.2.5.1 Event Observer Output Format

In order to have a common way of communicating alarm/warning events from the observers to the notification manager, a guideline has been established. This guideline specifies the usage of a common Alarm data structure defined inside the Middleware Framework.

Table 10.2 Example XML notification for push-button event

```
<?xml version="1.0" encoding="utf-8"?>
<messagexmlns="http://armor.tesyd.teimes.gr"
xmlns:xsi="http://armor.tesyd.teimes.gr/2013/XMLSchema-instance"
xsi:schemaLocation="http://armor.tesyd.teimes.gr message.xsd">
      <header>
      </header>
      <body>
              <event>
                      <id>129</id>
                      <type>alarm</type>
                      <datetime>
                              <timestamp>1363773623</timestamp>
                              <utc_offset>+01</utc_offset>
                      </datetime>

              </event>
      </body>
</message>
```

This data structure defines three common fields which must be specified in order to generate a valid notification:

- ID: Identifies the event (for instance, 129 = push button)
- Type: Describe the kind of notification (ALARM/WARNING)
- Timestamp: Describes the event detection time

With this data the Notification Manager will be able to generate valid alert messages for each defined interface. An example can be found in Table 10.2 for the push button event.

10.2.5.2 Notification Manager

The Notification Manager module is mainly designed to detect anomalies or special situations in DSMS input-output raw data. It generates custom messages and forwards them to the specified endpoints. We can see in Fig. 10.5 how it is integrated within Middleware Data workflow.

After detecting an event that requires to be notified, the Notification Manager generates a suitable message (according to its configuration rules) and queues it. Then an automatic process takes the message and sends it through many interfaces as needed using any specific communication channel or protocol that each interface requires.

By default this module is configured to send XML messages related to alarms/events. The XML basic structure is defined to identify when the event happened, the event type and to which ID it is related. Table 10.2 shows an example of xml packet related to a notification of an alert event.

10.3 Conclusions

Sets of multiple sensors are required to acquiring the data needed to handle patient's epileptic disorders. That's why the middleware is the most important part of the whole system. The encrypted data streams will be decrypted, fused and streamed to the DSMS. The DSMS handles the live data analysis and will detect events of special interest. These events will then be reported via PHR to the clinicians, who will have direct access to the important data.

Tests under clinical conditions showed that the described system is able to handle online analysis with the sensor data. Events like a push button or the detection of alpha rhythms can be detected reliably and sent from the notification manager towards the PHR.

References

1. Mannion M, Keepence B (1995) SMART requirements. ACM SIGSOFT Software Engineering Notes 20(2): 42–47. http://www.win.tue.nl/~wstomv/edu/2ip30/references/smart-requirements.pdf
2. Schaaff K, Mueller L, Kirst M. http://www.xaffect.org
3. Kirst M, Ottenbacher J. www.unisens.org
4. Bideaux A, Anastasopoulou P, Hey S, Cañadas A, Fernandez A (2014) Mobile monitoring of epileptic patients using a reconfigurable cyberphysical system that handles multi-parametric data acquisition and analysis. In: Proceedings of the 2014 EAI international conference on wireless mobile communication and healthcare (MobiHealth), Athens, pp 377–380