Remote Accessibility to Diabetes Management and Therapy in Operational healthcare Networks.

REACTION (FP7 248590)

D10-2 Integration and Test Plan for BAN/PAN infrastructure

Date 2011-02-28

Version 1.4

Dissemination Level: Public

Table of Content

Document control page

Code	
Version	1.4
Date	24-01-2011
Dissemination level	PU
Category	R
Participant Partner(s)	DELTA, IMM, FORTH-ICS, CNET, FORTHNET
Author(s)	Rasmus G Haahr, Matts Ahlsen, Peeter Kool, Franco Chiarugi, Thomas Klotzbucher, Manolis Stratakis
Verified and approved by	
Work Package	WP10
Fragment	No
Distribution List	All
Abstract	This deliverable descibes a plan for integration and test of the individual components and their assemply to subsystems of the REACTION platform ralated to the sensor devices and the BAN/PAN networks.
Comments	
Status	☒ Draft ☐ Task leader accepted ☐ WP leader accepted ☐ Technical supervisor accepted ☐ Medical Engineering supervisor accepted ☐ Medical supervisor accepted ☐ Quality manager checked ☐ Project Coordinator accepted
Action requested	☒ to be revised by partners involved in the preparation of the deliverable ☐ for approval of the task leader ☐ for approval of the WP leader ☐ for approval of the Technical Manager ☐ for approval of the Medical Engineering Manager ☐ for approval of the Medical Manager ☐ for approval of the Quality Manager ☐ for approval of the Project Coordinator **Deadline for action**: N/A
Keywords	
References	
Previous Versions	

Version Notes

Version	Author(s)	Date	Changes made
0.1	Rasmus Haahr	2011-01-11	Template with initial outline
0.2	Rasmus Haahr	2011-01-04	DELTA contributions
0.3	Matts Ahlsen, Peeter Kool (CNet)	2011-01-06	CNET contributions
0.4	Giorgos Zacharioudakis	2011-02-07	FORTH contributions
0.5	Thomas Klotzbuecher	2011-02-11	IMM contributions

	1.0	Rasmus Haahr	2011-01-11	Document assembled, revised and prepared for review.
	1.1	Rasmus Haahr	2011-02-23	Consolidate comments from reviewers
	1.2	Matts Ahlsen	2011-02-23	Software sections updated according to reviews
	1.3	Rasmus Haahr	2011-02-24	Documented updated according to review. Added material on Solianis device
	1.4	Rasmus Haahr	2011-02-28	Adjusted formatting according to template version 3.0.
Internal review history	Reviewed by		Date	Comments made
	Lukas Schaupp		2011-02-17	Include Solianis device.
	Manolis Stratakis		2011-02-21	Corrections and extent summary.

1. Executive summary

This deliverable describes a plan for integration and technical test of functionality of the body area network (BAN) and personal area network (PAN). The BAN/PAN infrastructure consists of sensor devices and the REACTION application hosting client.

The sensor devices can measure physiological data such as glucose level, perform data analysis, and communicate the results and values to the REACTION application hosting client. The sensor devices communicate by ZigBee or Bluetooth in a data format defined by the Continua standards. The sensor devices can be either commercial available devices or devices which will be developed within REACTION. In the present deliverable, the integration and testing of sensor devices is illustrated with the ePatch and the Solianis device for continuous monitoring of glucose. The emphasis will be on the ePatch as this device is developed within REACTION. The ePatch can be used with a number of different sensing technologies. In this deliverable, a bio-potential sensor for electrocardiography and bio-optical sensors for glucose and oxygen saturation are described, both to be developed within REACTION.

The REACTION application hosting device (AHD) is in the present context a software package and not a physical device. The reason for this is that the software can be executed on a number of different devices and it has not been decided which device(s) will be used. The functionality of the REACTION application hosting device is networking and data fusion. In the context of BAN/PAN infrastructure only the networking with sensor devices will be considered. The REACTION application hosting device can connect to a number of sensor devices and integrate all data in a data fusion engine.

This deliverable is organized in two main parts. The first part describes the individual components, their technical integration both mechanical, electrical, software modules, and networking. The second part describes the test plan and test methods for functional tests of individual components, interfaces, subsystems, and the entire BAN/PAN infrastructure. This includes characterization of sensor performance, electrical test of sensor devices, software tests, and network interoperability.

2. Introduction

2.1 Overview of the REACTION project

The REACTION project aims to develop an integrated approach to improve long term management of diabetes through continuous glucose monitoring, monitoring of significant events, monitoring and predicting risks and/or related disease indicators, decision on therapy and treatments, education on life style factors such as obesity and exercise and, ultimately, automated closed-loop delivery of insulin.

Technically, the REACTION platform will be structured as an interoperable peer-to-peer communication platform based on service oriented architecture (SOA) where all functionalities, including the measurement acquisition performed by sensors and/or devices, are represented as services and applications consist of a series of services properly orchestrated in order to perform a desired workflow. The REACTION platform also will make extensive use of dynamic ontologies and advanced data management capabilities offering algorithms for clinical assessment and evaluation.
A range of REACTION services will be developed targeted to the management of insulin-dependent diabetic patients in different clinical environments. The services aim to improve continuous glucose monitoring (CGM) and insulin therapy by contextualized glycemic control based on patient activity, nutrition, interfering drugs, stress level, etc. for a proper evaluation and adjustments of basal and bolus doses. Decision support will assist healthcare professionals, patients and informal cares to make correct choices about glucose control, nutrition, exercise and insulin dosage, and thus to reach a better management of diabetes therapy.

REACTION will further develop complementary services targeted at the long term management of all diabetic patients, both Type I and Type II. Integrated monitoring, education, and risk evaluation will ensure all patients remain at healthy and safe blood glucose levels, with early detection of onset of complications.

Security and safety of the proposed services will be studied and necessary solutions to minimize risks and preserve privacy will be implemented. Legal framework for patient safety and liability as well as privacy and ethical concerns will be analyzed and an outline of a policy framework will be defined. Moreover, impacts on health care organizations and structures will be analyzed and health-economics and business models will be developed.

2.2 Purpose, context and scope of this deliverable

This deliverable describes the technical integration and test of the sensor devices and the REACTION application hosting client (AHD) developed in the REACTION project. The architecture of the REACTION platform and boundary of the present deliverable is illustrated in Figure 1.

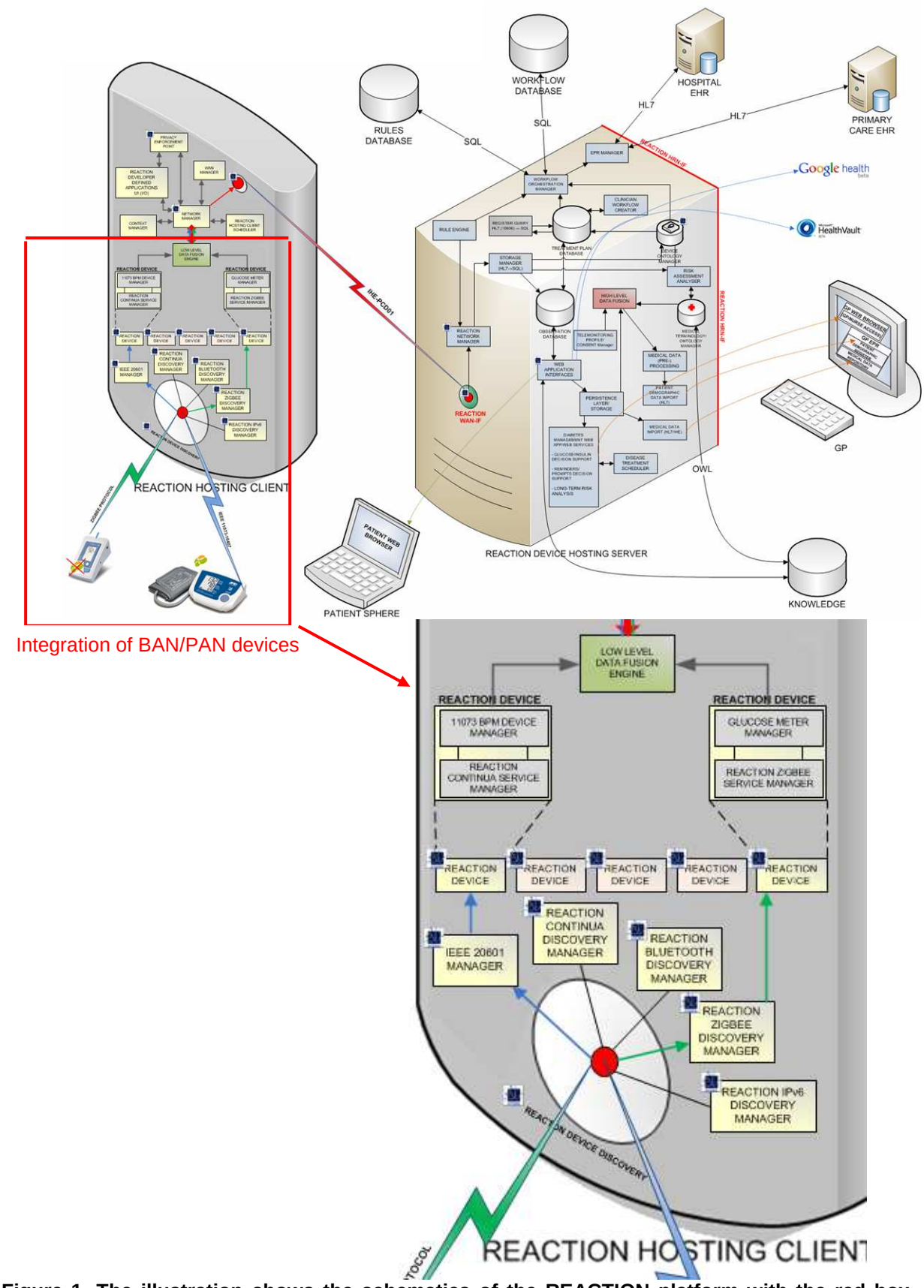

Figure 1. The illustration shows the schematics of the REACTION platform with the red box indicating the integration of BAN/PAN devices with the REACTION hosting client.

From this architecture the building blocks involved in the infrastructure of the body area network (BAN) and personal area network (PAN) will be the ones related to the sensors or the manager. The involved software modules are summarized in Figure 2.

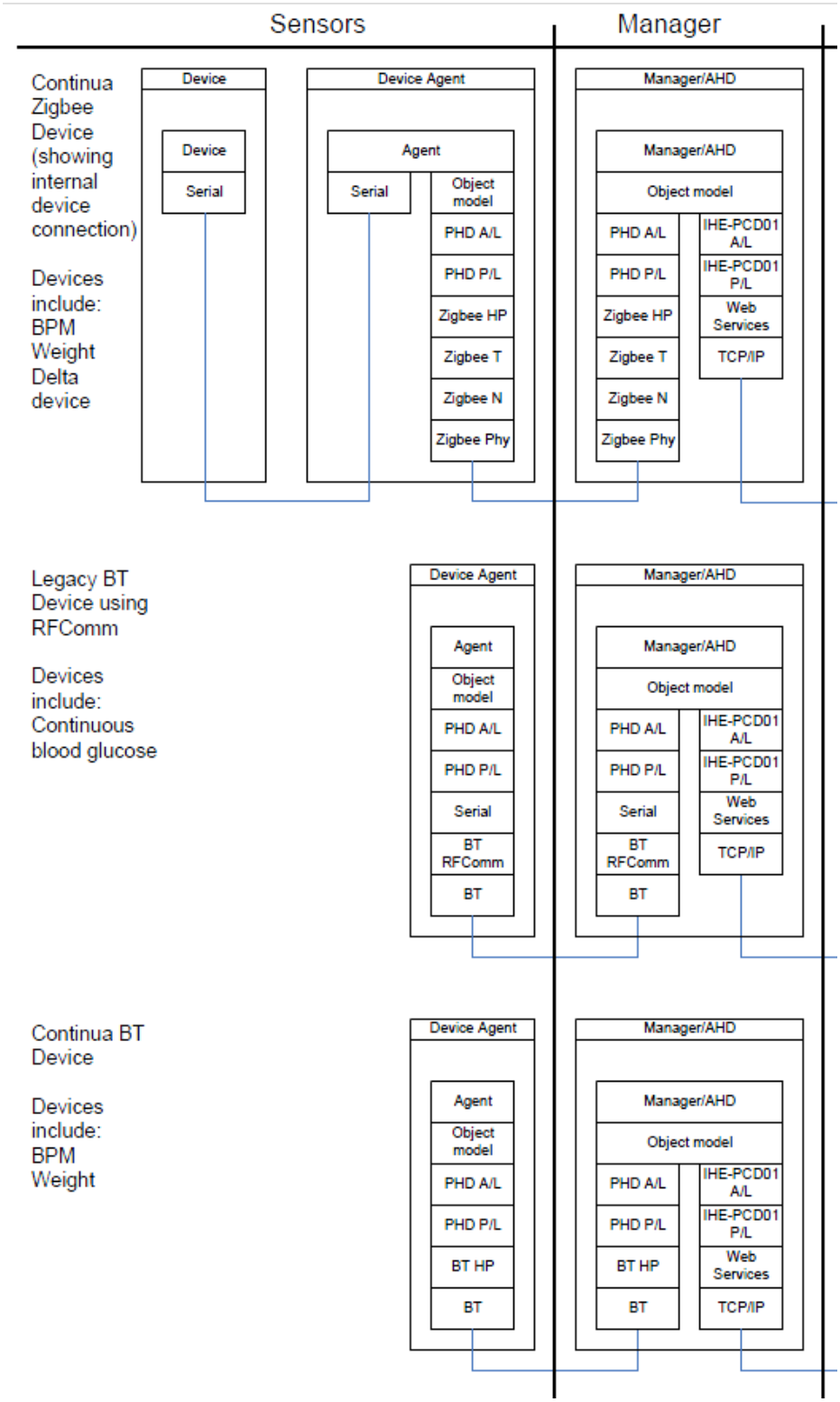

Figure 2: The illustration shows the software modules responsible for the networking in the BAN/PAN infrastructure and the software architecture of the networking functionality.

The involved devices can be Continua compliant or not and can be related to the body area (vital signs and health-related measurements) or to the personal area (environmental and ambient intelligence devices).In the next phases of the project the basic set of sensors for the different clinical

environments (in-hospital, primary care and closed-loop to the insulin pump) should be clearly defined as well as an extended set to be used for research purpose in the attempt to further improve the treatment and the clinical outcome.

These sets of sensors should include both body area sensors and personal area sensors because the integration of BAN & PAN (also for sustainability reasons) will be performed in the same node (the REACTION application hosting device).The software modules will implement all the domain information models which refer to the device types used in the platform.

The scope of this deliverable is to set forth a plan for the technical integration of components to a working whole and to set forth a test plan for testing individual components, interfaces, and the subsystem of the REACTION platform related to the sensors, sensor devices, body area network, and personal area network of the REACTION platform.

The deliverable considers the technical integration and internal testing of the technical functionality. In this context the integration and testing is independent of the application environment. Furthermore, the deliverable does not consider validation or compliance with legislation on medical devices and regulatory requirements by authorities in EU member states.

2.3 Methodology

In this section, a number of terms relevant for the deliverable are defined. The REACTION architecture consists of a number of subsystems each consisting of a number of components. The components and subsystems have interfaces to one another. This deliverable describes the plan for testing the individual components and a plan for integrating the components into subsystems. Since this deliverable only concerns the devices related to BAN/PAN integration, the deliverable will not consider the integration and test of the final REACTION platform. Figure 3 illustrates the definition of “Component”, “Interface”, and “Subsystem”. A Component is a single operating part of the larger system such as a sensor or network driver. Components interface to one another by defined interfaces, and a number of components form a subsystem which is to be considered as a functional part of the REACTION system. This deliverable deals with two subsystems Sensor Devices and the REACTION application hosting device as illustrated in Figure 1.

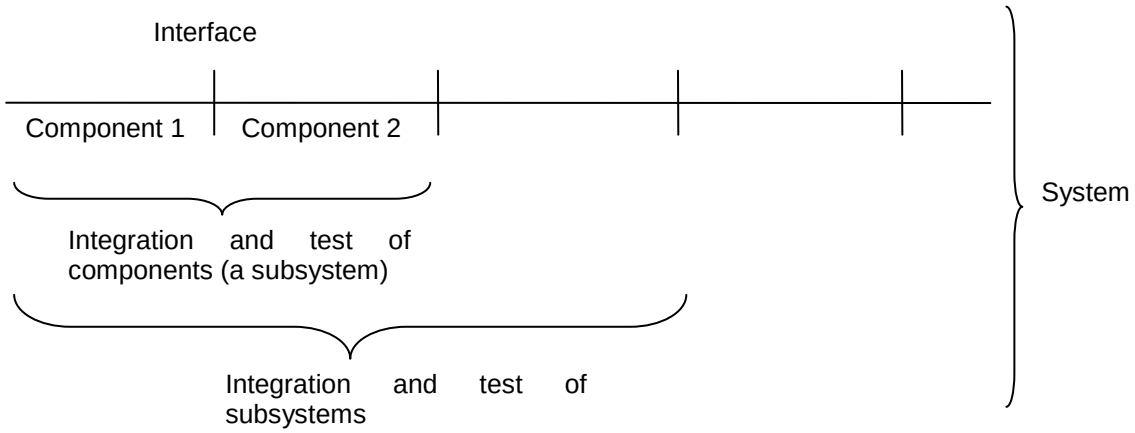

Figure 3: The figure illustrates the terms “Component”, “Interface”, and “Subsystem” as understood in this deliverable.

2.4 Organization of the document

The following chapter, **chapter 3**, describes the integration plan from a top level point of view organizing the sequence in integration and testing.

Chapter 4 presents the technical integration of hardware and software components on the sensor devices and the application hosting device. First, the individual components are described with definition of their interfaces, and then their technical integration on the sensor devices and application hosting in devices are described.

In **chapter 5** the test plan is developed. The test framework of the test methods is set forth. In context with the methodology presented above, the functional testing of 1) components, 2) integration of components, and 3) the subsystem of sensor devices and the application hosting device is described.

Chapter 6 summarizes the conclusions of this deliverable.

3. Integration plan

The development work of the REACTION platform follows an iterative process with an enhanced REACTION prototype each year of the project. Each prototype has an increasingly complex architecture and an increasing number of components and functionalities. The integration of the BAN/PAN part of the REACTION platform follows this iterative development scheme in a way that gradually increases the number of sensors and data management modules.

The overall development of the BAN/PAN infrastructure progresses according to the following priorities: First the application hosting device will be developed. This enables BAN/PAN networking with both commercial sensors and novel sensor technologies developed within REACTION. Second, the sensors will be integrated into the REACTION platform; third the data management modules will be extended to enable analysis and finally feedback to sensor devices which enable closed-loop glucose control.

The integration and testing of a developed component follows a four steps procedure:

1. Functional testing of the component to ensure compliance with the specified requirement as set forth in deliverable D2.5 and updates.
2. Technical integration of the component into the subsystem
3. Functional test of the interface to other components
4. Functional test of the subsystem.

In the following two chapters the technical integration and test of hardware and software components are described according to the above four steps procedure.

4. Technical integration

This chapter describes the individual components of the BAN/PAN part of the REACTION platform and it defines the interfaces between components for their technical integration. The interface can be mechanical, electrical, hardware-software, software and networking.
First, the individual component is described; and second, their interfaces and integration to one another are described.

4.1 Components

The REACTION platform has hardware and software subsystems. Each subsystem consists of a number of components. Components can interface to other components within the same subsystem or to components in other subsystems. Thus the interfaces between subsystems are defined by components.

In the following the hardware and software components are described. The hardware components are sensor devices and the software components are located both on the sensor devices and on the REACTION Application Hosting Device (AHD). Since the software for the REACTION AHD can run on a number of hardware devices such as, cell phones, and computers it will not be considered as a hardware device within the integration framework of this document.

4.1.1 ePatch and sensors

The ePatch is a sensor device in the REACTION platform which is currently under development. The ePatch has two parts; the first part is the ePatch processor which contains microelectronics and wireless communication technologies. The second part is the ePatch sensor embedded in an adhesive material which connects to the ePatch processor by a mechanical and electrical interface. The sensors can be either electrical for monitoring of electrocardiography (ECG) or optical for monitoring of oxygen saturation (SpO_2), and glucose. Figure 4 shows two pictures of the ePatch with an electrical sensor. Currently, the electrical sensor has been developed and the optical sensors are under development.

Figure 4: The pictures show the ePatch. The left picture show the ePatch processor connected to an ePatch sensor. The picture to the right shows an ePatch sensor with the mechanical and electrical interface for connection to the ePatch processor.

The ePatch is an example of a sensor device subsystem of the REACTION platform and its interfaces to the REACTION application hosting device with the purpose to transmit measured data. The ePatch sensors and ePatch processor are described in the following section.

4.1.2 Sensors

The ePatch sensors are based on electrical and optical transducer mechanisms. The ePatch sensors connect to the ePatch processor mechanically and electrically. The raw signals from the sensors are thereby transmitted to the ePatch processor for amplification, conversion, and wireless transmission to the REACTION AHD.

4.1.2.1 ECG

The sensor for electrocardiography (ECG) is an electrical sensor. It has three electrodes of low impedance hydro gel embedded in an adhesive material to form an adhesive sensor. The hydro gel is attached to conducting silver-silicone wires which pass through the mechanical connection interface to provide electrical connection to the ePatch processor. The hydro gel is the transducer converting the potential of the skin given by ion charges to a potential given by electrons. This conversion occurs as an electrochemical process between the hydro gel and silver-silicon wires. Figure 5 shows the adhesive ePatch sensor and an example of a measured ECG signal.

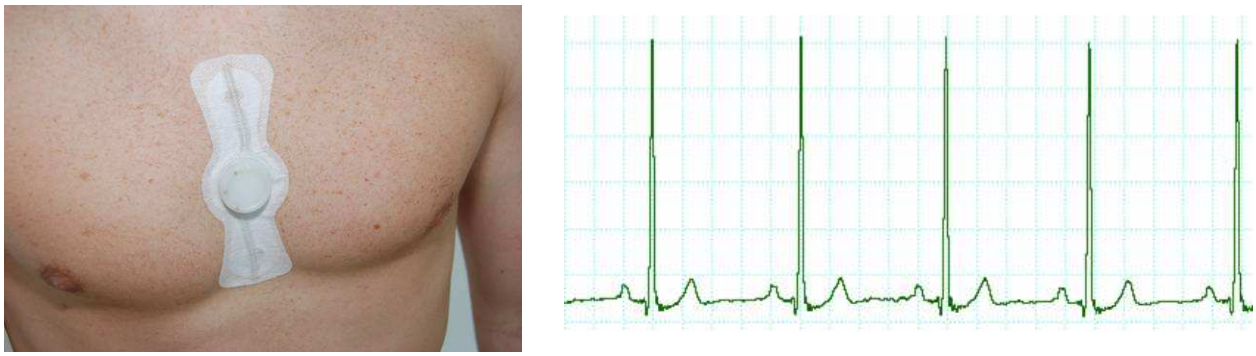

Figure 5: The picture to the left shows the adhesive ePatch sensor for ECG. To the right is shown an example of a measured ECG signal with this sensor and an ePatch processor.

4.1.2.2 Pulse oximetry

The pulse oximetry sensor is an optical sensor. Whereas the electrical sensor has the transducer separated from the ePatch processor this is not the case with the optical sensor. The active optical components for light emission and detection are located in the ePatch processor. The optical components are non-coherent light sources of light emitting diodes (LEDs) with wavelengths of 660 nm and 940 nm, and photo detectors for measuring light which is backscattered from the tissue. The ePatch processor also contains electronics to convert and amplify the signal from the photo detector. The adhesive ePatch sensor has an optical hydro gel embedded in the sensor which makes mechanical and 'optical' contact to the surface of the skin. The gel has a refractive index matched to the skin which limits reflection from the skin surface. The adhesive properties of the hydro gel make a mechanical contact to the skin which limits reflection artifacts e.g. due to air gabs between the sensor and skin surface. The sensor interface mechanically and optically to the ePatch processor. An illustration of the pulse oximetry sensor is seen in Figure 6.

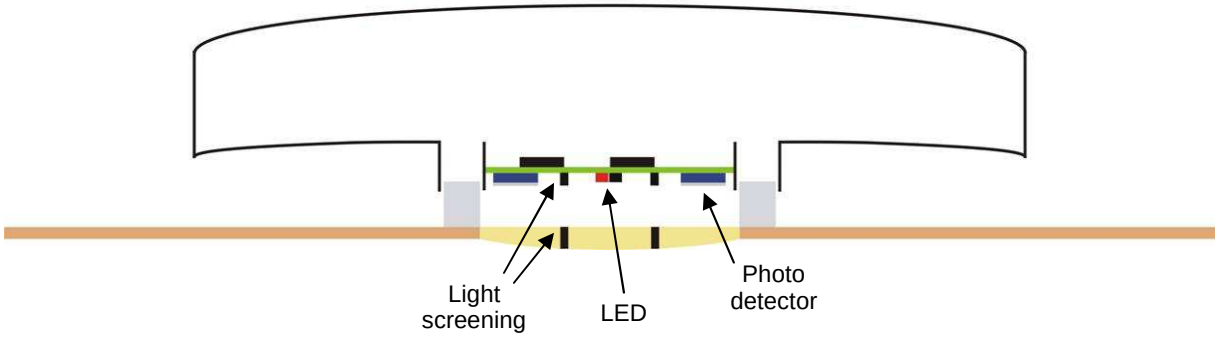

Figure 6: The illustration shows the ePatch sensor and ePatch processor for pulse oximetry. The actual transducer components are located in the ePatch processor. The ePatch sensor makes mechanical and optical contact to the skin surface.

4.1.2.3 IMM glucose sensor

Two different glucose sensors will be integrated into the REACTION platform, the IMM IR-spectroscopy based sensor and the SOLIANIS impedance spectroscopy based sensor (described in the following section).

The IMM glucose sensor is an optical sensor based IR-difference-spectroscopy and in its first approach will consist of a stand-alone unit exhibiting its own electronics with wireless communication, following the IEEE 11073 (CONTINUA) standard protocol.

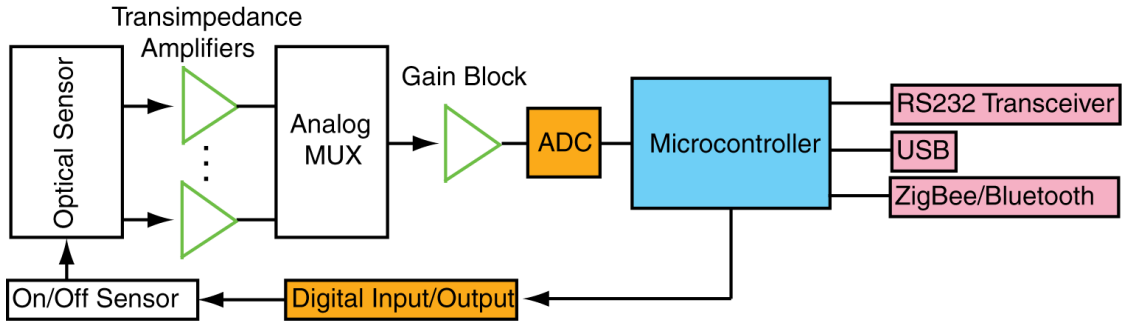

Figure 7: Schematic of glucose IR-sensor, including microcontroller unit and wireless communication via Bluetooth or ZigBee.

The sensor unit itself contains all relevant hardware, a glucose value is calculated via an integrated microcontroller and data (a single glucose value given in mg/dL) is transmitted via Bluetooth or ZigBee, using the IEEE 11073 standard to the REACTION discovery manager (Figure 7).
Since the first iteration glucose sensor will be combined with a micro-dialysis system, the first sensor hardware will also include a micro-dialysis pump and catheter which can be connected to the sensor unit (Figure 8). The sensor unit will be housed in an adequate manner and disposable chips (for the first iteration concept) will be delivered in sufficient number, to be connected to the sensor platform as well as the dialysis setup.

The dialysis equipment is a state of the art medical device with medical approval following the medical device directive (MDD). The IMM sensor will be undergoing a risk analysis after the MDD, before applied in clinical studies.

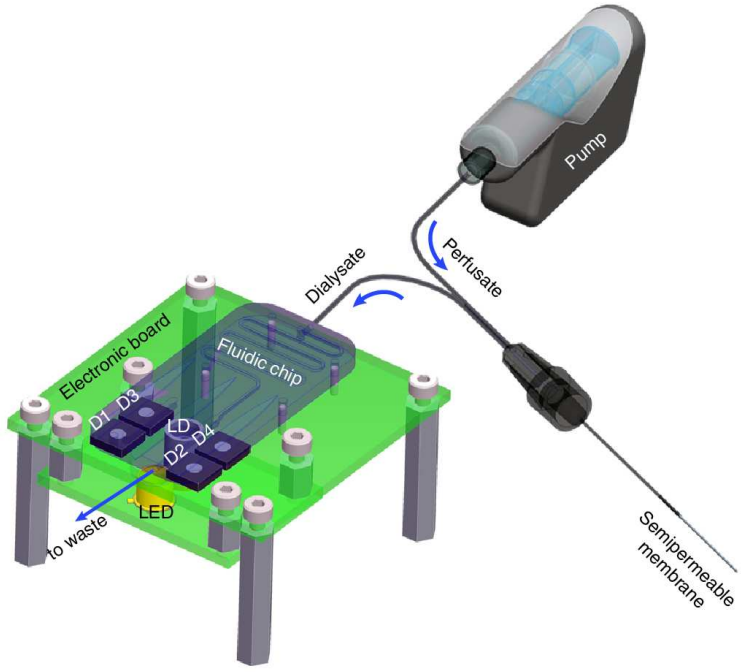

Figure 8: Schematic of the IMM IR-sensor unit together with the dialysis needle and the pump, respectively.

4.1.2.4 Solianis glucose sensor

The Solianis glucose sensor is based on non-invasive impedance measurements of the skin. In addition, a number of secondary sensors are used to correct from changes in interfering parameters;

this includes temperature, skin hydration, and perfusion[1]. The glucose sensor is therefore a multi-sensor system. This device is currently in the final phase of development and can thus be applied in the REACTION platform. The safety, usability, and performance of the Solianis sensor is documented in deliverable D3.3 “Solianis Continuous Glucose Management: System Description”. In relation to integration and testing of the Solianis sensor in the REACTION platform it will be handled similar to devices available from other vendors. The reason for this is that the Solianis device is well-documented with respect to internal components and performance as a sensor device including previous clinical testing.

This multi-sensor glucose monitoring system (MGMS) consists of two parts: A multi-sensor and a laptop. The multi-sensor non-invasively measures physiological parameters and transmit the raw data to the laptop. The laptop then compensates the data for inter-device variations, applies filtering, removes various artefacts, and calculates a signal describing variations in blood-glucose concentrations. The communication between these two units is ensured by means of a Bluetooth serial link. The communication protocol is proprietary; and the system is not designed to simultaneously communicate with external components.

Figure 9 demonstrates the MGMS configuration. The multi-sensor and the laptop form a unity and cannot be separated. The interface to the REACTION platform is through IP communication with the laptop. Currently, by an Ethernet cable, but the interface could possibly be offered by WIFI as well. The glucose measurement is communicated using the IEEE 11073-10417 standard (not officially released).

[1] Refer to deliverable D3.3 ”Solianis Continuous Glucose Management: System Description” for details.

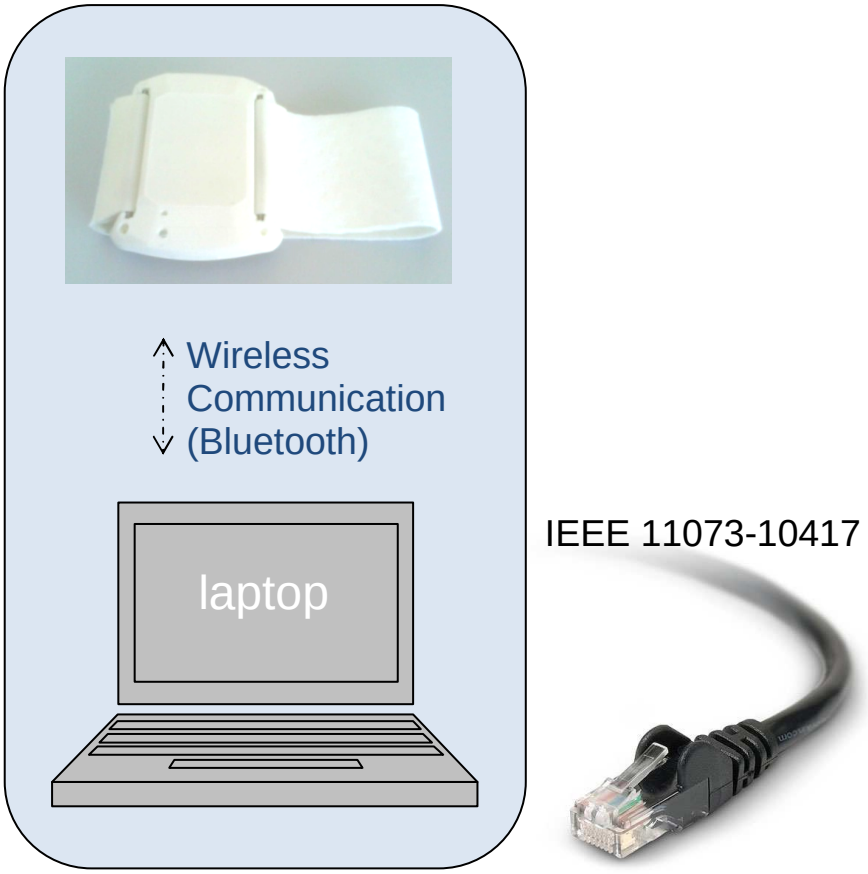

Figure 9: The Solianis sensor device consists of two parts, a multi-sensor attached to the body and a laptop which communicates with the multi-sensor using Bluetooth. The laptop calculates the glucose level and interface to the REACTION application hosting device by IP communication using the IEEE 11073-10417 standard. Currently, the interface is wired by an Ethernet cable, but the interface could possibly be offered by WiFi as well.

4.1.3 Networking on sensor devices

The networking of sensor devices is based on either ZigBee or Bluetooth, both operating at 2.4 GHz in the ISM (industrial, scientific and medical) band. The networking and data communication are based primarily on the Continua standards for networking between health care devices. These standards describe a fully defined communication structure between sensor devices and the REACTION application hosting device (AHD). ZigBee networking, Bluetooth networking and the Continua standards have been described in the REACTION deliverable D5.1.

4.1.3.1 Networking on the ePatch

The ePatch processor is equipped with a Texas Instruments CC2530 microprocessor. This microprocessor has a 2.4 GHz radio communication built-in for ZigBee networking. The ZigBee is based on the Z-stack from Texas Instruments and currently communicates in a private data format; however, it can be developed to become compliant with the Continua standards once all standards are completed by the Continua Health Alliance and underlying layers, e.g. the ZigBee health care profile for the CC2530 microprocessor, are available from vendors.

First, the networking component will be expanded with the ZigBee Health Profile when this becomes available from Texas Instruments. Second, the IEEE 11073-20601 communication standard and IEEE 11073-10406 ECG specialization will be implemented. The latter is currently in the final development

phase by the IEEE 11073 workgroup. It is expected that IEEE 11073-10406 will move to first round of ballot within Q1 of 2011.

The completion of the networking component of ePatch is thus dependent of the expected update of the Z-stack and the introduction of the IEEE 11073-10406 ECG specialization.

4.1.4 Networking of REACTION application hosting device

The REACTION application hosting device will need different networking capabilities with respect to which devices will be supported. We expect that the minimum supported protocols for device communication will be Bluetooth and ZigBee. Within these protocols we aim to support the different standards proposed by Continua. There might also be a need to support existing legacy or proprietary protocols that run over Bluetooth and ZigBee.

4.1.5 Software modules on REACTION hosting device

The architecture of the REACTION platform is depicted in Figure 1 and described in depth in the deliverable D4.2. It is foreseen in the Description of Work (DoW) that the architecture will be constantly evolving till the end of the project, based on the evaluation from each year's prototype and the feedback from the users of the platform, so the architecture depicted is the latest available version up to the production of this document.

We can divide the software modules of the REACTION platform to 2 main categories, the server side and client side modules. For the purposes of this document we focus mainly on the client side software modules, since these modules play the most important role for the integration of the BAN & PAN devices and infrastructure. According the proposed architecture, the client side of the REACTION platform is hosted to a device which acts as an integrator, data fusion and network gateway to the server side of the REACTION platform, where data is stored and further processed. These modules are based on the underlying Hydra middleware, adapted as necessary to fit the requirements of the REACTION project. Below we list the components that are most relevant for the BAN/PAN interface functionality.

- **Discovery Managers:** Discovery managers are responsible for detecting devices on the network that could be accessed by an AHD.
- **The REACTION Discovery Managers:** are based on the Hydra discovery architecture.
- **The Continua discovery manager:** a discovery manager for Continua compliant devices, part of the Continua standard.
- **Reaction Device:** This is the common device abstraction used in the AHD. It is a logical device implementation acting as a proxy for a corresponding physical device in the network.
- **Low level Data Fusion Engine:** This manager represents a high-level BAN/PAN interface delivering integrated device data.

The data from the various devices and sensors is collected from the application hosting device (REACTION AHD) using several kinds of wireless transport protocols such as Bluetooth, ZigBee etc. The transport protocols, their characteristics and their applicability to REACTION are analyzed in more detail in the deliverable D5.1. The Device Discovery module is responsible for discovering the devices in proximity which are supported and relevant to the REACTION platform. For this reason, the Device Discovery is comprised of several dedicated Discovery Manager modules for each one of the supported protocols. If it is indented to support more transport protocols in the future then there will be the possibility to implement the corresponding Discovery Manager and attach it to the general Device Discovery mechanism.

When the Device Discovery discovers a new device or sensor, through the appropriate Manager, it creates a new instance of a REACTION device. This instance is a wrapping module which encapsulates the functionalities of the newly discovered device and establishes a connection for the delivering of data between the device and the REACTION AHD.

The data collected from all the surrounding BAN & PAN devices, is gathered and fused together in the Low Level Data Fusion Engine. This is a low level process, since there is also the equivalent High Level Data Fusion performed in the server side modules of the REACTION platform. This low level fusion serves mostly on the basis of rejecting artifacts and integrating the data according to some common data structure taxonomies (see D4.2) which adhere to established data transport formats such IEEE 11073, HL7 standard and the Continua guidelines.

Furthermore, the data is sent to the server side of the REACTION platform through the Network Manager module, by utilizing the available network communication technologies (see D5.1).

4.2 Integration of components by interfaces

In the following sections the interfaces of components and their technical integration are described. First, the assembly and integration of the various components of sensor devices is described, and second the integration of the sensor devices and the REACTION AHD is described.

4.2.1 Sensor devices: Sensors, ePatch, and networking

The sensors developed within REACTION consist of three components: A sensor component, an ePatch processor with electronics, and a software component for networking embedded in the microprocessor of the ePatch processor. The sensors are either based on an electrical transducer or an optical transducer. The integration of the sensors is described in the following sections.
The ePatch processor connects ePatch sensors with the REACTION AHD. Other sensor devices e.g. the Solianis device or commercial devices (e.g. the Roche Acccu-Check) connect directly to the REACTION AHD. The networking is implemented within the ePatch processor as described in section 4.1.3.1 and the interface with the REACTION AHD is described in section 4.2.2.1.

4.2.1.1 ePatch with electrical sensors

The interface between the ePatch processor and the ePatch sensor with the electrical transducer is 1) mechanical and 2) electrical. The mechanical interface is of a snap-fit. The electrical interface is an analog connection made by silver-silicone wires as described in section 4.1.2.1

4.2.1.2 ePatch with optical Sensors

The mechanical interface between the ePatch processor and the ePatch sensor with the optical transducer is 1) mechanical and 2) optical. The mechanical interface is similar to the mechanical interface for the electrical sensors. The optical interface is an optical transmission channel formed in the ePatch sensor by hydro gel. From a signal point of view the optical ePatch sensor interface to the ePatch processor electrically where the interface is at the output of the optical detector.

4.2.2 Sensor devices and REACTION application hosting device

4.2.2.1 Networking

The network integration should be simple as long as all devices use standard protocols such as ZigBee and Bluetooth. Though even if there are standards there can still be problems with interoperability between Bluetooth stacks and different ZigBee implementations. Therefore a test-bed to test the networking between the different sensor devices and the REACTION application hosting device should be created. The integration tests should include:

- Discoverability; i.e. that the sensor device is discovered.
- Robustness; i.e. that the communication does not suffer from breakdowns
- Interoperability; i.e. that using multiple sensor devices and protocols does not cause problems.

4.2.2.2 Software modules

The integration of the various software modules is a process which consists of several steps and involves various layers and aspects, such as the functional integration, the data format integration and the semantic integration. As noted in the section “Integration plan” of this document, there are some distinct phases until the integration of the system as a whole, and it is also an iterative process which will be repeated for every prototype release until it converges to the final prototype.

The hardware components, at least those that are not developed by the REACTION consortium, have a specific functionality, communicate by using specific protocols and adhere to specific standards or technological restrictions. As a consequence, much of the flexibility of the REACTION platform is imbedded in the software modules which means that the proper architecture and software design is important for the integration of the platform. It is to the advantage of the project that some of the software modules are already available as parts of the Hydra middleware and this means that these modules have already been tested for their functionality and some of them can be used “as they are” or with minor adaptations. The proper design and implementation of the rest of the software modules will bridge the gap between them and eventually facilitate the seamless integration of the whole platform.

The integration of the software starts from the technical integration of the available components and hardware. By using various software development tools and automating some of the technical procedures, we speed up the integration of these modules and ensure a common model and common techniques. In a following phase, the functional integration of the components is facilitated by adopting a common architectural model (D4.2), a common data model (D4.2) and a testing and validation plan (D2.7) which will expose the functional problems.

4.2.2.3 Data structures

The integration of data structures, i.e. the message formats, will be mostly based on the Continua standards. This means that the integration of data structures can be done with respect to existing standards. Therefore the initial integration tests can be done with respect to standards compliance. Of course there is also a need to test the integration between sensor and the REACTION hosting device in order to ensure see that the correct data is transmitted.

For the non-Continua devices there is a need to test the data structures between the sensor and the REACTION application hosting device already at the initial testing phase if the data format is proprietary.

4.3 Integration Infrastructure

This chapter describes the infrastructure required for the realization and implementation of the integration plan of the REACTION platform.

4.3.1 Configuration Management: Subversion

An important part of modern software development is the Software Configuration Management. Configuration Management is often considered to have originated in the frame of software development in recent years and that it merely consists of a tool for versioning of source files. In software development the core of configuration management is made up of a version control system. Source code (or all source documents of the final product, respectively) is considered a system that spans time and space. Files and directories (which can also contain files) form the space, while their evolution during development forms the time. A version control system serves the purpose of moving through this space.

Configuration management contains version control, and extends it by providing additional methods of project management. Software configuration consists of software configuration items (SCI) which can be organized in three categories and which exhibit several types of versions. The following activities constitute configuration management:

1. Identification of SCIs and their versions through unique identifiers
2. Control of changes through developers, management or a control instance
3. Accounting of the state of individual components
4. Auditing of selected versions (which are scheduled for a release) by a quality control team.

The Institute of Electrical and Electronics Engineers (IEEE) sees configuration management as a discipline that uses observation and control on a technical as well as on an administrative level. Software configuration management deals with the governing of complex software systems. In

REACTION we base the software configuration management on Subversion Source Code Management with Subversion (http://subversion.tigris.org/), further detailed in the following sections.
Continuous Integration has its origin in the concepts of extreme programming, where it is common practice to immediately commit every change to a revision control system, no matter how small it is. The rationale behind is that other developers should always work with the latest version of the code base. For source code management we have set up a Subversion (SVN) repository where all artifacts of the development process will be stored and organized. Subversion, like many other version control systems, manages different versions of documents by calculating editing operations between these and then only saving these operations. This approach usually leads to very good compression results.
We use Subversion because it is a modern revision control system. It has been designed to resolve many of the short-comings of widely-used Concurrent Versions System (CVS), while still being free software. Subversion attempts to fix most of the noticeable flaws of the CVS and is very powerful, usable and flexible. New to Subversion is the way how it internally treats directories and documents: directories (which are also documents) only contain links to specific revisions of other documents. Thus, when a source file is copied, Subversion will not copy all the data, but rather create a new directory entry referencing the same object. Hence, it is possible to copy the entire development branch at very low cost.

Access to the Subversion code repository is provided through secure SSL connections using our web server. Thus, this repository allows the geographically dispersed group of consortium members to collaborate and share their source code. It tracks changes to source code and allows a versioning of source code files. Additionally, Subversion remembers every change ever made to the files and directories so that earlier versions can be recovered in case that something was accidentally deleted.

4.3.2 General Setup of Subversion for the Project

The Subversion repository created for the REACTION project comprises three subdirectories:

- *branches*, i.e. for short time branches
- *tags*, i.e. a full copy of the source code for releases
- *trunk*, i.e. for the working source code as the current development version

Trunk is going to be the development branch where we will be storing our most recent version of the code. The different releases of the code will be stored in the Branches directory. For example, if a developer wants to create the Release-Version-1.1.0 of the code then she creates the directory in the Branches section and copies the most recent code stored in the trunk section to the release directory.
The *Release* branch will be used to correct the errors in the code during the alpha/beta testing phase. All the commits at this stage will take place in the Branch directory. When the developers are done with this they will copy this code to the tag directory and will name it as release version 1.1.0. Tag is defined to be a copy of the repository at any given moment. Although there is not much difference between the branch and tag, the tagged code will act as the final release of any given version (i.e. public release).

The common structure of the middleware directory of the SVN trunk currently comprises the following subdirectories:

- *bin* containing all dlls
- *src* containing all Java and .Net source files
- *lib* containing all library files
- *include* containing all include files
- *project* containing all Visual Studio Project Files
- *docs* containing all documentation corresponding to this release.

Version control is generally independent of the operating system and programming language used. The standard management program “svn” for command line usage is available for different platforms, including Linux and Windows. Subversion integrates well with programming languages that have C-like syntax and source file schemas. It is important to note that a revision repository should contain all files necessary to build the software system, but should not contain any files that can be derived from other files in an automated fashion. Such redundancy only pollutes the repository and is considered bad practice.

There exist differences in the integration of SVN support into the integrated development environment regarding the programming languages Java and .Net, which are addressed by the following paragraphs.

4.3.3 Subversion and Java

Besides the management of Java sources using the command line tool (which, of course, can be used for revision control of almost all kinds of software), integration of Subversion into Java IDEs is quite advanced. Most IDEs natively support Subversion. This is partly due to open source Subversion libraries, but also due to the fact that Subversion is following the success of CVS, especially with regard to open source software.

The integrated development environment *IntelliJ IDEA* supports Subversion out of the box. The integrated development environment *Eclipse* does not directly support Subversion, but a Subversion library is available. Eclipse plugins are easily installed with its plugins manager. However, Subversion support in Eclipse is not yet as advanced as CVS support.

4.3.4 Subversion and .Net

Microsoft has its own configuration management tool Visual SourceSafe in its portfolio, and hence, support for other revision control software cannot be expected natively. However, there exist other free and commercial plugins such as AnkhSVN and VisualSVN, which are available for Visual Studio and thus can be used for Subversion configuration management of the .Net developments in REACTION.
In addition, the command line tool TortoiseSVN allows for a configuration management of any directory in the file system. Furthermore, TortoiseSVN hooks into the Windows Explorer and allows for a more graphic-based checking in, updating and committing of changes. However, there is no further support such as integration of SVN functionality into an integrated development environment.

5. Test plan

In the following the testing of the technical functionality is described. Issues related to safety and compliance with medical device legislation is not part of the scope of this deliverable as set forth in the DOW. These issues are described in the device specific deliverables: D3.2.1 “First Generation ePatch”, D3.3 “Solianis CGM: System Description”, and D3.5 “Prototype of an IR-spectroscopy based CGM Sensor”. The technical functionality test is executed on four levels:

1. Individual components
2. Interfaces
3. Integration of components
4. System wide test

Sensor devices which are not developed within REACTION such as commercial glucose sensors are only tested with respect to their interface to the REACTION application hosting device. Sensor devices developed within REACTION are broken down into a number of components based on functionality as described in section 4.1. This is illustrated in Figure 10 for the ePatch.

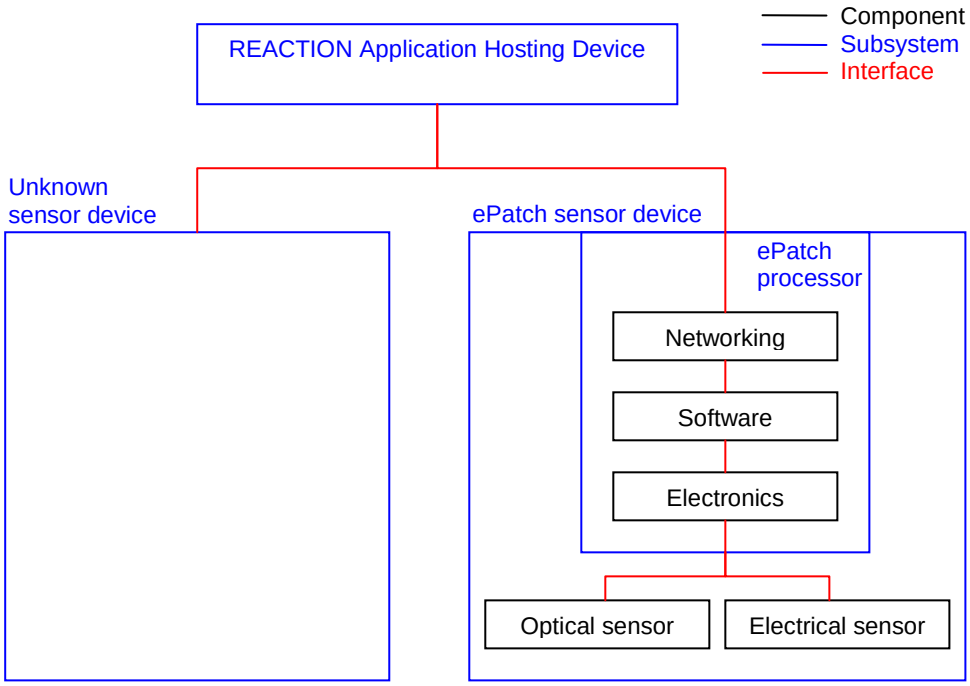

Figure 10: The illustration shows the break-down of the sensor devices into the components and interfaces that are tested.

For each component and interface the requirements, according to the Requirement Specification, are identified. Each requirement is broken down to a number of test cases based on the acceptance criteria of the requirement. An example of test cases for a sensor is given below.

Test case ID	CPNT_SENSOR_GLUCOSE_IR_001
Requirement	
Author	
Test case description	In-vitro test of the temperature range of the glucose sensor.
Component or interface to test	Near infrared optical sensor for glucose detection.
Assumptions	Temperature in the range 34 C to 39 C
Test input	Samples in the range of ZZ to QQ with steps of PP.
Acceptance criteria	Range is XX to YY.
Passed / Failed	
Comments	

Test case ID	INTF_ELECTRONICS_SOFTWARE_001
Requirement	
Author	
Test case description	The interface between the analog electronics for the sensor and the software is tested i.e. correct conversion of the data in the A/D converter and correct data representation in the software
Component or interface to test	Interface
Assumptions	
Test input	Applied voltage in the range XX to YY.
Acceptance criteria	Voltages in the range XX to YY is converted of a full 12 bit value.
Passed / Failed	
Comments	

In the following section the test methods applied to test the technical functionality of each type of component and interface are described.

5.1 Component and interface test methods

Testing of individual components of a system is essential prior to testing an entire system. However, before the individual components or subsystems can be tested, their interfaces must be specified. Components interact with each other and thus their interactions must be specified in order to test the individual components' reactions on inputs and to check the generated outputs according to specifications. The interfaces between the components may be of various kinds e.g. electrical, optical, mechanical or other.

5.1.1 Sensor devices

5.1.1.1 Sensors

The test methods applied in the technical functionality testing for the sensors developed within REACTION are based on in-vitro tests, approved in-vivo tests, and stability tests.

- **In-vitro** tests are the first approach in testing the functionality of a sensor. The in-vitro tests are based on either classified samples, samples that are checked with the golden standard, or simulated data. For the glucose sensor the sample based methods are used whereas for the ECG sensor simulations of bio-potentials are used.
- **In-vivo test** requires approval of the local ethics committee and need to comply with the Helsinki Declaration, EU legislation, and national legislation. In-vivo tests represent the final step in testing a sensor. The in-vivo tests are designed based on biostatistics principles. The approach is to apply and test the sensor with healthy subjects before proceeding to patients.
- **Stability tests** are performed both with in-vitro and in-vivo test. The tests make sure that the sensor functions are specified under the conditions that it will be exposed to when in use such as elevated and lower temperatures, humidity etc.

The interface to electronics is tested by standard electronic tests, described below.

5.1.1.2 Electronics

The tests of the electronics start during the design phase. In this phase relevant simulations will be carried out to verify amplifiers and analog filter designs. The simulations should give a basic knowledge of the physical functionality of those parts of the design.

After the production and assembly processes are added to the electronic design it is separated into functional and testable blocks each with a well-defined functionality i.e. an analog input voltage causing an expected output voltage. Due to practical reasons the electronics cannot be split into physical separated blocks for the testing of each function separately. The test will therefore be carried out as a hierarchically test starting at a top level test monitoring supply voltage and supply current. At a subsequent level strategic measurements are carried out throughout the electronics design having the design divided into further functional blocks. The measurements might include analog input and

output voltage levels and/or ranges, voltage levels at bus infrastructures, current measurements, quality of crystal oscillations etc.

5.1.1.3 Embedded software

The test of embedded software components will be carried out in successive steps as the software modules are developed. The tests include the following steps.

1. **Test of each subcomponent of a system component**. This includes tests of interfaces, i.e. to test if the software modules exchange information as expected. These tests mainly concern internal component information exchange and may reveal e.g. if a software module does not behave as specified. This can be detected by analyzing an output data value via interfaces.
2. **Interface test** based on simulated input data between components. These tests will test the embedded software components in relation to their interoperability with electronics.

5.1.1.4 RF and Networking

Testing of RF and networking is a two-step process: First the electromagnetic interoperability is tested i.e. operating frequency and antenna coupling and second the networking protocol. The networking protocol prescribes and organizes exchange of data i.e. addressing and connecting devices and exchange of data in applications. The fundamental electromagnetic operability is typically defined by components acquired by vendors (e.g. antenna and radio microchips). The focus in testing the RF and networking is therefore on the protocols which relates to data exchange. Therefore, the testing of the RF and networking interfaces is an issue of validating information exchange which is similar to a software test. In relation to REACTION, the protocols which define the interface between sensor devices and the REACTION AHC are the Continua standards and private data protocols e.g. applied in the first generation ePatch. These are therefore the protocols to test against.

5.1.2 Software modules

The testing plan and techniques that are described in the D2.7, regarding the overall REACTION platform can be also applied to the BAN & PAN subsystem integration.
First of all, we can utilize, when available, the development kits and boards which many times come with devices and sensors in order to enhance the development process. But usually such developments kits are not available, they can't be fine-tuned as needed or their cost is prohibitive. For this reason, a flexible tool is the simulations of devices and sensors for the testing of software. A custom built piece of software that simulates the behavior of a device and reproduces its communication protocol is a great tool to build its fitting piece of software, a device driver or communicating module. Specifically, for the integration and testing of BAN & PAN devices, the simulating module can be hosted as a REACTION Device instance in the AHD, functioning in equivalence as wrapping a real device and generating plausible data.

Of course this is not an easy task and many times it requires a lot of effort, depending on the complexity of the device, but it can prove a very helpful technique, especially to the early stages of the implementation of software, because it can simulate specific series of data, simulate custom or exceptional cases, produce repeatable results and its internal functioning can be fine-tuned as needed with greater easiness than of the corresponding hardware. Secondly, this can help in the debugging process as it can log and repeat specific series of events, or situations. Depending on the device, a simulator can usually be a lot cheaper, and can also be used for remote testing between collaborating partners, in situations where using the device itself could pose physical restrictions.

Furthermore, for the testing of the software we can utilize unit testing tools, such as JUnit or any other from the xUnit family of tools. With unit testing, we can formulate test cases which test the system's functionality against the requirements set by the users. The user requirements, described in depth in D2.5, can be formulated in the form of preconditions and post conditions of several functions, so essentially we could think of the unit testing process as a special case of simulation testing tool, as unit tests eventually do the same thing, that is to simulate specific tasks or series of events and provide feedback to the developer. Again, for the same reasons as the simulators, unit testing is a valuable tool because it guarantees consistency and repeatability of the results. There are available tools which can automate the building and running of the unit tests. It can be utilized both for the testing of distinct components but also for the testing of integrated components and greater sets of modules. The unit tests could be composed by a series of small tests which follow a workflow of

events or repeat the steps and functions of a specific scenario, in order to test the functionality of the system as a whole.

5.1.2.1 Software unit testing

A "unit" is in the current REACTION architecture a Manager (and possibly parts of a manager). Unit tests must be automated and be written using a unit testing framework, e.g., in the case of Java this could be JUnit; In the case of .NET: NUnit.Upon completion of an increment of a unit, the following must be considered:

- Code checked into the Subversion repository must not break the building process
- Prior to committing new versions of a unit to the Subversion repository there must be reasonable automated, functional unit tests. We do not prescribe any specific coverage criteria for black box or white box tests.
- Subversion 'Commit'-messages, should explicitly mention Requirement references (like 'REACTION-323') to allow JIRA tracking.
- Each project must provide an Ant build file with a “build” target for building the project
- Each project must provide an Ant build file with? a “test” target for performing unit tests. The project should provide and use the necessary stubs (considering any manager dependencies)

Reported errors should be corrected as soon as possible. To enhance quality of the software it is recommended to create a new unit test for each detected bug if this was caused not by an error in the programming but in a misunderstanding of a requirement. After fixing the bug, the commit statement has to contain the bug number so that an automated tracking of bug fixes can be performed.

5.1.2.2 Networking, BAN and PAN

To test the integration between the sensor devices and the REACTION hosting device there is a need to setup a testbed. The testbed should primarily be used for the following test:

- Sensor discoverability, i.e. that the REACTION AHD discovers the sensor
- Data transmission, i.e. that the sensor readings are received correctly at the REACTION AHD.

Interoperability between devices, i.e. to test that multiple sensor devices connected to the same REACTION AHD does not interfere with each other.

5.2 Integration testing

When all system components have been developed their interactions should be tested to see if the system adheres to system specifications. This is system integration. In this project, many components of various kinds need to interact with each other to form the desired system behavior.

5.2.1.1 Sensors and electronics

At this stage of testing the input and output signals of each functional block will be tested against the associated blocks in the design and as a complete system. The blocks of concern will interface between local blocks at the printed circuit board (PCB) and/or in the sensor unit and/or to global blocks through i.e. a wireless link. Block tests might include data transfer, data conversion, data acquisition, analog filtering, signal quality and SNR levels.
At this level of testing the integration with the embedded software will be carried out as well.

5.2.1.2 Sensors, electronics, and embedded software

Already tested sensors and electronics, will be used to test the functionality of the embedded software with real data. This may shed light on undetected misbehavior in the individual components not taken into account previously. It may also reveal actual flaws, e.g. in the electronics when the embedded software is using the electronics to make data acquisition. In addition, only when all components of a system are in place the functionality of the entire system can be experienced in full and therefore possible flaws and errors in the system will only occur and hence be detected when the system is working as a unit.

For the embedded software part, testing this component will be based on real input data as other interacting components become ready. This may include data acquisition, data processing, data storing, and data transfer?.

5.2.1.3 Sensor devices and software modules

In the basis of the testing of distinct components, the same tools and techniques can be also applied for the testing of the integrated components. First of all, a testbed will be set up consisting of a set of devices and sensors, to simulate the possible situations of the BAN & PAN that the REACTION platform will be dealing with. A listing of the reference devices should be agreed among the technical partners, so that, if possible, to set up the same testbed to the premises of more than one partner for the sake of collaboration and efficiency. In case that it is not possible or affordable, simulations of the devices or their functioning could be utilized in their replacement.

Furthermore, unit testing on a higher level can be used, not only to test certain functions of a component, but also combination of functions, series of events and usage scenarios. As noted also in D2.7, formal test cases should be written for the testing of the system that should resemble the common clinical practice in order to validate through the testing procedure the accordance of the integrated system with the needs of the final users.

5.3 System testing

On a system level (e.g., with an integrated middleware implementing a sample application), use case/high level functional requirement and non-functional/quality requirement testing can be tested. These tests will be automated whenever possible.

One advantage of the Volere schema is the fit criterion that specifies in a measurable way if a requirement is met or not. In some cases the fit criterion is measurable but cannot be measured by automated tests (e.g. hardware failure detection in 3 out of 4 cases). In these situations one has to revert to manual tests. However, in the majority of cases it is possible to compute a measure and compare it to the fit criterion and calculate the coverage of successfully met requirements.
A best practice in integration and system testing is that the developers who wrote parts of the system/integrated unit should not be the ones who write and perform the tests. Given that REACTION is a large, complex project it will almost always be the case that an integrator did not write part of the units being integrated and thus the integrator is permitted to write and perform the tests.

5.3.1 Security

Security relates to various layers of a system, and depends on many different factors and composing techniques. Security can relate to the radio level security and the possible encryption of information to the physical layer of the network, especially in the case of wireless networks where eavesdropping is easier. Usually this is provided through the usage of appropriate protocols that provide physical layer security and our role is somewhat restricted on choosing a protocol which provides the necessary level of protection. Furthermore, security can be applied at the transport layer through the usage of secure connections. Again, the security on the transport layer is mainly based on the usage of publicly available tools and protocols and their testing is out of the scope of this project, since their credibility is widely established. On an even higher level, the WS-* family of protocols can be utilized, such as WS-Security, WS-Policy and WS-SecureConversation.

In essence, much of the security aspects of the project and especially of the BAN & PAN integration subsystem are focused on the higher level techniques and policies that can be used additionally to the low level security mechanisms. To this end, the Policy Enforcement software module plays an important role to monitor and guarantee that the information exchanged with the server side adheres to certain policies which are applicable when dealing with sensitive data such as medical information.
What's more, security applies not only to the confidentiality and privacy of the exchanged information, but also to the integrity of data. Since the medical measurements could be used from the decision support tools and algorithms to assist the medical personnel, errors in the integrity of the data could lead to the depreciation of the system or even dangerous situations as a result of malfunctioning caused by erroneous transmitted information.

5.3.2 Web Service Performance Testing

Web service performance testing is mainly focused on the scalability and robustness of Web Services in a Service-Oriented Architecture (SOA). Testing here implies calling target Web Services with varying SOAP messages across a range of concurrent loading clients.

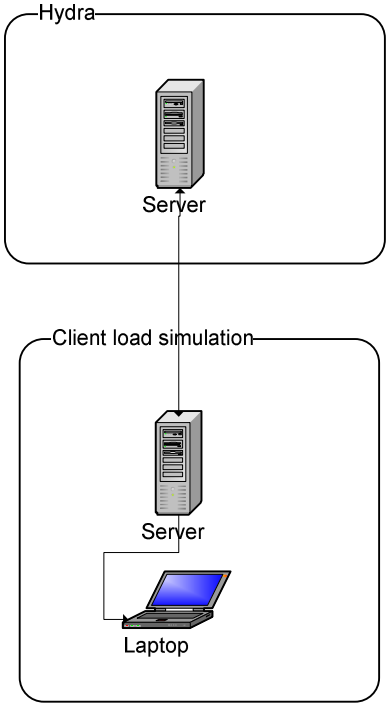

Figure 11: Performance test setup.

Performance testing will be executed using test software that creates multiple simultaneous processes which perform web service calls and then times the result. The result will basically provide information on how well a specific manager copes with increasing load and at which point it does not fulfill performance targets.

For these test, we manually monitor the memory consumption in order to detect memory is a problem. In principle the chosen development platforms should not have any problems with undesired memory leakage, since both the Java and .Net platforms contain garbage collection of unreferenced objects. But for instance badly designed caching of data, might still cause problems with memory consumption. Furthermore, resource efficiency is critical for embedded systems.
Making complete tests for all methods on all services is a very major task so we choose to performance test those functions we inherently know might have performance problems since they contain more advanced functionality.
Performance testing of the complete system will be performed with integrated systems (i.e., demonstrators), since it is hard to test and time simple actions in event-driven systems given that different events might trigger very different actions depending on the given context.

5.3.3 Web Service Interoperability testing

Interoperability between services is very important to the REACTION platform. This is partly because the platform will be developed in different environments as well as on different platforms (Java and .NET), but primarily because we want to have an open architecture which is based on SOA. It should be easy to replace and extend services without using a specific programming platform as long as one adheres to standards of Web Services.

The interoperability tests to be made on REACTION platform can be divided into two main groups:
Testing done on WSDL files between managers during development and integration.
More formal testing of managers using WS-I tools. These test both design time interoperability (based on a WSDL file) and run-time interoperability (whether the web services responds according to WS-I at run-time).

5.3.3.1 WS-I Profile conformance test setup

The Web Services Interoperability Organization (WS-I) has developed testing tools that evaluate Web services conformance to Profiles. These tools test Web service implementations using a non-intrusive, black box approach. The tools focus is on the interaction between a Web service and user applications.

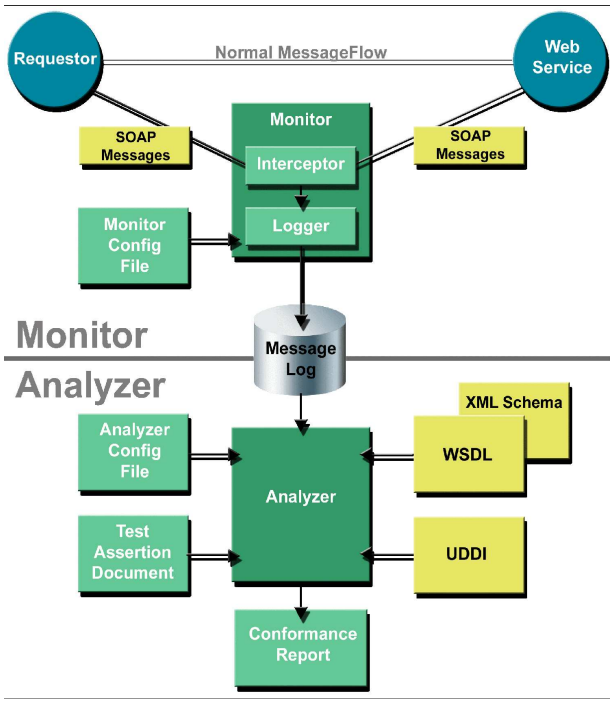

Figure 12. WSI test setup.

5.3.3.2 Testing Tools Architecture

The testing infrastructure is comprised of the Monitor and the Analyzer and a variety of supporting files:

- *Monitor*: This is both a message capture and logging tool. The interceptor captures the messages and the logger re-formats them and stores them for later analysis in the message log. The monitor is implemented using a man in the middle approach to intercept and record messages.
- *Analyzer:* This is an analysis tool that verifies the conformance of Web Services artifacts to Profiles. For example, it analyses the messages sent to and from a Web service, after these have been stored in the message log by the Monitor.
- *Configuration Files:* These are XML files used to control the execution of the Testing Tools.
- *Monitor Configuration File:* Controls the execution of the monitor
- *Analyzer Configuration File:* Controls the execution of the analyzer.
- *Test Assertion Document:* Defines the test assertions that will be processed by the analyzer.

Other files or data artifacts will be accessed, which are not part of the test framework, but dependent on the Web Service to be tested:

- *Web Service artifacts:* These inputs to the Analyzer are target material for testing, and will be reported on:
- *Message Log:* Contains the monitoring trace of messages captured at transport level.
- *WSDL definitions:* Contains the definitions related to the
- Web Service *UDDI entries* - contains references to Web Service definitions, as well as bindings.

- *Generated Files:* These are XML files produced by Testing Tools, that are specific to the Web Service being tested:
- *Message Log:* (also a "Web Service artifact")
- *Conformance Report:* Contains the complete conformance analysis from the specified inputs.

As the REACTION platform is based on two different environments (Java and .NET) we expect to have initial interoperability problems with regard to the WSDL files and service invocations. In order to understand a conformance report output one needs to understand that WS-I defines a number of assertions it checks. Table 1 shows an example of such an assertion.

Assertion: BP2416

Result	**passed**
Assertion Description	Every QName in the WSDL document that is not referring to a schema component, is either using the target namespace of this WSDL or the target namespace of a directly imported WSDL component.

Table 1 Sample WS-I assertion.

Section 4.2 contains an example of a test report.

6. Summary

This deliverable has presented the overall integration plan and test plan of the BAN/PAN infrastructure of the REACTION platform. The individual functionality and interfaces of the individual parts for the BAN/PAN infrastructure have been described. A detailed test plan with description of test methods for individual components and interfaces has been presented. The Patch and associated sensors for ECG, pulse oximetry, and glucose level have been presented. The assembly of the ePatch with these sensors and the integration with the REACTION AHD has been described. The ePatch is, in a first approach, integrated with the REACTION AHD by a private communication protocol. In a later approach, the ePatch can be developed to become compliant with the Continua standards once all standards are completed by the Continua Health Alliance and underlying layers, e.g. the ZigBee health care profile for the CC2530 microprocessor, are available from vendors.

The Solianis non-invasive continuous glucose monitor has also been described. This device is integrated into the REACTION platform similar to other sensor devices by the Continua standards. The Solianis sensor device has two parts a sensor and a laptop. The interface to the REACTION platform is through the laptop.

This deliverable does not described issues related to legislation on medical devices e.g. safety and biocompatibility. These issues are addressed in the deliverables for each of the specific sensor devices: D3.2.1 “First Generation ePatch”, D3.3 “Solianis CGM: System Description”, and D3.5 “Prototype of an IR-spectroscopy based CGM Sensor”. Or, will be fulfilled for CE certified devices included in the REACTION platform e.g. the Roche Accu-Check glucose monitor.

7. Glossary of terms

AHD	Application hosting device
BAN	Body area networks
CGM	Continuous glucose management
CVS	Concurrent versions system
ECG	Electrocardiography
IEEE	Institute of Electrical and Electronics Engineers
ISM	Industrial, medical and scientific
LED	Light emitting diode
MDD	Medical device directive
MGMS	Multi-sensor glucose monitoring systems
PAN	Personal area network
PCB	Printed circuit board
REACTION	Remote accessibility to diabetes management and therapy in operational healthcare networks.
RF	Radio frequency
SCI	Software configuration items
SOA	Service Oriented Architecture
SOAP	Simple and object access protocol
SpO_2	Oxygen saturation measured by pulse oximetry
SVN	Subversion
UDDI	Universal description discovery and integration
WDSL	Web service definition language
WS	Web service