ICT Policy Support Programme
Call 3 objective 1.3 ICT for ageing well / independent living

Grant Agreement No. 250505

inCASA

Integrated Network for Completely Assisted Senior citizen's Autonomy

D4.3 Advanced Monitoring System Implementation

Stefan Asanin CNET, Jordi Rovira Simón TID,
Dirk Lill, Axel Sikora SIG, Paola Dal Zovo, Cristina Barbero, Liudmila Mukhina, Andrea Prestileo, Vincenzo Cassani REPLY, Georgios Lamprinakos & Kostas Papadopoulos NTUA

Project start date: 1st April 2010

Duration: 30 months

Published by the inCASA Consortium
Coordinating Partner: SANTER REPLY Spa

15-07-2011 – version0.9

Project co-funded by the European Commission
within the CIP ICT-PSP Programme

Dissemination Level: Public

Document file: D4-3_Advanced_Monitoring_System_Implementation_v1.0.docx

Work package: WP4

Tasks: Task 4.3, Task 4.4 & Task 4.5

Document responsible: CNet Svenska AB

Document history:

Version	Author(s)	Date	Changes made
Draft01	Stefan Asanin	17/5 2011	Table of contents
Draft02	Stefan Asanin	8/6 2011	Refined ToCs
Draft03	Stefan Asanin	16/6 2011	Refined ToCs II
Draft04	Stefan Asanin	17/6 2011	Refined ToCs III
Draft05	Jordi Rovira Simón	21/6 2011	Chapters 4.2, 5 & 8.1
Draft06	Dirk Lill, Axel Sikora	27/6 2011	Chapter 4.1
Draft07	Paola Dal Zovo, Cristina Barbero, Liudmila Mukhina, Andrea Prestileo, Vincenzo Cassani	28/6 2011	Chapters 4.3, 4.4, 6.1, 7.1, 7.2, 8.2-8.4, 10.1 & 10.2
Draft08	Georgios Lamprinakos, Kostas Papadopoulos	30/6 2011	Chapters 5.1, 5.2, 8.5 & 9
Draft09	Stefan Asanin	14/7 2011	Chapters 1, 2, 3, 4, 6, 6.2, 7, 8, 10, 11, 12 & Executive Summary
Draft10	Stefan Asanin	15/7 2011	Final version

Peer review history:

Reviewed by	Date	Comments
Jordi Rovira Simón, TID	15/7 2011	Some typos have been corrected

Index

List of Figures

List of Tables

Executive summary

This document represents the Advanced Monitoring System implementation for the inCASA project. The content of that system comprises the inCASA SPP module and the relevant subsystems (i.e. SARA Services, Semantic Reasoner and other proprietary solutions). The document also describes for future implementations a dynamic integration approach, subsystem features and communication in order to give full insight in how the Advanced Monitoring System implementation is most properly conducted for inCASA. Here also the feature of the inCASA Device Connectivity Kit is presented as optional implementation by any specific Consumer application or other external developer. A development and test plan is continued from D4.1 and D4.2 that continues to iteratively run over the WP4 deliverables, For this deliverable a third more flexible set of test notations are given, analysed and reasoned upon.

1 Introduction

1.1 Background

Deliverable D4.3 is the third and last in order for WP4. This Work Package (WP) continues on the previous deliverables' work on fulfilling the EU e-inclusion targets[1]. That is, a major objective of EU policies and actions is the provision of access, accessibility and user friendliness of devices and services. These are prerequisites for the inclusive delivery of advanced services for the ageing society. Mainstream ICT products and services rarely address the needs of the older population, e.g. those related to the multiple progressive impairments associated with age. Markets tend to overlook the needs of older users: there are few guidelines, voluntary or mandatory standards and related regulatory frameworks. Toward these directions inCASA project is focused on the user-friendliness of devices and services designed and implemented in the project context, while special attention will be given to accessibility even for the mentally ill or elderly impaired. This will be reached by the project objectives, which are by realising and testing in specific pilots efficient integrated care systems that combine innovative technological platforms for ubiquitous communication, advanced healthcare monitoring and state of the art domotic systems. Here, and for the sake of this deliverable, an additional focus will be given on the intelligent features of the inCASA platform, i.e. the Advanced Monitoring System implementation.

The inCASA solution collects data unobtrusively and non-invasively on behaviour, using wireless detection of movement. The architecture combines multiple sensors (like PHS/BSN) and is also able to integrate further data to increase profiling accuracy and achieve the medical target. As a result from this, inCASA could prepare the launch of a venture product by which a major advantage would be related to greater financial sustainability of the project in the long-run. Conclusively this means that a centralized European start-up creates market attraction and visibility for inCASA as a provider of advanced ICT solutions. In order to achieve this, the project consortium make use of its proposed success of deliverables D4.1 and D4.2 where the inCASA architecture combines multiple type of sensors and base their potentiation by integrating further input and/or resource data. This increases profiling accuracy and achieves the medical target of the project over a scalable P2P inCASA network open for extensions. Then as the "*icing on the cake*" is here sanctioned by the advanced monitoring mechanisms described as an implementation approach throughout this deliverable.

1.2 Purpose and Content of this Deliverable

This is the final document on WP4's work on presenting the inCASA solution implementation where it describes the iterative implementation of all the parts of the inCASA system as well as the technical testing of its functionality. Even in this deliverable, the iterative processes of inCASA involve a development plan that describes how the different software modules, subsystems and systems developed in inCASA will be integrated in order to easily interoperate while the test plan will be the actual basis for the testing process. D4.3 will formalise an extension of the set of actions introduced in deliverable D4.1 and continued in D4.2 but also include the procedures, processes, equipment, materials, activities or systems relevant for D4.3. These sets will be further revised in order to help to understand whether the system performances meet both old and refined required specifications and quality attributes set in WP2.

The results will be validated according to certain validation criteria and needs for adjustments of the project user requirements specifications and design specifications will be determined by referring to previous deliverable's approach. The work is divided into the following implementation tasks (those estimated to be directly or in-directly relevant for D4.3 is presented in *italic* and with

[1] http://ec.europa.eu/information_society/activities/einclusion/index_en.htm

extended description in chapter 2.1). Each task focus on a specific component in the inCASA framework.

- Task 4.1 – Hydra Customization and Sensor network Set up
- Task 4.2 - Remote Monitoring gateway implementation
- *Task 4.3 - EPR and Application Customization*
- *Task 4.4 - Reasoning and Learning System*
- *Task 4.5 – Service Delivery Platform*
- Task 4.6 - Telehealth Applications

The purpose of this deliverable is to describe the iterative implementation of inCASA's Advanced Monitoring System. It will mainly concern the last steps where sensor measurements are processed through the basic services layer and intelligent layer and propagated as services in the final added value services layer (Figure 1). The implementation of the inCASA Advanced Monitoring System will also reinforce those aspects that were stated in D4.1 as well as in D4.2 saying that Hydra Middleware modifications were necessary in order to provide sufficient adoptability for the Remote Monitoring device functionalities. Though, these aspects concern here the service availability through the inCASA Data Collection and server side and the intelligence mechanisms in providing an advanced backbone for inCASA Consumer Applications and external resources. The purpose of this deliverable is also to present the third iteration on development and testing plans which may be further modified to fit the technological scopes and requirements set by tasks in deliverables D4.1 and D4.2 including WP3, WP5 and WP6.

1.3 Outline of this Deliverable

This document will establish a reasonable approach to the development activities, the expected prerequisites and issues regarding the Advanced Monitoring system implementation and uses are related to the complementary WP4 tasks, in Chapter 2. Chapter 3 continues the iterative work where both D4.1 and D4.2 implied that the renewal of the WP4 development and test plans possibly involves a new set of actions for the remote monitoring subsystems and components relevant for this deliverable. Chapter 4 details data collection per inCASA subsystem and how to retrieve these data. Chapter 5 specifies a socio-medical calendar while chapter 6 reviews how semantic could become useful in order to share its underlying data with person identification (chapter 7). Chapter 8 actually describes the entire point of this deliverable by the customization of inCASA Advanced Monitoring System. Chapter 9 presents different GUI approaches to fit regional demands (e.g. language). The continuation of Advanced Monitoring System is performed in chapter 10 presenting the inCASA business logic and habits modelling. Chapter 11 goes through the experiences and lessons learned in the test chapter of this deliverable. Finally, chapter 12 provides the reader with the final conclusion and validation for the WP4 deliverables.

2 Description of the Advanced Monitoring System

Generally elderly people have very fixed behaviour. They might wake at the same time every day, eat at the same time, and visit the bathroom with a regular pattern. Such behaviour, when monitored for a sufficient time, the inCASA platform can provide a detailed profile of the personal behaviour. This can be the starting point for an extended “Electronic Patient Record” (EPR), that might become an intelligent container for Social and Health Care (a “Smart Personal Platform”), combining personal data, social and healthcare data, together with action plans set by social workers and healthcare professionals. This information, managed following privacy and data security policies, can provide the normal baseline from which the settings for alerts can be defined and interactions with domotic tuned. Physiological monitoring devices can improve the quality of the profile from the clinical perspective, providing important data to the Healthcare professional.

The final step of the inCASA solution implementation is the Advanced Monitoring System. This will complete the platform as a citizen-centric technology by which services network help and protect frail elderly people and prolongs the time they can live well in their own home. The implementation of Advanced Monitoring System will be achieved by integrating from D4.1 and D4.2 solutions/services for health/environment monitoring that collects and analyses data in order to profile user behaviour, implement customized intelligent multilevel alerts/communication services. Data will be made available to care services through the Smart Personal Platform (SPP) with an embedded Behaviour Analysis Application which will include: access policies to preserve privacy; planning for day-by-day activities and therapies with multiple alerts; co-ordination of local public Social and Health Care Services; and help to deploy specialist community based services.

As a support to the Advanced Monitoring System, the adoption of distributed intelligence in home management and monitoring will provide both a safer environment and help daily living activities while certifying the service provider with sufficient environmental implications.

2.1 Task Related Work

Throughout all of the WP4 deliverable there is a set of work tasks that roughly relate to each deliverable’s content and intention. For D4.3 the major contributions that can be derived from the tasks are Task 4.3, Task 4.4 and Task 4.5. Here comes first a high-level relation to the implementation work of D4.3 and then a task specific description according to:

- Customize the different EPRs and applications (Tasks 4.3 & 4.5).
  - Collect and store structured data and documents about socio-health status.
  - Enhance and manage vocabularies and Person ID (PIDs) from different sources.
- Enhance business logic and ensure module functionalities (Task 4.4).
  - Create a behavioural model, compare data, perform logical reasoning, provide user support, and more.

Task 4.3: EPR and Application Customization; deals with the customization of the different parts of the present EPR system in order to monitor and manage elderly population included in the pilot and the overall inCASA platform by collecting and storing both structured data according to the elderly population PIDs while enhancing socio-medical semantics for information sharing.

Task 4.4: Reasoning and Learning System; will ensure the enhancement of different modules of business logic, ensuring that each module is able to successfully play its role within the inCASA platform. This involves modelling elderly person’s routine (habits model) by comparing both stored data and data received from the base station and perform logical reasoning based on clinical path, pathological values parameters, and stored rules together with the routines from the elderly persons.

Task 4.5: Service Delivery Platform; this task will cover the implementation of the operating central in term of remote healthcare provider platform services finalization, the work for D4.3 will be based on the extension for the service exploitation defined in both D4.1 and D4.2 [1, 2].

2.2 Draft Advanced Monitoring Architecture

The draft implementation models described for the Core Monitoring System in D4.1 and the Remote Monitoring Device implementation in D4.2 rely on the original abstract view the consortium had on developing the inCASA platform but where the implementation description in the deliverable highlighted these deliverables underlying mechanisms. In a similar way as done in D4.1 and D4.2, is also done for D4.3 where the advanced mechanisms and service intelligence are given focus.

The final step in the implementation model (Figure 1) for Advanced Monitoring System aim to present the involved SMEs and academic institutes by their innovative approaches and resources of platform subsystems and components designated for the inCASA Advanced Monitoring System. It will also try to lead the way for the second implementation iteration of D4.1 and D4.2 but also model how inCASA extension developers will interpret the platform implementation.

While the D4.1 Core Monitoring System architecture demonstrated how to implement middleware mechanisms for the inCASA gateway and server side and D4.2 Remote Monitoring Devices showed sensors, device and appliances together with their underlying processes, Figure 1 points at the current work of implementation for this deliverable, i.e. the Advanced Monitoring System with the SARA Services, Semantic Reasoner encapsulated within the SPP and for inCASA Consumer Applications as well as external developers.

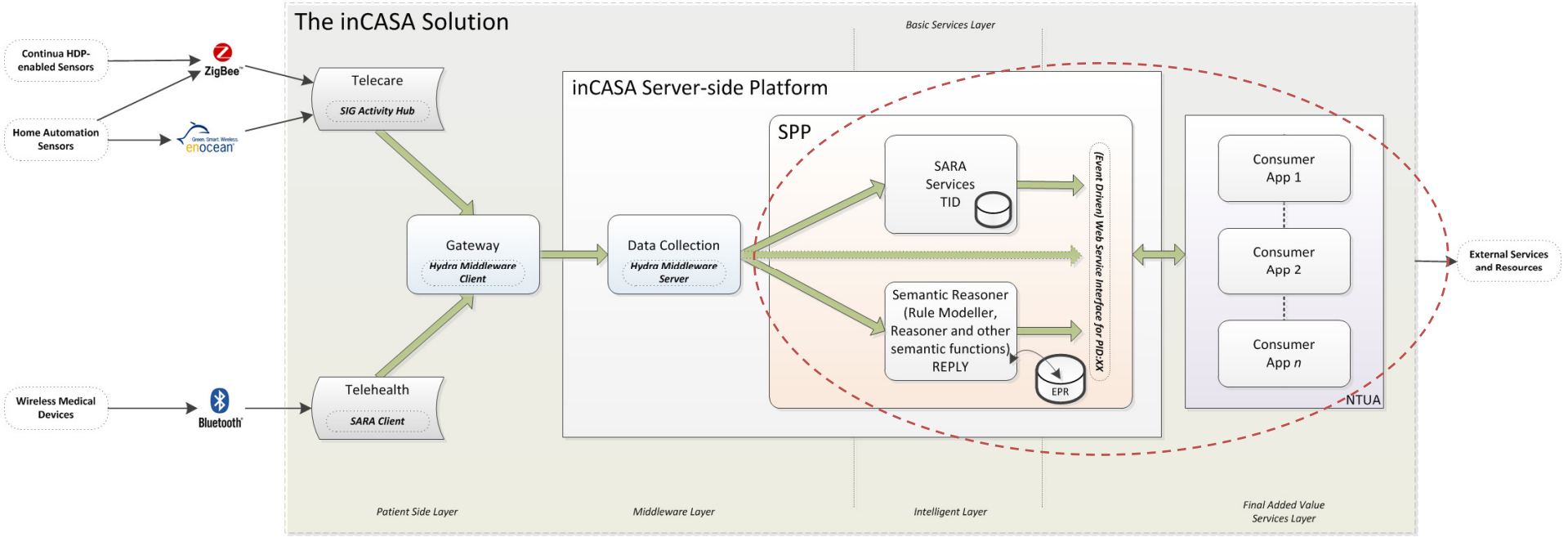

Figure 1 - Current draft of Advanced Monitoring System implementation where subsystem specific consortium partners involved are marked.

3 Project Development and Test Plans

Already known in D4.1 and D4.2 is that WP4 does not only deal with the iterative implementation of the inCASA system parts but also with the technical testing of subsystem functionalities in order to ensure that these are monitored according to a quality management procedure. The third iteration of the implementation and test planning process is taking place in D4.3. The development plan will describe how the different software services provided by the two subsystems (i.e. SARA Service and Semantic Reasoner) become relevant for the Advanced Monitoring System in the inCASA project. The document will describe the integration enabling these subsystems to easily interoperate with the inCASA SPP module at server side. Also similar to the approach in D4.1 and D4.2, the test plan will become the basis for the testing process and the results will be presented in chapter 11 where any needs for adjustments of the project user requirements specifications and design specifications will be determined in the summary (chapter 12).

The work here is divided into the implementation tasks presented in chapter 3.1. These will each assist in the description of implementing the Advanced Monitoring System by giving focus on two specific subsystems (i.e. software services) in the inCASA framework.

3.1 Quality attributes and set of actions for subsystems and components

Implementing an iterative methodology the inCASA work plan ensures a tight cohesion and continuous communication between all the different phases of the process so as to minimize the risks discussed in chapter 11.3. Testing the D4.3 implementation in a new environment (e.g. pre-pilots) will increase the customization that will be required to meet local requirements (e.g. pilots) by updating a set of quality attributes and set of actions for subsystems and components available for D4.1 and D4.2 refinements. These set of actions should involve any procedure, process, equipment, material, activity or system that will help the consortium partners to understand whether the system performances do meet the required specifications and quality attributes. These set of actions are deliverable specific, i.e. they relate to the matter of topic (i.e. here the implementation of the Advanced Monitoring System).

The Advanced Monitoring System implementation involves the quaternary communication between the inCASA SPP utility, service providers and other external actors. It affects all of the communication levels all throughout the inCASA architecture (D3.2) and enables a wide adoption of the prominent services found in the inCASA platform.

The Advanced Monitoring System implementation should preferably have the following quality attributes that can be reclaimed through D3.2 Reference Architecture Iteration 1 [3], D4.1 as well as D4.2 and matched against the cohesion of D4.3. These are transparent throughout of this deliverable and all contribute to fulfil the stated quality attributes above and are further anticipated to reveal separate subsystem's needs of counteractive actions to be set. A small sample of the quality attributes are the following:

1. Interfaces must be designed in order to allow information security at both physical and logical level,
2. User interfaces must provide different access level, both to data and functionality, according to user's profile and role within the system,
3. GUIs, in terms of presentation of data, should be separated from system logic.

3.2 Deploying an iterative process for the plans

The actual executions of subsystem and component actions are performed by an iterative process throughout the WP4 deliverables. D4.1 started off and D4.2 continued including arguments posed in WP3 deliverables and especially the D3.2. The submission of D4.3 contain additional outcomes of the subsystem and component validation that conduced altered quality attributes to accomplish

as well as a register of set of actions to be executed in forthcoming WP4 deliverable. As such, the WP4 implementation declares an iterative process for the parted but still collective development and test plans which will be separately and implicitly tested throughout the WP6 and WP5 as to generate reformed development plans.

In parallel throughout the duration of the third cycle, the architecture design will be refined, following pilot evaluation and monitoring where the final step involves use case activities and deployment plans. The data collected from the pilot monitoring, technical, objective and subjective data will be analysed and reported into a final evaluation of the applicability of inCASA solution. Any changes of devices, sensors, development kits or any subsidiary technologies for inCASA are to be found in WP5 or WP6 documents.

4 How to collect, store and retrieve monitored data

The previous two deliverables within WP4 has stated that the basic step of the platform is the data collection, via BlueTooth/Wi-Fi/ZigBee, from the environment monitoring system (wireless motion/contact sensors to monitor the senior in their own home and smoke detector, natural gas detector, flooding sensors/water electro valves, temperature in case of pre-existing domotic systems) and pre-existing human monitoring system. As such, we know that there will be data collection at base station level as described for the Hydra Middleware in chapter 5.2 in D4.2 but also collection performed for data at sensor and device level, i.e. Activity Hub and SARA Client.

Except the key functionality of the platform to synchronize the base station and the service providers and thereby allowing for the development of the behaviour model that is the core feature of the platform, the data collected by the base station also need to incorporate the preceding subsystems' features to collect and store data. This approach may improve clinical information to the emergency teams as well as strengthen the analysis of collected data to create and continuously improve users' behaviour models as well as better identify risks for the user and emergency needs.

4.1 Activity Hub storage

The communication to the backend is based on events, which means that only values that excess the given limits are transmitted to the backend. This implies that the models are known and download to the Activity Hubs before using them. For model development, this approach is not very suitable. Therefore, the raw data are stored in 512kB internal RAM before writing them to a microSD card of maximum 8GB. They are sent to the backend on request. As those microSD cards may be accessed, all data are AES128 encrypted before writing them to the storage device. To reduce the risk of manipulation, the encryption key is only known to the Activity Hub, and the data are decrypted again before transmitting the data to the backend. This encryption is supported by the micro controller's Cryptographic Acceleration Unit (CAU). The amount of data to store depends on the use case and on the sensor settings. As those settings can be modified through the radio interface, the sensor behaviour can be optimised during the project.

Theoretically, an 8GB memory device could store more than 500 million sensor values, and if they are not requested by the backend, they can be overwritten. The storage will also be used for event messages in case of missing backend connectivity.

4.2 SARA Client storage

The SARA client acts as gateway to make the bridge between sensors and the inCASA platform. However, there are certain types of measures that might require an average calculation (e.g. blood pressure values if taken too every second) or further processing before being sent. In those cases, an array of measures is stored within the SARA client to perform a data transformation or process it for applying inferring techniques.

4.3 Structured data about the socio-health status of elderly (SCDR)

In order to accomplish the Continua Alliance's guidelines the SPP implements the IHE-PCD transactions (PCD-01, PCD-02, PCD-04) both for communication with other external systems (Hydra, CA) and between its internal blocks.

Measurements and alerts are exchanged using the HL7 ORU^R01^ORU_R01 message; the data collected inside the patient's household are transmitted by Hydra to the SPP, which stores them inside the Socio-Clinical Data Repository realized by the SPP EPR.

The EPR maintains the HL7 messages received and processes them associating the sensors' data to the right person and location and extracting pieces of information useful for successive inquiry.

Alerts are managed keeping track of the alarm life cycle through the collection of all the messages related to its status change.

To retrieve data from the EPR an appropriate HL7 query message (QSB^Z02^QSB_Q16) must be issued; the result is returned formatted as a HL7 ORU^R01^ORU_R01 message. The HL7 message returned may coincide with one previously stored or, more often, can result from the aggregation of many measurements, can be composed by a list of alerts and so on; depending on the query it may relate to one or many patients and/or locations.
The result of the SPP Reasoner elaboration, such as the creation and enrichment of the person's habits model, is converted into a HL7 ORU^R01^ORU_R01 message as well and stored inside the EPR.

The SPP Mediator is responsible for the reception of the HL7 messages from outside the SPP and their routing to the due destination. For the measurements flow the HL7 messages are routed first to the EPR for storing and then to the Reasoner for elaboration. For the alerts flow the HL7 messages are sent in parallel to the CA for showing on the UI and to the EPR for storing.

Measurement and alert data flows are shown in Figure 1.

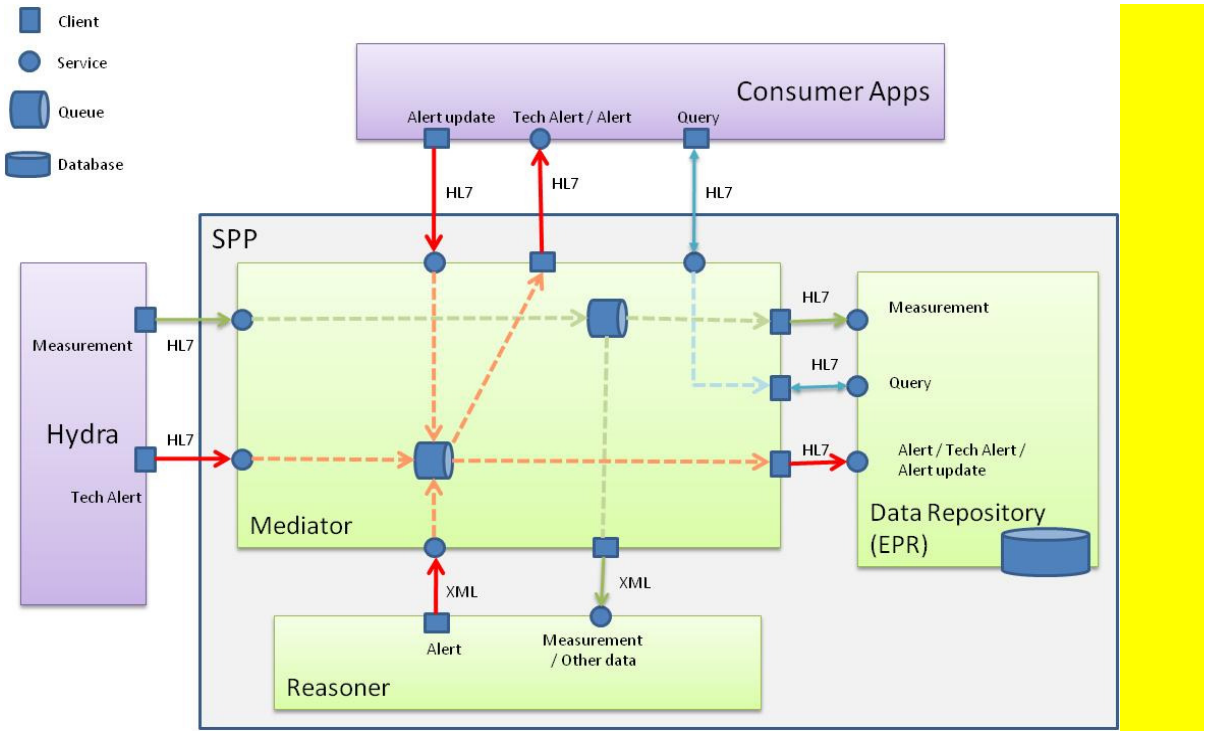

Figure 1 - Measurement and Alerts flow diagram

The SPP Reasoner, for its internal use, handles data in XML and in W3C Semantic Web languages (OWL).

All the communication between the modules making up the SPP (Mediator, Reasoner and EPR) and between the SPP and the external systems (CA and Hydra) is done using web services.

The Mediator exposes (at least) the following Web Services (WS):

- 1 WS in charge to receive measures from Hydra.
- 1 WS in charge to receive technical alert from Hydra.
- 1 WS in charge to receive queries from CA.
- 1 WS in charge to receive alert updates from CA.

- 1 WS in charge to receive parameter updates from CA.
- 1 WS in charge to receive alerts raised by the Reasoner.
- 1 WS in charge to receive data generated by the Reasoner and that need to be stored in the EPR.

In particular, the communication between the Mediator and the Reasoner is done using a subscription/notification mechanism on a dual binding channel: in this way each variation of a service exposed by the Mediator is automatically propagated to the Reasoner.

4.4 Periodic Data Retrieval

The frequency of data retrieval for the correct functioning of the Telecare monitoring services depends on the scenarios and on the reasoning they involve. To plan the frequency of the data retrieval from the sensors different scenarios were analysed.

As environmental parameters such as temperature and humidity don't change very quickly, the related scenarios don't require frequent retrieval of the data. One in an hour should be more than enough to signal abnormal situations and to have an overview of the temperature/humidity changes tendency in the apartments. For other scenarios instead the data should be retrieved much more frequently.

The indoor movement scenario would benefit in theory from analysing every movement of the person, but taking into account resource constraints, we deem that getting the movement every 30 seconds (if any) should be enough for the type of required monitoring.

Besides sensor that measure parameters and transmit data periodically, inCASA system includes sensors that transmit information only if an action occurred or a status changed.

These are sensors like the contact sensor for the door and the fridge, the bed/chair usage detector, the device usage detector, the gas/CO/water flood detector. The door sensor should send the data as soon as an interesting event occurs, like door open event/door closed event. The rest of the time no information are necessary.

The Gas/CO2 sensors need to send data only when the gas/CO2 level overrides the allowed limit, this will launch the technical alert.

5 Socio-medical Calendar Specification

An objective of the inCASA healthcare services provision model is the reduction of the in-hospitalization rate of patients. The remote monitoring capabilities of the platform ensure that healthcare professionals are notified in time when the patient's medical measurements deviate from normalcy, allowing them to intervene and avert a possible deterioration of the patient's health. The response of the doctors may escalate from forwarding a request for change/adjustment of the prescribed medication to arranging an appointment with the patient, for further medical examinations.

For the realization of both options the platform must be complemented with scheduling capabilities. From a functional viewpoint the main platform users must be allowed to program their activities and follow their defined schedule:

1. The doctor prescribes medication to the patient that is usually received in accordance to a regular (cyclical) schedule. The compliance to the prescribed therapy is enhanced when reminders are sent to the patient in accordance to this schedule.
2. On the other hand the doctor's have a limited time during their workday, to examine patients and offer their medical services. Their constrained schedule should be adjustable to accommodate for emergencies and should allow the prioritization of specific cases.

The Socio-medical Calendar [4] will be an easy-to-use scheduler for inCASA operators which will support operators to plan for activities within inCASA services, to set reminders for the elderly users and on which take notes on actions and on which record all the remarkable.

This module will be a scheduling rules package for a single operator as well as for a team. It will help to create an organized schedule for the whole staff and to plan for reminders to be sent to the users or surrounding people (relative/neighbour/nurse) through e-mail, SMS or phone call.

5.1 GUI Representation and Format Abstractions

Each operator will create a profile to run his personal schedule allowing the viewing of multiple profiles at a time, to view the schedules of several or even all operators of the team simultaneously.

It will allow jumping to the required date, editing appointments, managing patient data, planning for a reminder to be sent to single user's home gateway (where an interactive module is deployed) and record actions and related notes.

All information will be accessed through a GUI by a single user sign-on and linked to the "calendar entry" just for the session. The scheduler complete functionality will be exposed through the Consumer Applications front-end, possibly linked with a summary of the patients' health status screen, to allow doctors to make informed decisions on prioritizing appointments and scheduling them (i.e. for further medical examinations). Additionally, the GUI front-end will also allow doctors to make adjustments on the prescribed medication of patients (i.e. frequency and dosology).

Personal data will be stored in Service Provider's infrastructure repository, secured with an encryption algorithm and "single access key" protected preventing from unauthorized access. The recorded information in the Socio-medical calendar archive will be organized and contextualized to past events/actions/actors in the following fashion:

- Calendar event descriptive short title.
- Calendar event unique ID (Automatically generated by the scheduler module).
- Calendar event modality (Cyclical vs. Atomic).

- Event type/category (i.e. Appointment).
- Patient unique ID based on EHIC – European Health Insurance Card.
- Patient name.
- Operator user ID (as created for the Single Sign On module).
- Operator name.
- Operator user group (i.e. Doctor, social worker).
- Event planned date (i.e. Day, time).
- Place that event will take place (i.e. out-patient clinic).
- Contextual information.
    - Trigger Event (i.e. a patient's measurement that is outside normalcy), if applicable.
    - Previous reminders for calendar event, if applicable.
    - Activities or Interventions that already took place to respond to trigger event.
- Any other related information (side notes).

The secured archive will be auto backup function enabled. The archive will allow user to extract data from for example a HL7 based structure and export it to a variety of formats, including MS Outlook, XML, HTML, XLS, TXT, CSV.

5.2 Appointment functions

In order to facilitate the second option, the platform must be complemented with scheduling capabilities to allow doctors to arrange and accommodate appointments with patients and notify the users or their surrounding people accordingly, through reminders delivered to the patient tablet or pc, SMS messaging or even phone calls.

6 The concept of semantic enhancing socio-medical data and information sharing

As an outcome of on the European healthcare market, the consortium draws hope that the inCASA model based ontology concept will allow for rapid development and deployment of innovative telemedicine and eHealth applications. It is of particular interest to innovative SMEs where containment of development cost is critical and where both inCASA and external SMEs can interoperate with the inCASA platform in order to deliver specific healthcare value added services.

6.1 Semantic Reasoner (REPLY)

The SPP reasoning module uses W3C Semantic Web standard for modelling data. The semantic representation used by the Reasoner is ontology where each class is an “Entity”. The main types of entities are:

- PhysicalEntity: describes all objects that are materially detectable, such as devices, people, and areas.
- ReasoningEntity: describes reasoning modules and context awareness services, and entities used for modelling the context awareness module.
- VirtualEntity: describes entities that are not physically detectable but that may contribute to context recognition or decision making. An example is a weather forecast web service.
- MeasuringEntity: describes measures such as sensed data provided by sensors.

The knowledge base is structured in a base ontology with high level classes and different ontology (sub-) models for specific device/service classes, so that it can be dynamically managed and easily extended.

Starting from the basic ontology structure already available in the Reasoner prototype, new models (and new classes) are being added for inCASA project.

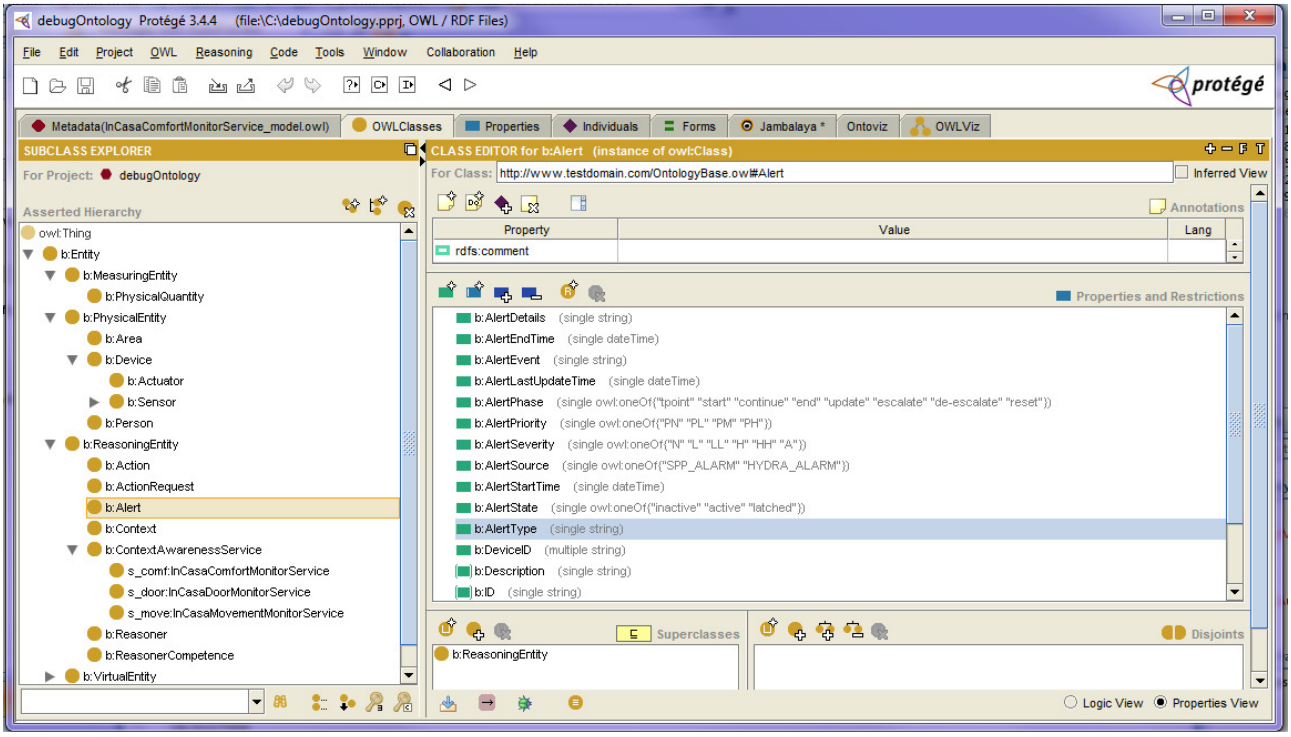

Figure 2 - SPP Ontology (as viewed in Protégé open source tool).

In particular there is a separate OWL model for each type of device, one for each type of context awareness service, one for each conceptual entity used by the Reasoner (e.g. user habits, alerts). In particular, the ontology provides a representation of context awareness services in charge of monitoring specific situations and their relation with devices, apartment, and monitored user.

On the previous page, Figure 2 shows a view of the SPP ontology used by the semantic Reasoner, as it can see in Protégé ontology editing and browsing tool (open source, not developed in inCASA). In particular the Class Editor frame on the right shows the definition of the properties of the Alert class.

Here, the Figure 3 shows the ontology representation of a Telecare context awareness service added in inCASA, and its relations with sensor it uses and possible alarms.

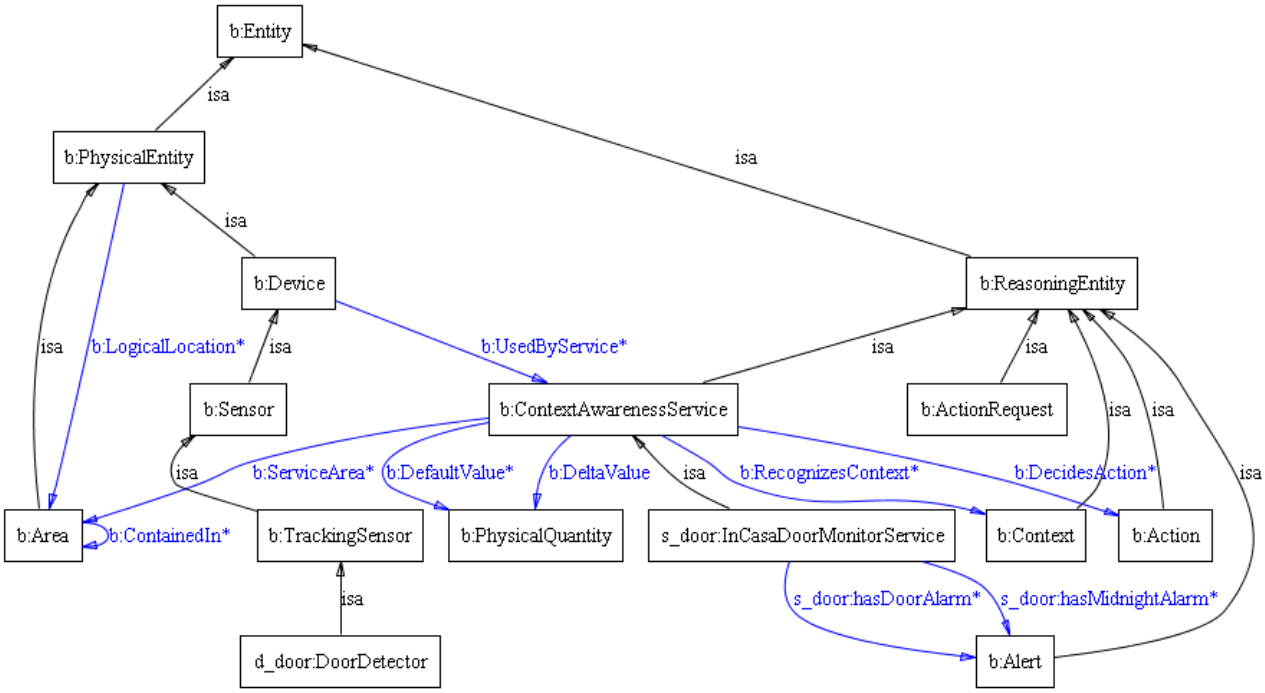

Figure 3 - Excerpt of ontology representation.

6.2 Hydra Device Ontology

Described in D3.1 and D3.2, Hydra's incorporation of a Semantic Model Driven Architecture allows for an enriched interconnection of devices, people, terminals, buildings, etc. but that also with its SOA approach provides interoperability at a syntactic level [3, 4]. However, the Hydra middleware provides the interoperability at a semantic level by extending semantic web services to the device level and thereby open up for semantic interoperability of inCASA AmI applications. In Hydra there is an Ontology Manager that stores all meta-information and knowledge about devices and device types. It provides an interface for using the Device Ontology that contains other models classifying what device and services are currently running in the inCASA base station domain. It describes the devices hardware/software capabilities, services, state-machines and security capabilities.

7 Managing Unified Person Identification

As part of the work done in D4.3 the customization of EPR and inCASA applications implicitly also involve the management of person unified identification and the reconciliation of records from different systems. It also concerns the relation to the behaviour profiling that is not only fixed in each pilot but come as a result from using various technologies and devices.

Another issue is the traceability of people throughout the inCASA network. If the movement of a person can especially be tracked through public networks, privacy cannot be guaranteed. To solve this problem, for the public communication, the specific ID of the person is to be kept secret, and communication should only be based on addresses that are randomly granted. Though, the current addresses may be publicly visible easing the message routing even through public networks and the inCASA platform. For this reason, the SPP must be able to correlate healthcare and Telecare data collected in the household to the person they are related to.

7.1 Handling Personal Records from different platform domains

During the installation phase of the inCASA blocks inside the base-station the same Person ID will be assigned to the elderly patient by both the Telecare and Telehealth systems. This same Person ID will be configured inside the EPR together with the other patient identifying data.

This approach nullifies the need of a process of reconciliation of the records belonging to one patient but coming from different systems.

Another benefit of adopting this solution is that Telecare and Telehealth data can be transmitted from the base-station to the remote provider platform without having associated any personal data of the patient (family name, given name...); this information would have been mandatory to reconcile measurements having different Person IDs.

7.2 ORU^R01 messages

HL7 messages are used by Hydra, SPP and CA to exchange measurements and alerts.
The PID segment of the HL7 messages is devoted to the transport of patient identification information. The PID-3 field is the Patient Identifier List and its data type is CX (extended composite ID with check digit); being a repeatable field the Identifier Type Code (CX-5, alias PID-3-5) sub-field is used to discriminate the kind of data contained inside each repetition. The Assigning Authority sub-field (CX-4, alias PID-3-4) is then used to identify the authority responsible for the patient identifier assignment.

In the HL7 messages exchanged inside the inCASA project the PID-3 field will be valued as follow:

- PID-3-1 = unique Patient Identifier
- PID-3-4 = inCASA
- PID-3-5 = PI (Patient Internal Identifier)

7.2.1 Unique PIDs

The inCASA solution may adopt as Unique Person Identifier the personal identification number taken from the European Health Insurance Card (EHIC), owned by every European citizen covered by a social security scheme.

This choice avoids the need to assign a proprietary person identifier to the users of the inCASA system and allows a simpler configuration, since the information is easily retrievable from the patient itself. The EHIC personal identification number is a unique alphanumeric string 20 characters long.

8 Customizing the Advanced Monitoring System

In order to make the integration of the basic services layer and the intelligent layer functional with the inCASA Data Collection (i.e. Hydra Middleware server side) this chapter will describe the implementation of each of the Advanced Monitoring Systems, i.e. SARA Services and Semantic Reasoner within the inCASA SPP.

Following this description is should fairly easy to map how any future extending inCASA solution type fulfilling a specific proprietary value added service could be implemented.

8.1 SARA Services (TID)

The SARA Services are specific health care services for the medical portal consumer app implemented in inCASA. These services are accessible thanks to Hydra tunnelling that integrates different subsystems both in the client side (Activity Hub + SARA Client) and the platform side (SARA Services + SPP).

The main idea is that the SARA services provide a low level API so more skilful developers can make medical applications using “raw” information coming from the sensors, while the SPP provides an API that involves a processing of this basic information to get an intelligent output (i.e. semantic reasoned) for developer.

The Services exposed both for the sensors and the final consumer app that has been implemented follows the next **API**:

- **Kit Service:** This service is used to make the association between the patient and the Sara client, also called gateway. These are its operations:

Concept	Description
storeKit	Create a new kit on the database associated with an user
removeByKit	Delete elements from the database associated with a kit;the kit may also be deleted.
modifyKit	Modify information about a kit.
getKits	Retrieve kits that fit the parameters given.
addDevicetoKit	Associate an already existing device with a kit.

When creating a new kit, it is necessary to indicate the following parameters:

- Ap: parameters needed to authenticate within the project (userWS, passWS and id).
- IdUser: id of the user the kit will be associated with.
- IdGateway: id of the gateway associated with the kit.
- Name: name of the kit.
- userWS: name used to access the kit.
- pwdWS: password used to access the kit.
- Status: not used.
- Type: not used.

This is the SOAP message that creates a kit (i.e. operation storeKit):

```
<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:impl="http://impl.webservice.bio.gtssd.ehealth.tid.es"
xmlns:biow="bioWebService_Kits">
   <soapenv:Header/>
```

```
   <soapenv:Body>
      <impl:storeKit>
         <impl:ap>
            <biow:id>1</biow:id>
            <biow:passWS>wstest</biow:passWS>
            <biow:userWS>wstest</biow:userWS>
         </impl:ap>
         <impl:idUser>30</impl:idUser>
         <impl:idGateway>1</impl:idGateway>
         <impl:name>kitPrueba</impl:name>
         <impl:userWS>kitP</impl:userWS>
         <impl:pwdWS>kitP_pwd</impl:pwdWS>
         <impl:status>0</impl:status>
         <impl:type>0</impl:type>
      </impl:storeKit>
   </soapenv:Body>
</soapenv:Envelope>
```

- **Gateway Service:** This service is used to associate a Kit (i.e. patient + gateway) to its devices (i.e. sensors). These are its operations:

Concept	Description
storeGateway	Create a new gateway for a project
deleteGateway	Delete a gateway from a project.
modifyGateway	Modify information about a gateway.
getGateways	Retrieve gateways that fit the parameters given.

When creating a new gateway it is necessary to indicate the following parameters:

- Ap: parameters needed to authenticate within the project (userWS, passWS and id)
- Name: name of the new gateway.
- Type: type of gateway (1: media center, 2: mobile, etc.).
- hardwareId.
- Status: status of the gateway (0: normal, 1:broken, 2: lost).

This is the SOAP message that creates a gateway (i.e. storeGateway):

```
<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:impl="http://impl.webservice.bio.gtssd.ehealth.tid.es"
xmlns:biow="bioWebService_Gateways">
   <soapenv:Header/>
   <soapenv:Body>
      <impl:storeGateway>
         <impl:ap>
            <biow:id>1</biow:id>
            <biow:passWS>wstest</biow:passWS>
            <biow:userWS>wstest</biow:userWS>
         </impl:ap>
         <impl:name>mobile4</impl:name>
         <impl:type>2</impl:type>
         <impl:hardwareId>333333</impl:hardwareId>
         <impl:status>0</impl:status>
      </impl:storeGateway>
   </soapenv:Body>
```

- **Device Service:** This service is used to add an remove sensors for the patient. These are its operations:

Concept	Description
storeDevice	Create a new device for a project
deleteDevice	Delete a device from a project.
modifyDevice	Modify the information about a device.
getDevices	Retrieve information about the devices associated with a project.

When creating a new device it is necessary to indicate the following parameters:

- Ap: parameters needed to authenticate within the project (userWS, passWS and id)
- Name: name of the new device.
- File: name of the file used to parse the device (not used at the moment).
- ContentFile: content of the file that parses the device (not used at the moment).
- Type: type of device:
  - 4: GPS
  - 5: Glucometer
  - 6: Accelerometer
  - 7: Weight Scale
  - 8: Heart rate
  - 9: Oxygen saturation
  - 10: Blood pressure
  - 11: ECG
  - 12: Questionnaire
  - 13: Subcutaneous pump
  - 14: Base-station
- HardwareId: the unique identifier of the device.
- Status: status of the device (0: normal, 1: broken, 2: lost).

This web service returns the id of the new device created. This is the SOAP message that creates a gateway (i.e. storeDevice):

```
<soapenv:Envelope      xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:impl="http://impl.webservice.bio.gtssd.ehealth.tid.es"
xmlns:biow="bioWebService_Devices">
   <soapenv:Header/>
   <soapenv:Body>
      <impl:storeDevice>
         <impl:ap>
            <biow:id>1</biow:id>
            <biow:passWS>wstest</biow:passWS>
            <biow:userWS>wstest</biow:userWS>
         </impl:ap>
         <impl:name>BloodPressure</impl:name>
         <impl:file>bp.xml</impl:file>
         <impl:contentFile></impl:contentFile>
         <impl:type>10</impl:type>
         <impl:hardwareId>3333</impl:hardwareId>
         <impl:status>0</impl:status>
      </impl:storeDevice>
   </soapenv:Body>
</soapenv:Envelope>
```

- **Measure Service:** This is the service that will allow sensors to store data and consumer apps to get data from the sensors.

Concept	Description
getMeasures	Retrieve some measures
getLastMeasures	Gets only the last measures (one for each name) that have the same date.
storeMeasure	Store a single measure.
storeMeasures	Store one or more measures.

The getMeasures method provides all the measures that fit all the requirements established in the request. For each measure we will obtain the following information:

- DateMeasure: when the measure was taken.
- DateReception: when the measure was received.
- DateTransmission: when the measure was transmitted.
- Id: Measure id
- IdDevice: id of the device that took the measure.
- Idkit: id of the kit associated with the measure.
- Type: type of value.
- Value: one value is obtained for each type of value; the value located at the value label is the value of the measure codified into Base64 (as it was stored), it also returns an decoded value in its own label (intvalue label for example).
- Name: name of the measure
- Status: current status of the measure
- Unit: unit used

This is a SOAP message that gets measures (i.e. getMeasures):

```
<soapenv:Envelope      xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:impl="http://impl.webservice.bio.gtssd.ehealth.tid.es"
xmlns:biow="bioWebService_Measures">
   <soapenv:Header/>
   <soapenv:Body>
      <impl:getMeasures>
         <impl:ap>
            <biow:id>1</biow:id>
            <biow:passWS>wstest</biow:passWS>
            <biow:userWS>wstest</biow:userWS>
         </impl:ap>
         <impl:idKit>1</impl:idKit>
         <impl:id/>
         <impl:idDevice>7</impl:idDevice>
         <!--1 or more repetitions:-->
         <impl:names>BloodPressure</impl:names>
         <impl:status>0</impl:status>
         <impl:dateAfter>1292533492751</impl:dateAfter>
         <impl:dateBefore>1292533492753</impl:dateBefore>
      </impl:getMeasures>
   </soapenv:Body>
</soapenv:Envelope>
```

8.2 Semantic Reasoner Services

The semantic Reasoner provides its functionalities by:

- notifying detected alert situation,
- providing contextual derived information to be stored in the EPR for further inquiry. This information includes basic measures, typically indication the duration of a situation (e.g. how long a person remains in the bed) and information computed in more complex ways, typically correlating other data in multiple steps (e.g. habits model).

The semantic Reasoner subscribes to services it is interested to and is notified of changes by the Mediator. It uses services exposed by the Mediator for storage of data in the EPR data repository.

The communication between the Reasoner and the Mediator is based on the XML format. A related XSD schema defines the XML logical models used in communication and allows validation of data.

The XML schema used for this communication defines the entities involved in the system (context awareness services provided by the Reasoner, devices, person) as types derived by a unique type, with a hierarchy aligned to that defined in the semantic Reasoner ontology, for a seamless update of the two representations.

The ReasonerConfig.xml is the file that allows configuring some aspects of the semantic Reasoner (e.g. it could be used to blacklist some types of received data or data related to certain apartments).

Figure 4 shows how a service is defined in the semantic Reasoner, by an OWL model, where some properties indicate files of rules and of queries it uses, other properties indicate which devices it uses; an excerpt of a context-recognition inference rule associated to the services monitoring temperature and humidity is also shown.

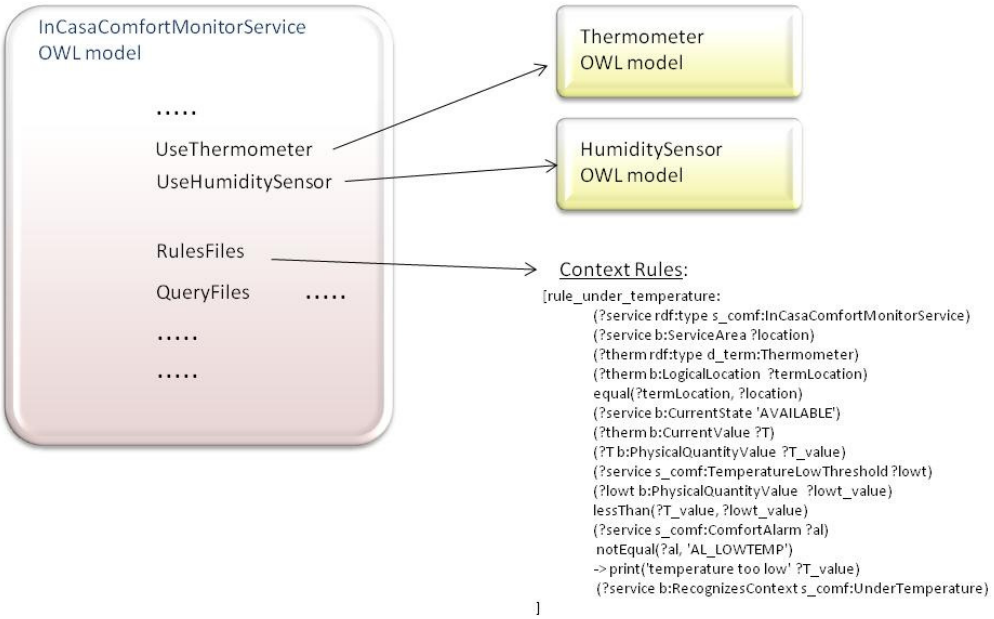

Figure 4 - How a context awareness service is managed in the SPP Reasoner.

8.3 SPP Mediator

The Mediator includes a legacy module for the management of Web Services, used for the communication with the Reasoner; the Mediator can expose Web Services, described with homogeneous XML logical models, through different modalities (dual bindings, REST or SOAP/XML single bindings). It can use different service discovery modalities, based on WS-Discovery protocol or UDDI.

The Mediator handles the interaction with the Reasoner, and notifies any registered client about any data update related to the services exposed.

Besides this communication, it has been enhanced to support point-to-point Web Services communication (based on SOAP/XML) as required in the inCASA SPP and to manage HL7-based communication. The Web Services that the mediator exposes to Hydra and the Consumer Applications are described in Section 8.3.1.

The Mediator is an executable based on the .NET framework and it includes HAPI library for HL7 parsing. Relevant configuration files are the WSConfig.xml, containing configurations related to Web Services, and the LogConfig.xml, related to log settings; both configuration files are under the folder /ConfigFiles.

8.3.1 Services exposed by the Mediator

The SPP Mediator provides communications with modules outside the SPP, namely Hydra and the Consumer Applications. The Mediator exposes SOAP Web Services which conform to the IHE PCD standard definition. There are 5 Web Services, all having the same schema structure, dedicated to different flows of information towards the CA and Hydra.

- **Measurements**

The following Web Service must be invoked by Hydra to provide devices' measurements which will be first queued and then sent to the EPR for storage, and, in case of 'ACK' received from the EPR, to the Reasoner too: http://<host>:<port>/PCDData

The correspondent description file (WSDL) is available at the link:
http://<host>:<port>/PCDData?wsdl

```
<?xml version="1.0" encoding="utf-8"?>
<wsdl:definitions name="DeviceObservationConsumer_host"
targetNamespace="http://tempuri.org/">
  <wsp:Policy
wsu:Id="WSHttpBinding_IDeviceObservationConsumer_Binding_Soap12_policy">
    <wsp:ExactlyOne>
      <wsp:All>
        <wsam:Addressing>
          <wsp:Policy>
            <wsam:AnonymousResponses/>
          </wsp:Policy>
        </wsam:Addressing>
      </wsp:All>
    </wsp:ExactlyOne>
  </wsp:Policy>
  <wsdl:types>
    <xsd:schema targetNamespace="http://tempuri.org/Imports">
      <xsd:import schemaLocation="http://<host>:<port>/PCDData?xsd=xsd0"
namespace="http://tempuri.org/"/>
      <xsd:import schemaLocation="http://<host>:<port>/PCDData?xsd=xsd1"
namespace="http://schemas.microsoft.com/2003/10/Serialization/"/>
    </xsd:schema>
```

```
  </wsdl:types>
  <wsdl:message
name="IDeviceObservationConsumer_Binding_Soap12_CommunicatePCDData_InputMessage"
>
    <wsdl:part name="parameters" element="tns:CommunicatePCDData"/>
  </wsdl:message>
  <wsdl:message
name="IDeviceObservationConsumer_Binding_Soap12_CommunicatePCDData_OutputMessage
">
    <wsdl:part name="parameters" element="tns:CommunicatePCDDataResponse"/>
  </wsdl:message>
  <wsdl:portType name="IDeviceObservationConsumer_Binding_Soap12">
    <wsdl:operation name="CommunicatePCDData">
      <wsdl:input
wsam:Action="http://tempuri.org/IDeviceObservationConsumer_Binding_Soap12/Commun
icatePCDData"
message="tns:IDeviceObservationConsumer_Binding_Soap12_CommunicatePCDData_InputM
essage"/>
      <wsdl:output
wsam:Action="http://tempuri.org/IDeviceObservationConsumer_Binding_Soap12/Commun
icatePCDDataResponse"
message="tns:IDeviceObservationConsumer_Binding_Soap12_CommunicatePCDData_Output
Message"/>
    </wsdl:operation>
  </wsdl:portType>
  <wsdl:binding name="WSHttpBinding_IDeviceObservationConsumer_Binding_Soap12"
type="tns:IDeviceObservationConsumer_Binding_Soap12">
    <wsp:PolicyReference
URI="#WSHttpBinding_IDeviceObservationConsumer_Binding_Soap12_policy"/>
    <soap12:binding transport="http://schemas.xmlsoap.org/soap/http"/>
    <wsdl:operation name="CommunicatePCDData">
      <soap12:operation
soapAction="http://tempuri.org/IDeviceObservationConsumer_Binding_Soap12/Communi
catePCDData" style="document"/>
      <wsdl:input>
        <soap12:body use="literal"/>
      </wsdl:input>
      <wsdl:output>
        <soap12:body use="literal"/>
      </wsdl:output>
    </wsdl:operation>
  </wsdl:binding>
  <wsdl:service name="DeviceObservationConsumer_host">
    <wsdl:port name="WSHttpBinding_IDeviceObservationConsumer_Binding_Soap12"
binding="tns:WSHttpBinding_IDeviceObservationConsumer_Binding_Soap12">
      <soap12:address location="http://<host>:<port>/PCDData"/>
      <wsa10:EndpointReference>
        <wsa10:Address>http://<host>:<port>/PCDData</wsa10:Address>
      </wsa10:EndpointReference>
    </wsdl:port>
  </wsdl:service>
</wsdl:definitions>
```

- **Alerts**

The following Web Service must be invoked by Hydra to provide technical emergency alerts which will be sent in parallel to the CA for displaying and to the EPR for storage.
http://<host>:<port>/TechAlert

The WSDL (available at http://<host>:<port>/TechAlert?wsdl) is analogous to the one describing the Measurements WS, except where the URL of this specific WS is used (i.e. in the tags xsd:import schemaLocation, soap12:address location and wsa10:Address).

In order to update alerts, the Consumer Applications shall invoke the following Web Service:
http://<host>:<port>/AlertUpdate

The WSDL shall be available at:
http://<host>:<port>/AlertUpdate?wsdl

For this service and for the other exposed by the Mediator described in this section, the WSDL is not reported, since it is absolutely similar to the one of measurements flow, apart for the specific URL.

- **Queries**

The Mediator allows the Consumer Applications to interrogate the EPR and get measurements or alerts from it. For the queries the following Web Service hosted on the Mediator must be invoked by the CA:
http://<host>:<port>/Query

The relevant WSDL shall be available at:
http://<host>:<port>/Query?wsdl

- **Parameter updates**

The Mediator allows the Consumer Applications to update parameters (e.g. thresholds) in the EPR. For such update the following Web Service must be invoked:
http://<host>:<port>/UpdateParams

The correspondent WSDL shall be available at:
http://<host>:<port>/UpdateParams?wsdl

8.4 EPR system configurations

8.4.1 Application setup

The EPR application is composed by a WAR executable and a database schema.
The WAR executable has to be deployed to Tomcat application server release 5.5.26 or higher (releases 6 and 7 are supported too).
The application needs Java 1.6.

In order to configure the EPR database connection it is necessary to edit the file:
Tomcat_home/conf/Catalina/localhost/EPR-WS.xml

It is necessary to provide the database connection information:

```
<Resource auth="Container"
          name="jdbc/EprDS"
          type="oracle.jdbc.pool.OracleConnectionPoolDataSource"
          factory="oracle.jdbc.pool.OracleDataSourceFactory"
          url="jdbc:oracle:thin:usr/psw@server:port:sid"/>

  <Resource auth="Container"
          name="jdbc/VirtualEprDS"
          type="oracle.jdbc.pool.OracleConnectionPoolDataSource"
          factory="oracle.jdbc.pool.OracleDataSourceFactory"
          url="jdbc:oracle:thin:usr/psw@server:port:sid"/>
```

The database schema has to be deployed on Oracle 11.2 RDBMS (Oracle 10.2 is supported too). Schema creation scripts (to be run by Oracle System user) and Oracle impdp repref.dmp file are provided.

8.4.2 Environment and Data

The start-up database schema is pre-configured. Environment configuration is necessary in order to store data. Patients, homes and rooms have to be registered inside 'paziente' and 'struttura' tables. These tables can be filled with a specific data entry procedure.

The following tables contain column list and their attributes and description; mandatory fields are in **bold**.

8.4.2.1 Paziente table

Paziente table contains information describing assisted citizen.

EXTERNAL_ID	VARCHAR2(20)	Patient ID (unique identifier taken from PID-3 field)
COGNOME	VARCHAR2(100)	Family name
NOME	VARCHAR2(100)	Given name
SESSO	VARCHAR2(1)	Sex (M/F)
CODFISC	VARCHAR2(40)	Tax ID number
TESSERA	VARCHAR2(40)	Health insurance ID
DATANASC	VARCHAR2(8)	Birth Date
IPROVNASC	VARCHAR2(3)	Birth Place ISTAT code (County)
ICOMNASC	VARCHAR2(3)	Birth Place ISTAT code (Town)
IPROVRES	VARCHAR2(3)	Home Town ISTAT code (County)
ICOMRES	VARCHAR2(3)	Home Town ISTAT code (Town)
LOCALRES	VARCHAR2(60)	Home Locality
INDRES	VARCHAR2(40)	Home Address
NCIVRES	VARCHAR2(12)	Home Dwelling Number
CAPRES	VARCHAR2(5)	Home Postal Code
NUMTEL1	VARCHAR2(400)	Telephone 1
NUMTEL2	VARCHAR2(400)	Telephone 2
NUMFAX	VARCHAR2(400)	Fax
EMAIL	VARCHAR2(60)	Email
INCASACOD	VARCHAR2(20)	House code. Must refer to a livello 3 record (codice esterno field) from struttura table.

8.4.2.2 Struttura table

Struttura table contains information describing assisted citizen's homes and rooms.
Home description is mandatory, rooms' description is optional depending on room code usage in HL7 messages.

All the home and room codes used in HL7 messages have to be described inside the struttura table.

LIVELLO	NUMBER	3 for homes, 4 for rooms
CODICE_ESTERNO	VARCHAR2(20)	Home/room code (taken from PV1-3/PL1 for home or from PV1-3/PL2 for room)
CODICE_REF (1)	VARCHAR2(20)	Home the room belongs to (codice_esterno)
CODICE_REGIONALE	VARCHAR2(20)	Home/room regional code
NOME	VARCHAR2(64)	Name
DESCRIZIONE	VARCHAR2(128)	Description
IPROV	VARCHAR2(3)	Home ISTAT code (County)
ICOM	VARCHAR2(3)	Home ISTAT code (Town)
INDIRIZZO	VARCHAR2(40)	Home Address
NUMERO_CIVICO	VARCHAR2(20)	Home Dwelling Number
CAP	VARCHAR2(20)	Home Postal Code
TELEFONO	VARCHAR2(40)	Telephone
STABILIMENTO	VARCHAR2(20)	Building
ALA_SCALA	VARCHAR2(20)	Department
PIANO	VARCHAR2(20)	Floor
NOTE_REPERIBILITA	VARCHAR2(2000)	Notes

- (1) - Mandatory for rooms (livello=4), not used for homes.

8.4.3 EPR Services

EPR exposes SOAP Web Services, which conform to the IHE PCD standard definition. The SPP Mediator consumes all the Web Services. There are three Web Services, all having the same schema structure, dedicated to different flows of information.

- **Measurements**

To store devices' measurements or SPP Reasoner's habits model into the EPR the following Web Service must be invoked:

http://host:port/EPR-WS/services/DeviceObservationConsumer_Port_Soap12

The WSDL which describes the service is the following:

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions name="DeviceObservationConsumer"
    targetNamespace="urn:ihe:pcd:dec:2010"
    xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/"
    xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:wsaw="http://www.w3.org/2006/05/addressing/wsdl"
    xmlns:tns="urn:ihe:pcd:dec:2010">
    <wsdl:types>
        <xsd:schema>
            <xsd:import namespace="urn:ihe:pcd:dec:2010"
```

```
schemaLocation="DeviceObservationConsumer.xsd"></xsd:import>
        </xsd:schema>
    </wsdl:types>
    <wsdl:message name="CommunicatePCDData_Message">
        <wsdl:documentation>Communicate PCD Data</wsdl:documentation>
        <wsdl:part name="body" element="tns:CommunicatePCDData" />
    </wsdl:message>
    <wsdl:message name="CommunicatePCDDataResponse_Message">
        <wsdl:documentation>Communicate PCD Data Response</wsdl:documentation>
        <wsdl:part name="body" element="tns:CommunicatePCDDataResponse" />
    </wsdl:message>
    <wsdl:portType name="DeviceObservationConsumer_PortType">
        <wsdl:operation name="CommunicatePCDData">
            <wsdl:input message="tns:CommunicatePCDData_Message"
                wsaw:Action="urn:ihe:pcd:2010:CommunicatePCDData" />
            <wsdl:output message="tns:CommunicatePCDDataResponse_Message"
                wsaw:Action="urn:ihe:pcd:2010:CommunicatePCDDataResponse" />
        </wsdl:operation>
    </wsdl:portType>
    <wsdl:binding name="DeviceObservationConsumer_Binding_Soap12"
type="tns:DeviceObservationConsumer_PortType">
        <soap12:binding style="document"
            transport="http://schemas.xmlsoap.org/soap/http" />
        <wsaw:UsingAddressing wsdl:required="true" />
        <wsdl:operation name="CommunicatePCDData">
            <soap12:operation soapAction="urn:ihe:pcd:2010:CommunicatePCDData"/>
            <wsdl:input>
                <soap12:body use="literal" />
            </wsdl:input>
            <wsdl:output>
                <soap12:body use="literal" />
            </wsdl:output>
        </wsdl:operation>
    </wsdl:binding>
    <wsdl:service name="DeviceObservationConsumer_Service">
        <wsdl:port binding="tns:DeviceObservationConsumer_Binding_Soap12"
name="DeviceObservationConsumer_Port_Soap12">
            <soap12:address location="http://www.example.org/" />
        </wsdl:port>
    </wsdl:service>
</wsdl:definitions>
```

- **Alerts**

To store the alerts generated by Hydra or by the SPP Reasoner into the EPR or to let the CA to update their status the following Web Service must be invoked:

http://host:port/EPR-WS/services/AlertConsumer_Port_Soap12

The WSDL which describes the service is the following.

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions name="AlertConsumer"
    targetNamespace="urn:ihe:pcd:dec:2010"
    xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/"
    xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:wsaw="http://www.w3.org/2006/05/addressing/wsdl"
    xmlns:tns="urn:ihe:pcd:dec:2010">
    <wsdl:types>
        <xsd:schema>
```

```
            <xsd:import namespace="urn:ihe:pcd:dec:2010"
schemaLocation="DeviceObservationConsumer.xsd"></xsd:import>
        </xsd:schema>
    </wsdl:types>
    <wsdl:message name="CommunicatePCDData_Message">
        <wsdl:documentation>Communicate PCD Data</wsdl:documentation>
        <wsdl:part name="body" element="tns:CommunicatePCDData" />
    </wsdl:message>
    <wsdl:message name="CommunicatePCDDataResponse_Message">
        <wsdl:documentation>Communicate PCD Data Response</wsdl:documentation>
        <wsdl:part name="body" element="tns:CommunicatePCDDataResponse" />
    </wsdl:message>
    <wsdl:portType name="AlertConsumer_PortType">
        <wsdl:operation name="CommunicatePCDData">
            <wsdl:input message="tns:CommunicatePCDData_Message"
                wsaw:Action="urn:ihe:pcd:2010:CommunicatePCDData" />
            <wsdl:output message="tns:CommunicatePCDDataResponse_Message"
                wsaw:Action="urn:ihe:pcd:2010:CommunicatePCDDataResponse" />
        </wsdl:operation>
    </wsdl:portType>
    <wsdl:binding name="AlertConsumer_Binding_Soap12"
type="tns:AlertConsumer_PortType">
        <soap12:binding style="document"
            transport="http://schemas.xmlsoap.org/soap/http" />
        <wsaw:UsingAddressing wsdl:required="true" />
        <wsdl:operation name="CommunicatePCDData">
            <soap12:operation soapAction="urn:ihe:pcd:2010:CommunicatePCDData"/>
            <wsdl:input>
                <soap12:body use="literal" />
            </wsdl:input>
            <wsdl:output>
                <soap12:body use="literal" />
            </wsdl:output>
        </wsdl:operation>
    </wsdl:binding>
    <wsdl:service name="AlertConsumer_Service">
        <wsdl:port binding="tns:AlertConsumer_Binding_Soap12"
name="AlertConsumer_Port_Soap12">
            <soap12:address location="http://www.example.org/"/>
        </wsdl:port>
    </wsdl:service>
</wsdl:definitions>
```

- **Queries**

To interrogate the EPR and get measurements or alerts the following Web Service must be invoked:

http://host:port/EPR-WS/services/Query_Port_Soap12

The WSDL, which describes the service, is the following.

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions name="DataSubscription"
    targetNamespace="urn:ihe:pcd:dec:2010"
    xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/"
    xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:wsaw="http://www.w3.org/2006/05/addressing/wsdl"
    xmlns:tns="urn:ihe:pcd:dec:2010">
    <wsdl:types>
```

```
        <xsd:schema>
            <xsd:import namespace="urn:ihe:pcd:dec:2010"
schemaLocation="DeviceObservationConsumer.xsd"></xsd:import>
        </xsd:schema>
    </wsdl:types>
    <wsdl:message name="CommunicatePCDData_Message">
        <wsdl:documentation>Communicate PCD Data</wsdl:documentation>
        <wsdl:part name="body" element="tns:CommunicatePCDData" />
    </wsdl:message>
    <wsdl:message name="CommunicatePCDDataResponse_Message">
        <wsdl:documentation>Communicate PCD Data Response</wsdl:documentation>
        <wsdl:part name="body" element="tns:CommunicatePCDDataResponse" />
    </wsdl:message>
    <wsdl:portType name="DataSubscription_PortType">
        <wsdl:operation name="CommunicatePCDData">
            <wsdl:input message="tns:CommunicatePCDData_Message"
                wsaw:Action="urn:ihe:pcd:2010:CommunicatePCDData" />
            <wsdl:output message="tns:CommunicatePCDDataResponse_Message"
                wsaw:Action="urn:ihe:pcd:2010:CommunicatePCDDataResponse" />
        </wsdl:operation>
    </wsdl:portType>
    <wsdl:binding name="DataSubscription_Binding_Soap12"
type="tns:DataSubscription_PortType">
        <soap12:binding style="document"
            transport="http://schemas.xmlsoap.org/soap/http" />
        <wsaw:UsingAddressing wsdl:required="true" />
        <wsdl:operation name="CommunicatePCDData">
            <soap12:operation soapAction="urn:ihe:pcd:2010:CommunicatePCDData"/>
            <wsdl:input>
                <soap12:body use="literal" />
            </wsdl:input>
            <wsdl:output>
                <soap12:body use="literal" />
            </wsdl:output>
        </wsdl:operation>
    </wsdl:binding>
    <wsdl:service name="DataSubscription_Service">
        <wsdl:port binding="tns:DataSubscription_Binding_Soap12"
name="DataSubscription_Port_Soap12">
            <soap12:address location="http://www.example.org/" />
        </wsdl:port>
    </wsdl:service>
</wsdl:definitions>
```

All the above WSDL include the following XSD.

```
<?xml version="1.0" encoding="UTF-8"?>
<schema                                  xmlns="http://www.w3.org/2001/XMLSchema"
targetNamespace="urn:ihe:pcd:dec:2010" xmlns:tns="urn:ihe:pcd:dec:2010">
  <element name="CommunicatePCDData" type="tns:UnsolicitedObservationResult"/>
  <element name="CommunicatePCDDataResponse" type="tns:GeneralAcknowledgement"/>

  <simpleType name="UnsolicitedObservationResult">
    <restriction base="string" />
  </simpleType>

  <simpleType name="GeneralAcknowledgement">
    <restriction base="string" />
  </simpleType>
</schema>
```

8.5 Exploiting the inCASA Platform for Application Development

WP3 has specified and described the required support and tools to provide to the application developers (APIs to easily create Use Case applications). This mainly relies on the Hydra Middleware that allows developers to create interconnected AmI services for inCASA through a SOA and Model Driven Architecture approach.

The Hydra Middleware can therefore be seen as a computer software that connects software components or applications by consisting of a set of services that allows multiple processes running on one or more machines to interact. The technology behind Hydra evolved to provide for interoperability in support of the move to coherent distributed architectures, which are used most often to support and simplify complex, distributed applications and particularly services advertised by the SARA Service and Semantic Reasoner.

As such, the Hydra Middleware can be seen as to sit "in the middle" between application software that may be working on different operating systems. It is similar to the middle layer of a three-tier single system architecture, except that it is stretched across multiple systems or applications. It is often used to create a homogeneous environment created from otherwise heterogeneous components.

In order to assist application developers in addressing a wide variety of mobile and stationary devices and networks, the Hydra middleware hides device-dependent and network-dependent details and provides comprehensive open interfaces to the display, communication port, input facilities and memory management of each class of device or application service. These are enabled in such way that they appear in the inCASA network as services in a transparent way for the application developer and end use of applications.

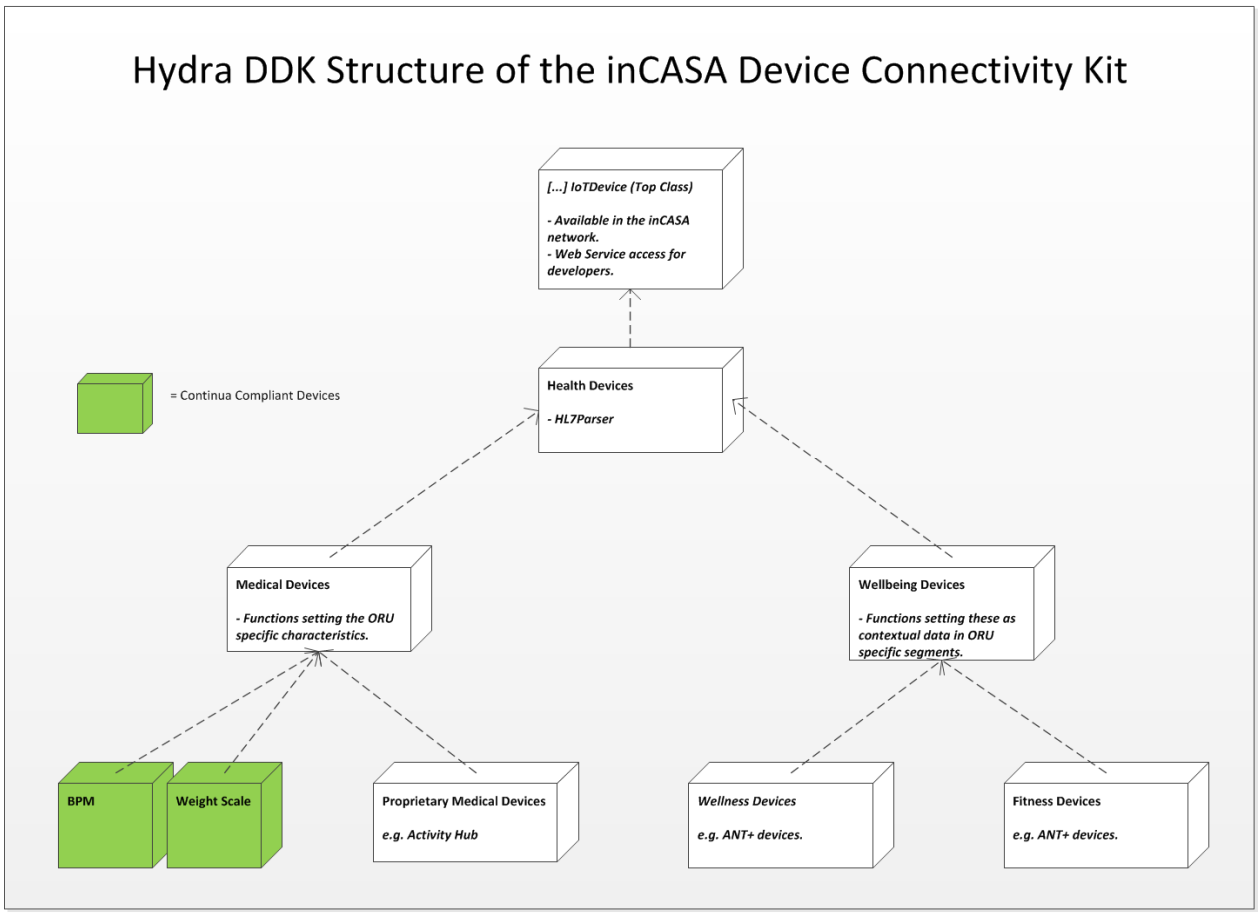

Figure 2 – Initial draft model of the inCASA Device Connectivity Kit.

The deliverable D3.2 explains in chapter 5.4.1.4 "*Using Hydra DDK and SDK for inCASA Module Integration*" how the above described functionalities are open to use for the inCASA developers through the DDK and SDK provided by Hydra.

Besides public Web Services for the inCASA developers, the project has the aim to provide a development kit called the inCASA Device Connectivity Kit. A first draft model is shown in Figure 2. This kit allows for seemingly connecting devices and services based on the patient layer over the inCASA network (i.e. basic and intelligent layers) without deeper knowledge on how to create HL7 messages or publish internal as well as external Web Services. It will use the Hydra DDK and SDK in order to provide a fast way to "plug-n-play" medical and environmental devices and sensors mapping them to relevant IEEE 11073 specialization. All new devices and services will respond to the top class "iotdevice" by their UPnP UDN. The only configuration necessary in the Device Connectivity Kit will possibly be any hardware changes (e.g. dongles or Internet access) but the estimation is that using Web Services the data flow will in a network be infallible.

Documentation of the inCASA Device Connectivity Kit will most appropriately be presented in a separate document specific for the implementation of additional devices or services as well as a cookbook available for external inCASA solution developers.

9 Customizing regional GUI presentations

The GUI representation will be part of the Consumer Applications (CAs) component. Summarizing their functionality CAs will render patient's data and alerts to end users screens offering them also some additional services like appointment arrangement or SMS communication. To support these requirements, CAs will have a bi-directional communication with the Smart Personal Platform (SPP) component including:

1. Call to a web service exposed by the SPP whenever patient data are requested.
2. Call to a web service exposed by the SPP whenever alerts retrieval is requested.
3. Expose a web service to be called by SPP when a new alarm is generated.

All input and output parameters will be HL7 queries or responses respectively.

9.1 Java as Programming language

The Java EE (Enterprise Edition) is a widely used platform for server programming. It differs from the Standard Edition in that it adds packages and libraries in order to deploy fault-tolerant, multi-tier, distributed application written in Java on application servers. Meanwhile, Java provides widely used stubs for SOAP web service calls that will be needed to communicate with SPP.

9.2 Application Server selection

The use of an Application Server is mandatory for the CAs component. As already explained, apart from calling SPP services, CAs are needed to expose services to be called upon alarm generation in order to proceed with the pre-defined relevant update (Screen Alert, SMS to the relatives etc). Moreover, CAs should not be considered as stand-alone applications since this would eliminate their portability. In the context of inCASA project, a dynamic web page accessed by professional end users is required.

There are several choices related to the selection of the application server. Some of them are:

- Glassfish open-source application server (Sun Microsystems), which is fully Java EE 6 Certified. It comes under two licenses (CDDL & GPL).
- JBoss AS (JBoss). A cross-platform, open-source application that supports the Java EE platform (Web Profile).
- Apache Tomcat (Apache Software Foundation) is an open-source application server that implements the Servlet and JavaServer Pages specifications by Sun Microsystems.
- Axis2 under Tomcat (Apache Software Foundation) is a core engine for developing web services. It provides the capability to add web services interfaces at web applications, but it can be used as a stand-alone application server also. Tomcat provides the Java implementation of Axis2.

Our final selection is Glassfish Open-Source 3.1. This edition of Glassfish is fully Java EE 6 certified. It is a very good solution for developing Java Applications and Web Services in Java. It is also highly customizable, provides a variety of modules and offers a user-friendly administration platform.

9.2.1 Web Framework Selection

Some of the prevailing web frameworks used to build a web application is stated below:

- Struts
- Cocoon
- Vaadin

- Spring
- JSF
- Google Web Toolkit

Concerning our web GUI implementation, we chose the Spring Framework. As many other web frameworks it supports the MVC (Model-View-Controller) software architecture. Spring provides also a variety of modules, such as the remote access framework (web services), the inversion of control container, data access and transaction manager. The remote access framework provides integration with the Apache Axis web services framework

9.3 IDE Selection

The IDE of our choice will be the NetBeans IDE 7.0. NetBeans provides an easy-to-use and code environment. It provides the tools to build, configure and deploy a Web Application (written in Java in our case) to the application server of our choice. It provides full integration with the Glassfish application server, but it also supports various other application servers.

10 Deployment of the inCASA Business Logic

In WP4, the task 4.4 dealing with the Reasoning and Learning System of inCASA will ensure the enhancement of different modules of business logic, ensuring that each module is able to successfully play its role. This task will later feed knowledge into WP6 task 6.3 (Pilot Enhancement and Maintenance) that in turn will customise the inCASA solution to fit the language(s) and any other specific requirements of each pilot site by exploiting the capabilities of the configurability found in the business logic rules and ontology of the platform.

10.1 inCASA Learning and Reasoning System

inCASA reasoning system includes a Semantic Reasoner, which allows modelling complex domains and making formal inferences, and an additional module for processing capabilities used for habits model computation.

The semantic Reasoner is the module that performs inferences for context recognition and reaction based on the semantic representation of the system; it is in charge of managing the knowledge base and of updating it in case of system changes.

Every change in the system is communicated to the reasoning module from the Mediator as events, which are communicated through web services and queued for processing. The communication is based on the logical model representation, and can express appearance of new entities in the system, requests to delete entities, or changes to existing entities.

The knowledge base is updated according to queued events. Sensed context change may trigger changes in the ontology model composition, in aggregate values, and in the set of available reasoning services.

The semantic Reasoner integrates Jena Semantic Web, using it in Microsoft .NET environment through IKVM.NET. Jena's APIs are exploited to create and manage the ontology model, to execute queries on it and to make inferences.

The Reasoner uses W3C Semantic web standard languages (RDF, OWL, and SPARQL), while Jena rules have a syntax very close to SWRL candidate recommendation..

The main tasks of the semantic Reasoner are Monitoring and Alerting activities. Home Comfort and Technical Alarms (UC-TC-3) are alerts triggered when an abnormal situation occurs in the user's premises or when comfort indicators (humidity / temperature measurements) deviate significantly from expected values. Habit-related Alert (UC-TC-2) describes alarming deviations from user habits; some types of alerts are considered "absolute warning" as they are triggered whenever a user action causes concern, i.e. he/she goes out without closing the door, and this is independent of his/her specific habits. A separate processing flow is implemented for building user profile used for habit monitoring (UC-TC-1).

Section 10.1.1 describes how rules are organized and managed in the semantic Reasoner, while Section 10.1.2 describes the model that has been defined for characterizing normal user habits. Finally, Sections 10.1.3 and 10.1.4 describe how data from sensors and the normal habits model are used for detection anomalies.

10.1.1 Rules storage and handling

In the semantic Reasoner, the logic is grouped in "context awareness services". These are the logical entity of the system which are dedicated to satisfy the specific needs and to handle the special functionality based on the context provided by the system.

Each context awareness service is represented both in a XML model, and in OWL for the more expressive internal ontological representation. A context awareness service has associated context recognitions and actions rule-sets. Rules are stored in external files, so that the business logic is clearly separated from the code.
The files containing the rules for a certain type of context awareness service are specified in an initialization file for the service itself.

Rules associated to a certain type of context awareness service are loaded into the system when the related service is instantiated.

10.1.2 Periodically retrieving data and populating a behaviour model

All the data from the system are going to be collected by sensors network located at the users home and transferred by Hydra middleware to the platform for further analyses. Hydra provides to the SOAP based web service exposed by the Mediator the data gathered from the sensor network in real time. All the data after being stored immediately in the EPR data repository are sent to the Reasoner, provided that it is registered to receive that kind of data.

Received data are queued for processing. When these events are handled, the knowledge base is updated accordingly. Sensed context change may require simply changing a property (e.g. currently sensed value), or can require creating/deleting individuals in the ontology. The Reasoner, besides monitoring the current situation, updates counters and organizes information so that they can be used for habits model population.

A separate module in the reasoning system is responsible for aggregating historical data and deriving a behavioural profile for each user.

10.1.3 inCASA behavioural model, routines and remote adjustments

The user behavioural model is composed of indicators of the actions performed every day during the week. For instance, it includes the usual wake-up time, the sleeping duration, the average time spent watching TV, an indicator of the amount of user movements, an indicator of the normal usage of the fridge.

In the first monitoring timeframe (e.g. the first two weeks of the pilot) the data from the sensor networks in the houses will be collected in order to have data for building a behavioural profile about each user; row data and aggregated habit profile will be stored in the EPR data repository for the further comparison.

In a second time period (“tuning timeframe”) the data from the sensor networks in the houses will be collected and analysed in order for a verification of the correctness of the initial user behaviour.
After the first profiling and the tuning, the reasoning module will use the normal habit profile of each user to highlight deviation from habits. Deviations will trigger habit-related alert, which shall be handled by a human operator, who shall check user conditions, and, in case of a false alarm, adjust the habit model parameters to reflect real or changed user habits.
No automated adjustments will be done.

10.1.4 Habits analysis for detection of anomalies

Habits analysis and monitoring involves comparing data observed over a certain timeframe against a normal behavioural model in order to detect changes in habits that can indicate a problem such as a deterioration of health conditions or that can provide related information to professional caregivers. This task is executed periodically, with a periodicity, which can be daily, weekly or monthly.

Periodically habit indicators are aggregated, analysed, and compared with the related facets of the normal behavioural model. Indicators can be average values, maximum or minimum “normal”

values or could be other (e.g. based on percentiles) if pre-pilot data analysis suggests that other indicators are more appropriate.
The Reasoner retrieves historical data from the EPR by querying it, through the Mediator, by an appropriate HL7 QSB^Z02^QSB_Q16 query message.

10.2 Emergency management

This section describes how SPP deals with detected anomalies. Use cases the SPP has to implement are not tele-emergency use cases, and there are no real time requirements.

10.2.1 Risk management, rules and generated events

As a result of the application of the reasoning, alert events can be generated. The Reasoner triggers and manages alerts based on the results of inferences it executes. Alerts can have different types, priorities, severities and its state can be updated as the situation evolves.

An example of the XML alert representation is shown below.

```
<?xml version="1.0" encoding="utf-8"?>
<Alert xmlns="http://tempuri.org/IoT_inCasa_XMLSchema.xsd"
             xmlns:iot="http://tempuri.org/IoT_inCasa_XMLSchema.xsd">

  <ID iot:Changed="true" iot:Mode="r">S_1234</ID>
  <PersonID iot:Changed="true" iot:Mode="rw">01234567890123456789</PersonID>

  <AlertType iot:Changed="true" iot:Mode="irw">UnusalBedPermanence</AlertType>
  <AlertSource iot:Changed="true" iot:Mode="irw">SPP_ALARM</AlertSource>
  <AlertState iot:Changed="true" iot:Mode="irw">inactive</AlertState>
  <AlertPhase iot:Changed="true" iot:Mode="irw">reset</AlertPhase>
  <AlertPriority iot:Changed="true" iot:Mode="irw">PM</AlertPriority>
  <AlertSeverity iot:Changed="true" iot:Mode="irw">H</AlertSeverity>

  <AlertEvent iot:Changed="true"
iot:Mode="irw">Bed_Permanence_Moderately_Higher</AlertEvent>
  <AlertDetails iot:Changed="true" iot:Mode="irw">User permanence in bed is
higher than usual, in the last  week.</AlertDetails>

  <DeviceID iot:Changed="true" iot:Mode="irw">D001234</DeviceID>

  <AlertStartTime iot:Changed="true" iot:Mode="irw">1900-01-
01T01:01:01</AlertStartTime>
  <AlertLastUpdateTime iot:Changed="true" iot:Mode="irw">1900-01-
01T01:01:01</AlertLastUpdateTime>
  <AlertEndTime iot:Changed="true" iot:Mode="irw">1900-01-
01T01:01:01</AlertEndTime>

</Alert>
```

Table 1 - highlights the mapping between the HL7 and the XML representation.

HL7	XML (Alert)
PID-3	PersonID
OBR-3	ID
OBX-18	DeviceID
OBR-4	AlertSource
OBX-3	AlertType
OBX-4	---
OBX-5 Facet 1	AlertEvent
OBX-5 Facet 2	can be used by Hydra to indicate the subsystem of the sensor that generated the alarm
OBX-5 Facet3	AlertPhase
OBX-5 Facet4	AlertState
OBX-8	AlertSeverity ~ AlertPriority ~ ---
OBX-14	AlertStartTime AlertLastUpdateTime AlertEndTime
NTE-3	AlertDetails (if coming from Reasoner) contains operator comments (if coming from NTUA CA)

Table 2 - Mapping between fields in HL7 and XML alert representation.

For the AlertSeverity, the alarms abbreviation indicated in the IHE specifications (IHE PDC Technical Framework Supplement 2008-2009–Alarm Communication Management) are used:

Abnormality Type	Abbreviation
Normal, not abnormal	N
Below low normal	L
Below lower panic limits	LL
Above high normal	H
Above higher panic limits	HH
Abnormal (for non-numeric results)	A

Table 3 - Alert severity enumerations.

For the AlertPriority, the abbreviations indicated in the IHE specifications for alarm management are used:

Alarm Priority	Abbreviation
no-alarm	PN
low priority	PL
medium priority	PM
high priority	PH

Table 4 - Alert priority enumerations.

The Event phase could be one of the following values: *tpoint*, *start*, *continue*, *end*, *update*, *escalate*, *de-escalate*, *reset* (as per IHE ACM specifications). The Alarm state could be one of the following values: *inactive*, *active*, *latched* (as per IHE ACM specifications).

10.2.2 Workflow storage and handling of anomalous issues

The workflow of a detected anomalous situation is defined in the rules associated to a certain type of services, and it may depend on the specific scenario.

General monitoring

In general, the workflow for an alarm directly generated due to a generic issues is shown in Figure 3, which highlights states and phase transitions.

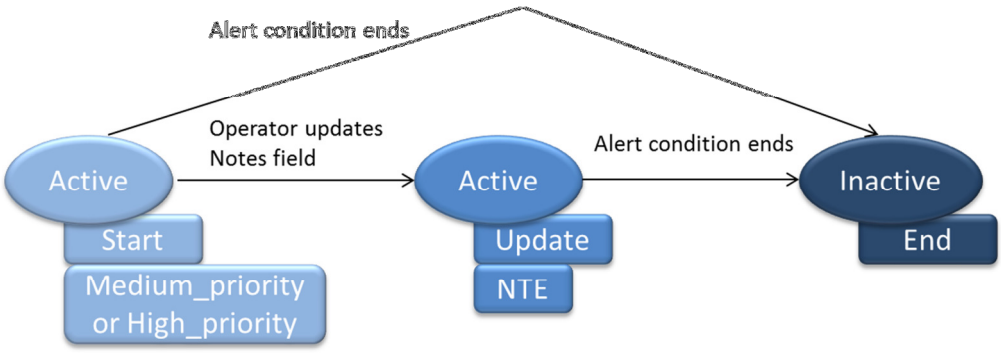

Figure 3 - Alert state/phase transitions in a generic sensor monitoring.

The alert is generated when a generic sensor (e.g. temperature / humidity sensor, indoor movement sensor, etc) detects an anomalous event: its state is Active, its phase is Start, and the priority can be medium or high.

The Consumer Application will show the alert to the operator that can write a note and also mark the alert as handled or forwarded to social services. These operator's actions make the alert change to an Update phase. The alert field NTE can be used to store the mark of the operator on the alert (handled/forwarded) and the written notes.

When the situation comes back to normality as per sensor's measurements, the alert is definitively closed (its state becomes Inactive and its phase becomes End).

For alerts deriving from unexpected level of changes reported by habits monitoring the workflow is similar. The alert is generated when a generic sensor monitoring habit changes (e.g. changes in bed staying, in TV watching, etc), detects over a given report time period (a week or a month) significant habit changes: its state is Active, its phase is Start, and the priority can be medium or high.

Escalation and monitoring anomaly persistence

In other scenarios the workflow for handling of anomalous includes the possibility the priority is escalated if the situation worsen or if it persists for a certain time.

As an example from the implementation of the logic for the monitoring of the Door Opening/Closure Monitoring in the ATC pilot, the possible state/phase transitions of the associated alerts is shown in Figure 4.

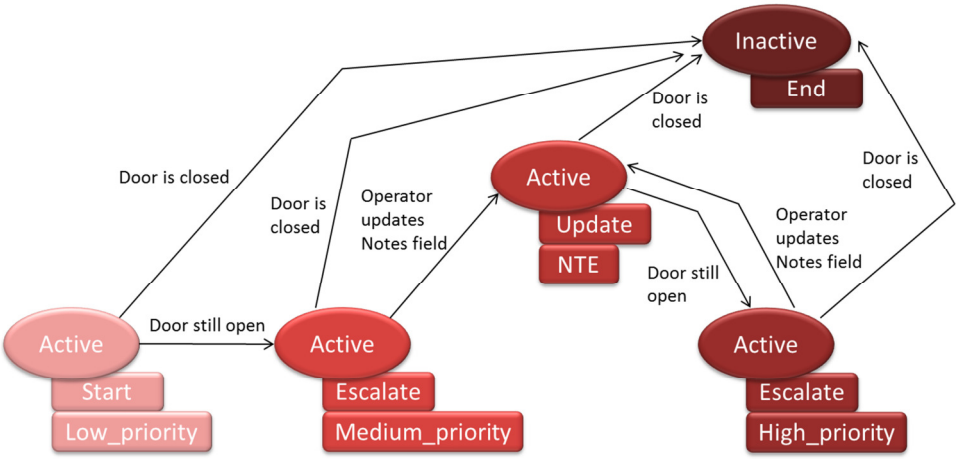

Figure 4 - Alert state/phase transitions in scenario of Door Opening/Closure Monitoring.

The alert is generated when the door remains open for X time (e.g. 30 min): its state is Active, its phase is Start, and the priority is low. For indicating the severity of the alert, we use the Abnormality Type field. The timestamp of the alert generation is used for the AlertStartTime property of the XML alert model (see Section 10.2.1).

If the door is closed within another X time, its state becomes Inactive (i.e. Off) and its phase becomes End. If, on the contrary, the door is still open after another X time, the alert changes its state in Escalate and its priority becomes Medium. In this case, the Consumer Application shows the alert to the ATC operator that can write a note and also mark the alert as handled or forwarded to social services. These operator's actions make the alert change to an Update phase. The alert field NTE can be used to store the mark of the operator on the alert (handled/forwarded) and the written notes.

If the alert is still not closed after another Y time, its priority becomes high. The operator can still write a note associated to the alert, which causes the alert to move to an Update phase. Each alert state/phase transition is "registered" by associating the related timestamp to the AlertLastUpdateTime property of the XML alert model. The transition to the Inactive state (i.e. the definitive closure of the alert) is "registered" by associating the related timestamp to the AlertEndTime property of the XML alert model.

11 Test Notations

The final deliverable for WP4, i.e. D4.3, was created with the aim on delivering descriptions on the implementation of inCASA's Advanced Monitoring System. For the concerned service types (i.e. basic and intelligent layers), the field trials in inCASA will highlight possible priorities and needs that are estimated to have an impact on the ultimate inCASA service model especially when it comes to Advanced Monitoring System configuration and development by services.

D4.1 and D4.2 confirmed the enhancement of the inCASA project objectives by realising and testing in specific pilots efficient integrated care systems that combine innovative technological platforms for ubiquitous communication, advanced healthcare monitoring and state of the art domotic systems. D4.3 will follow this line and further elaborates on any upcoming improvements for Advanced Monitoring in the inCASA system.

Integrating software subsystems within the inCASA SPP and the layers for basic and intelligent services sets a couple of actions to be taken. It involves the procedures of integrating these services into the inCASA platform where they can serve to their functionality and the process to do so is exemplified inside of this document. Each specific procedure of the subsystems is here described fulfilling the inCASA objectives. These actions will assist both the inCASA developing partners as well as future platform and application developers and end users at the service provider's side to understand how the system performances met for the project stated required specifications and quality attributes and map this knowledge to future implementations. For now, these results are presented here for later validation according to the criteria and needs for adjustments of the project user requirements specifications and design specifications found in chapter 12.

11.1 Aspects of Data Flow

It is already known by D4.1 that the data flow occurring throughout the inCASA platform is requires service related control by parallel bidirectional communication. D4.2 stated that this communication pathway is optional for the inCASA developers but where the primary communication must provide supported delivery of communicated data.

In D4.3. the data flow is rather dynamic as it allows for interconnected services to occur. Data can also be separately defined to meet the requirements of any proprietary solution but for the inCASA project and its platform solution the communication technology will be based on HL7 messages and Web Service calls.

11.2 Tested Devices and Applications

Both the SARA Services and the Semantic Reasoner residing in the inCASA SPP module strongly indicate on the wide adopted interoperability between different modules strived for inCASA where a future market for products and services for ICT and ageing is still in its infancy due to low market awareness and visibility, lack of standards and interoperability eventually will be met. As such, the tested software services on its basic and intelligent layers show that the project, by the implemented diverse support of appliances and services, will guarantee interoperability and modularity across the different system solutions, thereby decreasing costs to the end users, enhancing economy of scale, and contribute to the flourishing of the market for ICT and ageing.

D4.3, likewise D4.2, assists the Pilot Scheme Set Up phase by addressing the issue on integrating software subsystems, here the SARA Services, the Semantic Reasoner (SPP) and the Hydra Middleware. The tested directives of software services prove how easy integration should be in the inCASA platform and set the standards on how to proceed in WP5 and WP6.

11.3 Integration Risk Analysis

As the final deliverable of WP4, D4.3 also concatenates the previous one, i.e. D4.1 and D4.2 in a conceptual representation of the success of the solution implementation for inCASA. Any failure at this level would imply a moderate to high risk of the inCASA architecture, as the software engineering process so far should have proven as an integral part of the inCASA development. This architecture need to be able to react to change where integration of new and unproven subsystem developments balances the design of the inCASA platform between tools and components. The major integration risk is estimated to take position at the transboundary interfaces in the inCASA platform where platform extensions or new solutions adopted to the inCASA does not meet the technologies constraints (e.g. Web Service interfaces). The counteract is expected to be accommodated by the inCASA Device Connectivity Kit granting further independent and interoperable development of a scalable inCASA solution by any external innovators.

11.4 Re-designed specification

A re-designed solution implementation specification is regarded as not necessary.

12 Conclusion (ALL)

This deliverable began with chapter 2 that described the expected prerequisites and an issue regarding the Advanced Monitoring System implementation and the chosen approach was confirmed as reliable. The related WP4 tasks are shown to be well suitable for this deliverable and its intention which has been to describe the implementation of an Advanced Monitoring System based on the structured network approach and a SOA model found in D4.1 as well as the device and software specific remote implementation in D4.2. A number of significant driving requirements (in this document seen as quality attributes) from WP3 has been considered and established with respect to the contents of this document. The main quality attributes have been realised through the features found in the inCASA basic and intelligent layers subsystems. These are:

- Semantic Reasoner performs the intelligence behind the data derived from the Activity Hub and includes it in the various service components of the SPP.
- SARA Service is able to receive medical data from the client via the Hydra SOAP tunnel.
- Hydra assists in the development of home automation and allows for distributed and remote control of domestic applications.

The referred work tasks for WP4 are considered to coordinate with the content of this deliverable. D4.3 has managed to incorporate Task 4.3 by customizing both the EPR and the applications built on top of the SPP services collecting person specific data. It has also dealt with Task 4.4 where these services enhanced the business logic by introducing habits modelling and data comparison. Finally, it has also dealt with Task 4.5 which covered the implementation of the operating central in term of remote healthcare provider platform services finalization where this deliverable's work has been based on the extension for the service exploitation defined in both D4.1 and D4.2.

The test conducted throughout the Advanced Monitoring System implementation will hereafter feed knowledge and implemented technology to WP5 and WP6. This will possibly give expected return in terms of refined or re-designed implementation specification.

As a conclusion we can see that the adoption of distributed intelligence in home management and monitoring will definitely provide a safer environment and help daily living activities. For that reason, the consortium is currently continuing on the research and development of the inCASA middleware technology for SOA, device connectivity and applications Web Services as well as the inCASA server side ambient intelligence solutions for Consumer Applications in both healthcare and facility management. The future market will benefit from the pre-existing infrastructure found in inCASA, both in terms of common types of technology and from the interoperability with other services and shared data.

13 Glossary

WP	Work Package
ICT	Information Communication Technologies
SPP	Smart Personal Platform
HIS	Healthcare Information System
WS	Web Service
P2P	Peer-To-Peer
LAN	Local Area Network
HID	Hydra ID
HSN	Home Sensor Network
HMS	Human Monitoring Sensors
DIM	Domain Information Model
WLAN	Wireless Local Area Network
SME	Small Medium Enterprise
GPRS	General Packet Radio Service
HL7	Health Level Seven
IHE	Integrating the Healthcare Enterprise
SQL	Structured Query Language
OSGi	Open Services Gateway initiative framework
PHS	Personal Health Sensor
BSN	Body Sensor Network
CA	Consumer Application

14 References

[1] S. Asanin, T. Brodén, M. Ahlsén, J. Simón, R., and A. Sikora, "D4.1 Core Monitoring System Implementation," inCASA Project – 250505, Report D4.1, 2011-06-30 2011.

[2] S. Asanin, J. Simón, R., and A. Sikora, "D4.2 Remote Monitoring Device Implementation," inCASA Project – 250505, Report 2011-07-15 2011.

[3] M. Rosin, "D3.2 Reference Architecture iteration 1," inCASA Project – 250505, Report 22-04-2011 2011.

[4] G. Lamprinakos, "D3.1 System and Functional Specifications," inCASA Project – 250505, Report 18-04-2011 2011.